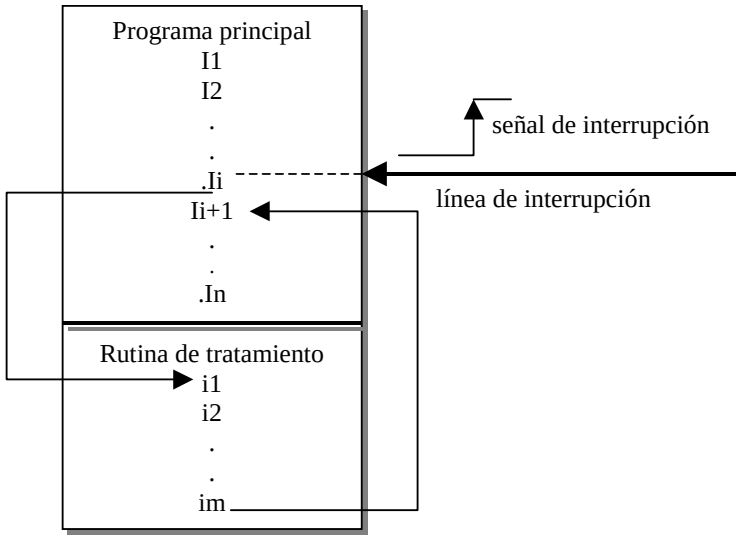
Tema 9: Interrupciones

1. E/S por interrupción: gestión de interrupciones
2. Tipos de sistemas de interrupciones: prioridades
3. Enmascaramiento de interrupciones.
4. Anidamiento de interrupciones
5. Ejemplos

1. E/S por interrupción: gestión de interrupciones

En la E/S programada el procesador tiene que esperar un tiempo considerable a que el módulo de E/S esté preparado para realizar la operación. El procesador espera comprobando repetidamente el estado del módulo de *E/S*, degradándose significativamente el rendimiento de la *CPU*. Para evitar este inconveniente se introdujo el sistema de interrupciones en los procesadores.

Básicamente una interrupción viene determinada por la ocurrencia de una señal externa que provoca la bifurcación a una dirección especifica de memoria, interrumpiendo momentáneamente la ejecución del programa. 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.

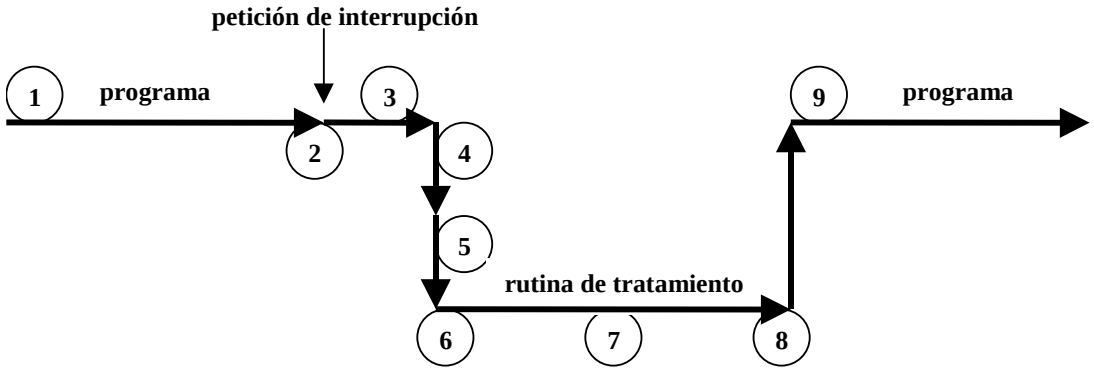


Podemos, pues, ver una interrupción como un salto a subrutina (*rutina de tratamiento*) ocasionado por una señal externa, y no por una instrucción del programa. De esta forma se pueden eliminar los tiempos muertos de consulta de la *E/S* programada

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 natural del programa. De esta forma se 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 de consulta y salto introducidas detrás de cada instrucción útil del programa. Sería algo así como una *E/S* programada en la que en lugar de preguntar en el mismo punto del programa por el estado del periférico, se replica la consulta por todo el programa, intercalándose con la ejecución de las instrucciones útiles. Pues

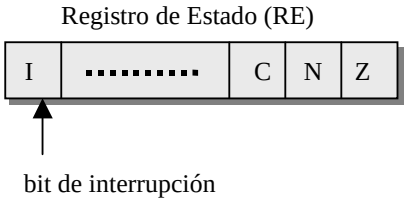
bien, un sistema de interrupciones podemos verlo como la integración en hardware del supuesto software anterior, es decir, la integración de la consulta y posible salto dentro de la ejecución de cada instrucción del repertorio.

El mecanismo de interrupción de un procesador tiene que implementar todas las medidas que hagan que se pueda bifurcar a la rutina de tratamiento y recuperar el estado del programa interrumpido cuando la rutina finaliza su ejecución. En el siguiente esquema se detallan las fases que tienen lugar en la atención a una interrupción producida por algún dispositivo periférico:

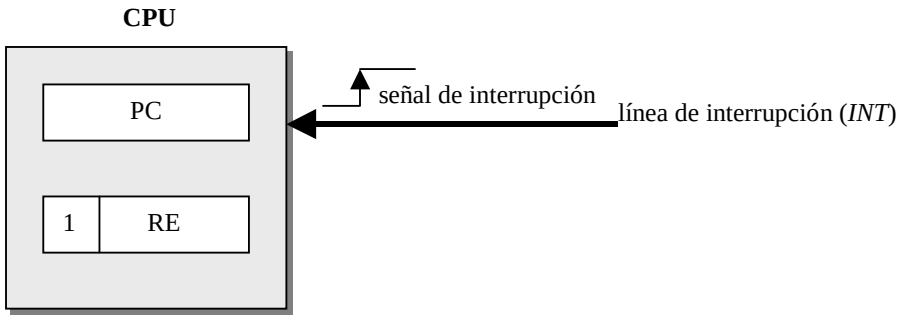


(1) El programa en ejecución (*CPU*) activa el sistema de interrupciones utilizando instrucciones que operan (ponen a 1 y a 0) sobre el bit de capacitación de las interrupciones *I* del registro de estado (*RE*):

STI I <--1; *CLI I <-- 0*

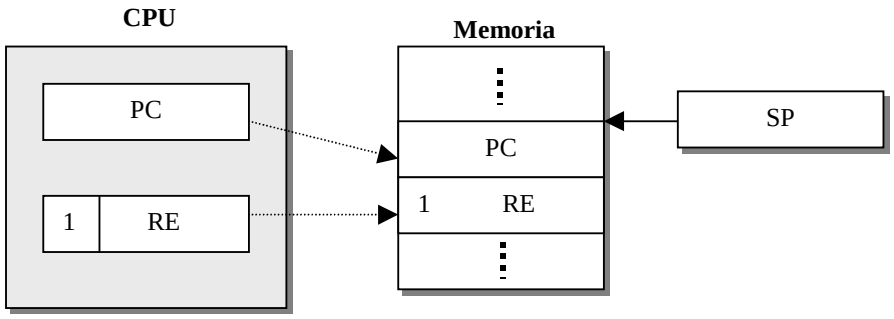


(2) Se produce la petición de interrupción por parte de algún dispositivo periférico en un instante de tiempo impredecible para la *CPU*.

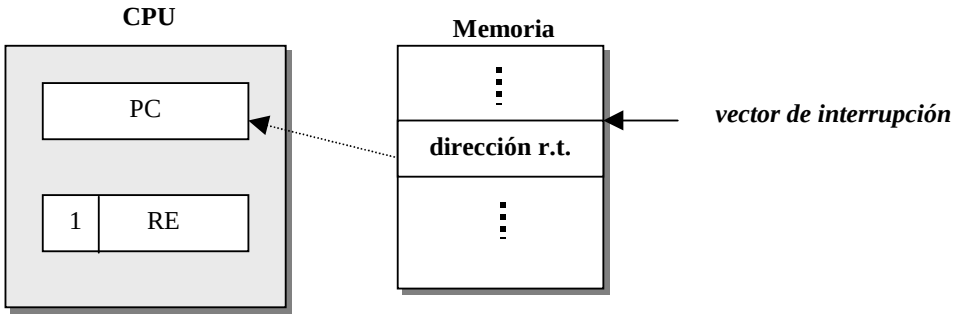


(3) La *CPU* finaliza la ejecución de la instrucción en curso.

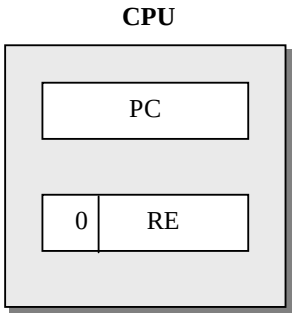
(4) La *CPU* salva automáticamente el estado en la pila, es decir, el contador de programa (*PC*) y el registro de estado (*RE*):



(5) La *CPU* obtiene la dirección de la rutina de tratamiento a partir del *vector de interrupción* (*VI*), que usualmente se ubica en memoria.



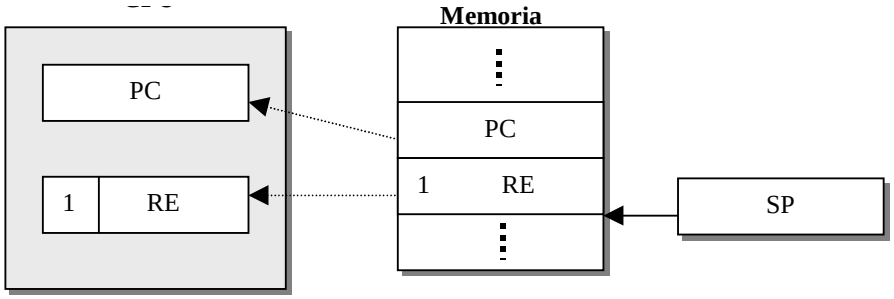
(6) La *CPU* descapacita las interrupciones (*I = 0*) para que durante la ejecución de la primera instrucción de la rutina de tratamiento no se vuelva a detectar la misma interrupción y provoque un bucle infinito.



(7) La *CPU* ejecuta la rutina de tratamiento de la interrupción que realiza lo siguiente:

- Salva en la pila los registros a utilizar
- Realiza la operación de Entrada/Salida
- Restaura desde la pila los registros utilizados

(8) Finaliza la rutina de tratamiento con la ejecución de la instrucción de retorno de interrupción (*RTI*), que restaura automáticamente el estado de la *CPU* desde la pila y vuelve al programa interrumpido:



(9) La *CPU* continúa la ejecución del programa interrumpido, quedando las interrupciones capacitadas automáticamente al recuperarse el valor *I = 1* del *RE*.

2. Tipos de sistemas de interrupciones: prioridades

Clasificaremos las interrupciones atendiendo a dos criterios independientes: la fuente que produce la interrupción, y el modo de obtener la dirección de la rutina de tratamiento o vector de interrupción.

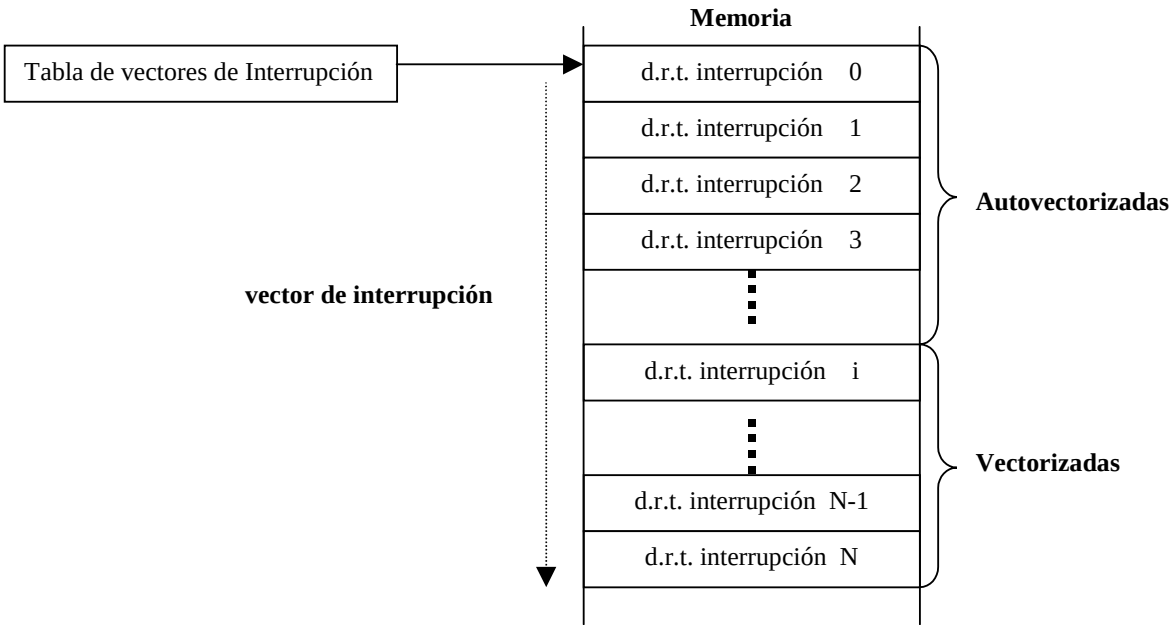
a) <u>Atendiendo a la fuente que produce la interrupción</u>:

- **Interrupciones hardware**
  - <u>Internas</u>: producidas por la *CPU*
    - división por cero
    - desbordamiento
    - instrucción ilegal
    - dirección ilegal
    - logaritmo de cero
    - raíz cuadrada de negativos
    - etc.
  - <u>Externas</u>: producidas por los dispositivos de *E/S*
    - vectorizadas
    - no vectorizadas
- **Interrupciones software**: producidas por la ejecución de instrucciones de la *CPU*.

b) <u>Atendiendo al modo de obtener el vector de interrupción</u>:

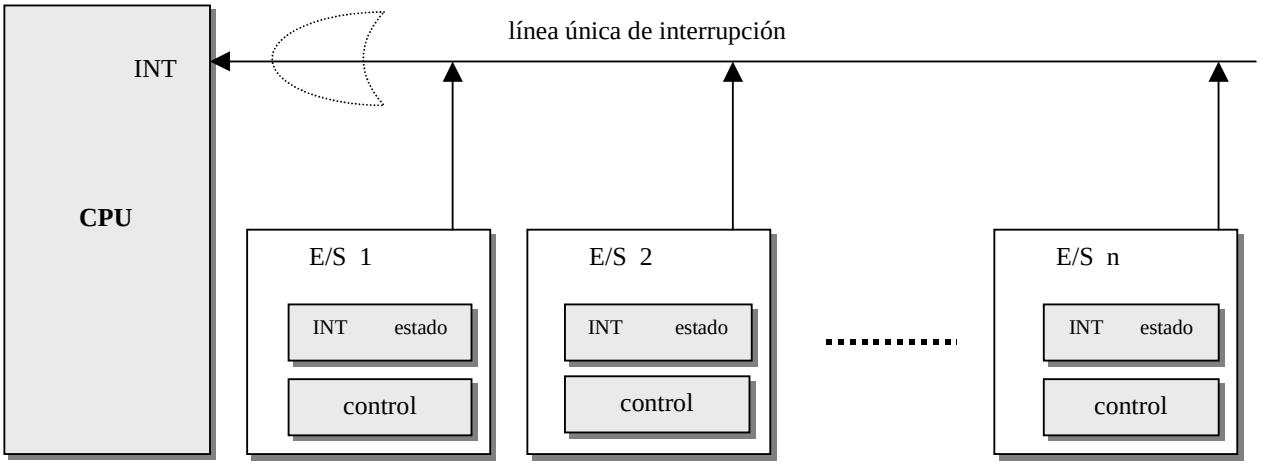
- <u>Interrupciones autovectorizadas</u>: el vector de interrupción es fijo, una posición de memoria asociada a la línea de interrupción.
- <u>Interrupciones vectorizadas</u>: el vector de interrupción o parte de él lo suministra el propio periférico cuando se le reconoce la interrupción.

  El método usual de obtener la dirección de la rutina de tratamiento de la interrupción en ambos casos es a través de la tabla de vectores de interrupción, tabla que contiene las direcciones de las rutinas de tratamiento de cada interrupción.

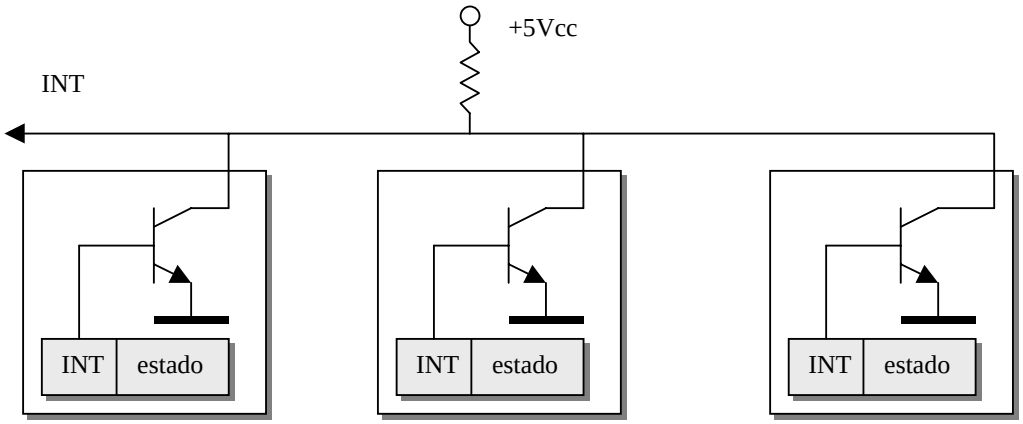


2.1 Interrupciones autovectorizadas

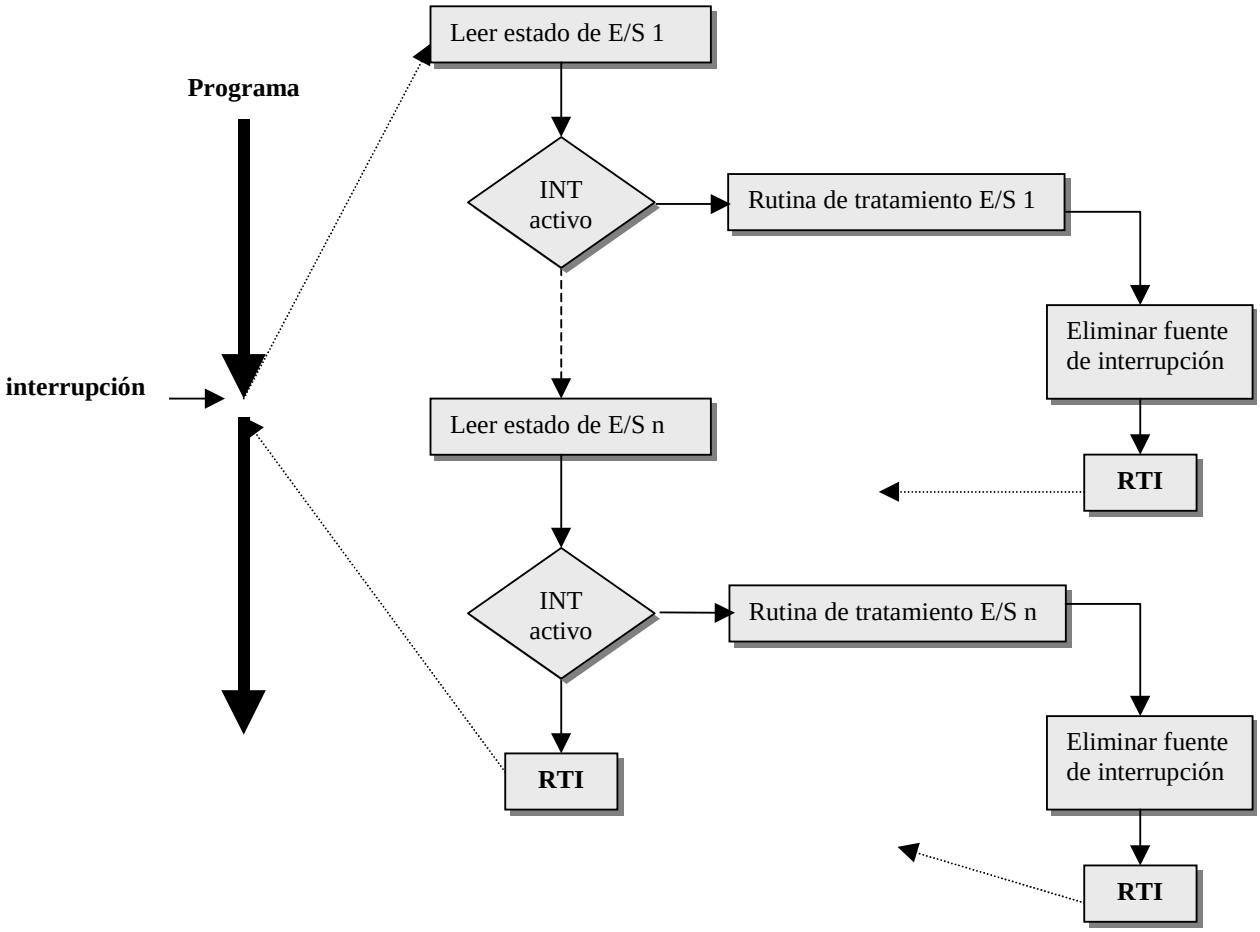
Disponen de una línea única de petición interrupción a la que se pueden conectar más de un dispositivo periférico, efectuándose en el punto de conexión la función lógica *OR*. El vector de interrupción para esta línea ocupa una posición fija de la tabla de vectores de interrupción. Cuando uno o más de los dispositivo periféricos conectados a la línea genera una señal de interrupción, la *CPU* debe identificar por software el dispositivo causante de la interrupción, y en caso de ser varios, establecer el orden de prioridad entre ellos. La identificación se realiza consultando los registros de estado locales de cada módulo de *E/S*.



La *OR-cableada* en la línea de interrupción *INT* se realiza normalmente con tecnología de transistores con colector abierto. Por ello normalmente la señal activa de interrupción es en baja.

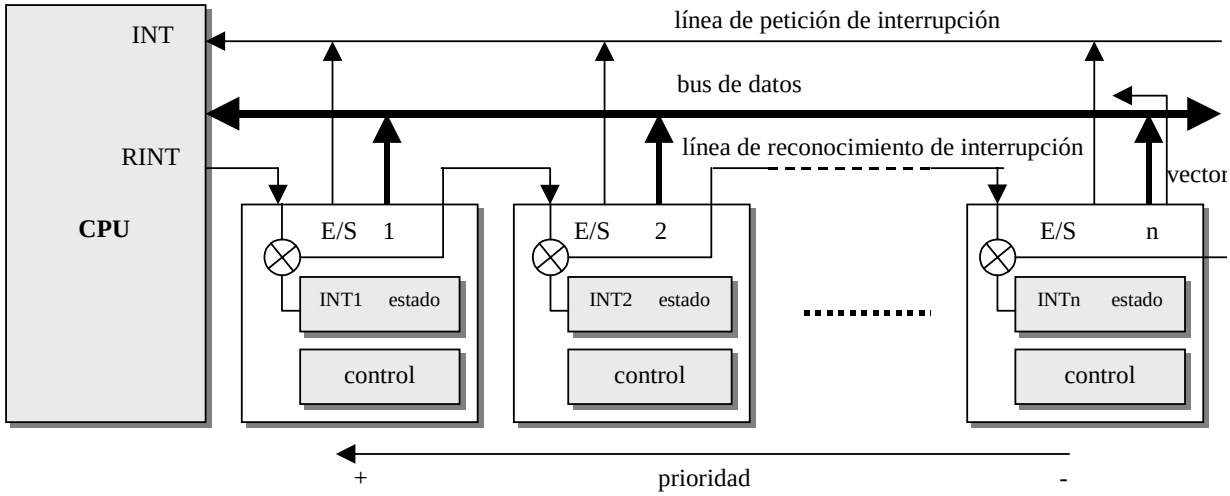


La rutina general de tratamiento asociada a la línea de interrupción comienza su ejecución identificando por encuesta el periférico causante de la interrupción, para bifurcar después a la rutina de tratamiento específica de dicho periférico, en donde se realiza la operación de E/S propiamente dicha:



2.2 Interrupciones vectorizadas:

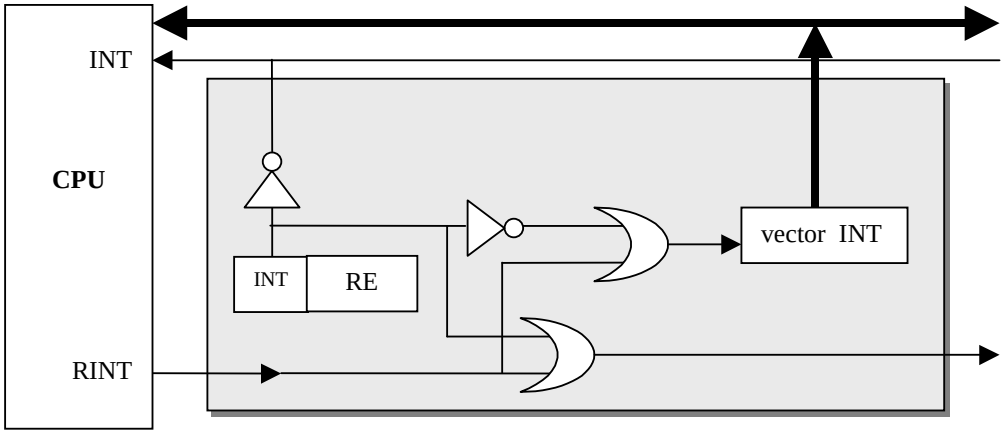
- Disponen de dos líneas de interrupción, una de petición y otra de reconocimiento.
- El vector de interrupción es generado por el dispositivo que produce la interrupción.
- Utiliza un mecanismo de *daisy chaining* en la transmisión de la señal de reconocimiento.



El proceso de interrupción se desarrolla en los siguientes pasos:

1. El dispositivo de *E/S* genera la petición de interrupción activando *INT*.
2. La *CPU* la reconoce activando *RINT*.
3. Los dispositivos de *E/S* que reciben *RINT* y no han interrumpido, la transmiten al siguiente elemento de la cadena.
4. El dispositivo de *E/S* que recibe *RINT* y ha realizado la petición coloca el vector de interrupción *n* en el bus de datos.
5. A partir del vector *n* la *CPU* bifurca a la rutina de tratamiento correspondiente al dispositivo.

La lógica interna del paso de la señal de reconocimiento por encadenamiento (*daisy chain*) para cada módulo se representa en la siguiente figura:



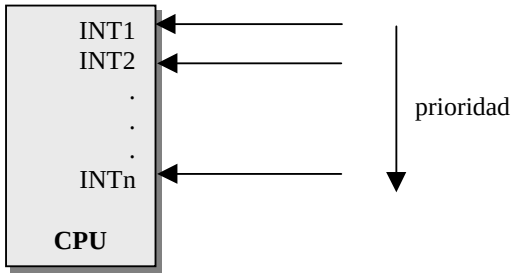
2.3 Prioridades

Cuando dos o más dispositivos de *E/S* generan una petición de interrupción, el sistema de prioridades determina la interrupción que se atiende en primer lugar.

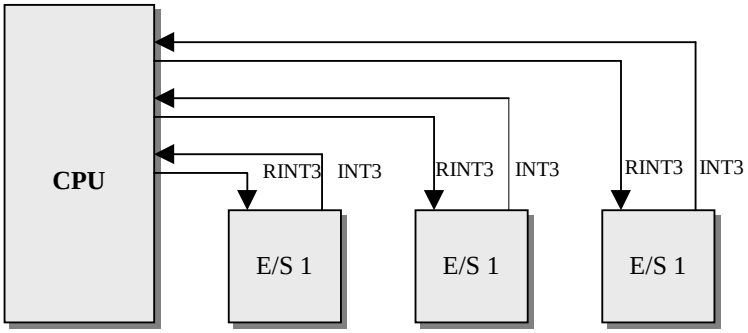
Interrupciones no vectorizadas: la prioridad la establece el software de encuesta con el orden de recorrido de todos los dispositivos de E/S salida conectados a una línea. Es decir, se establece por software.

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.

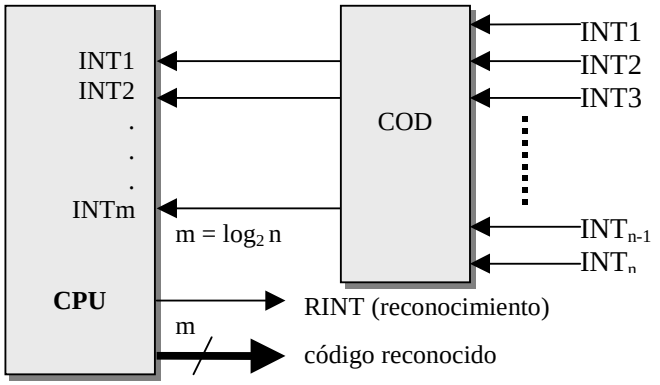
Cuando existen más de una línea de interrupción, la prioridad viene establecida por hardware en el diseño del procesador:



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



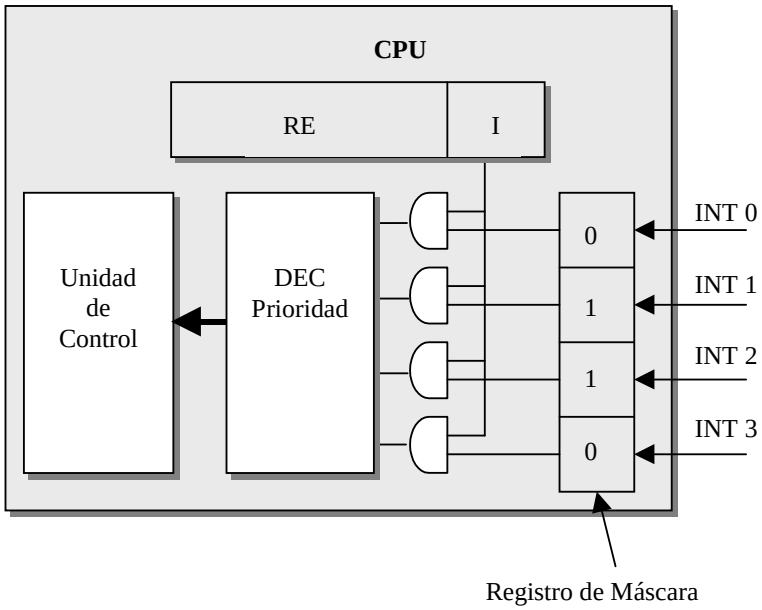
Aunque suele ser frecuente, para minimizar el número de pines del chip, que las diferentes líneas entren codificadas a la *CPU*, siendo necesario el uso de un codificador externo:



En estos casos suele existir una única señal de reconocimiento *RINT*, generándose por las líneas menos significativas del bus de direcciones el código de la interrupción reconocida, que será la más prioritaria entre las que realizaron la petición.

3. Enmascaramiento de interrupciones

El sistema de interrupciones de un procesador dispone en general de la posibilidad de ser inhibido, es decir, impedir que las interrupciones sean atendidas por la *CPU*. Esta posibilidad hay que utilizarla en determinadas ocasiones en las que por ningún concepto se puede interrumpir el programa en ejecución. Además de la inhibición total del sistema, existe la posibilidad de enmascarar individualmente algunas de las líneas de interrupción utilizando un registro de máscara:

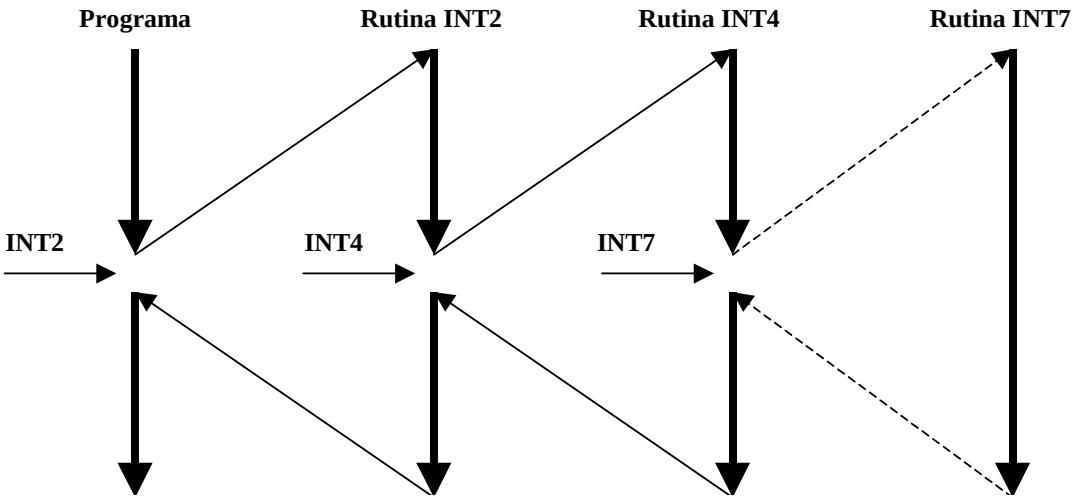


La descapacitación de algunas interrupciones también se puede realizar a nivel local del propio módulo de E/S, actuando sobre su registro de control.

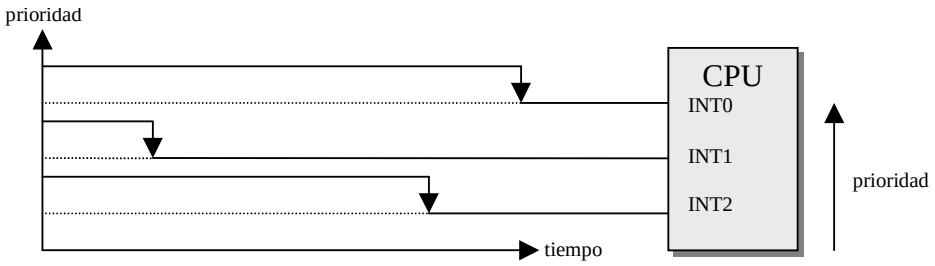
A nivel software también se pueden inhibir las interrupciones producidas sobre una misma línea por diferentes periféricos, en la fase de identificación de la rutina de tratamiento correspondiente.

4. Anidamiento de interrupciones

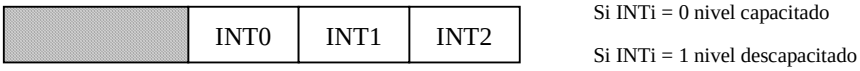
Un tema importante en un sistema de interrupciones es el de la capacidad de anidar interrupciones, es decir, la posibilidad de interrumpir la rutina de tratamiento de una interrupción por la generación de otra interrupción. Tal posibilidad viene determinada por el sistema de prioridades: *"un dispositivo sólo puede interrumpir a otro cuando su nivel de prioridad es mayor que el que se está atendiendo"*.



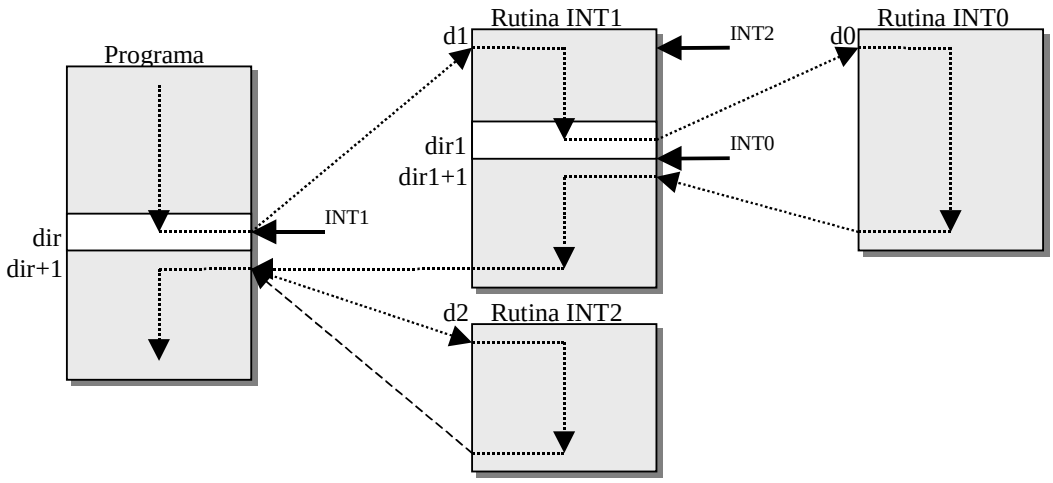
Para describir con más detalle la forma en que opera la combinación de las prioridades y el anidamiento en un sistema de interrupciones, consideremos una *CPU* con tres líneas de interrupción: *INT0, INT1, INT2*, siendo la primera la más prioritari, y la última la menos prioritaria. Supongamos que llegan tres peticiones de interrupción en el orden temporal del siguiente esquema:



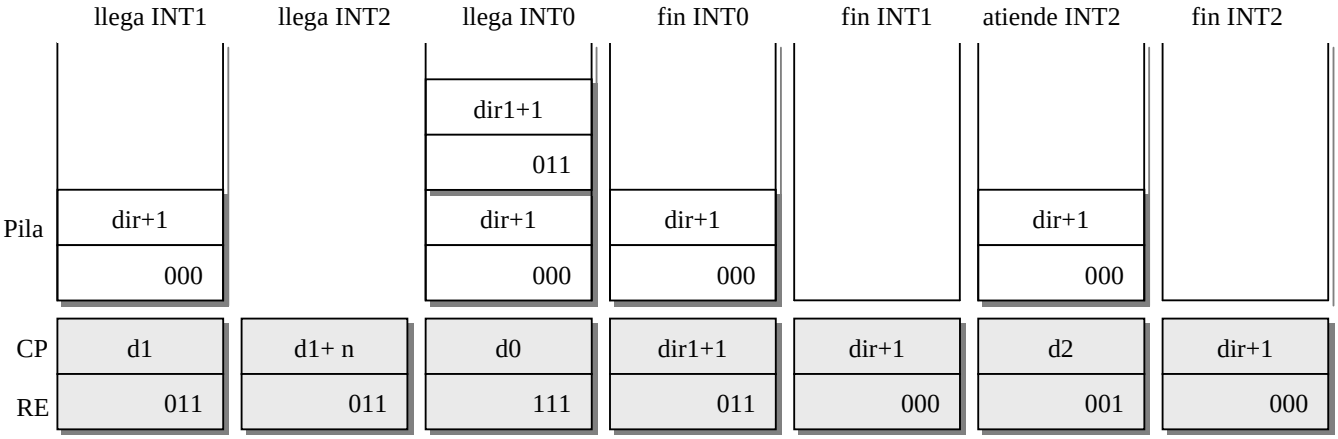
La *CPU* dispone de tres bits en su registro de estado para permitir la inhibición de forma independiente cada línea:



Las interrupciones con menor prioridad no podrán interrumpir la rutina de tratamiento de una de mayor prioridad, por lo que la secuencia de peticiones y servicios será la que aparece en el siguiente esquema:



La evolución consiguiente de la pila y el estado de la *CPU* (*CP* y *RE*) será el siguiente:



6. Ejemplos de sistemas de interrupciones

Estudiaremos en este apartado de forma general los sistemas de interrupciones de dos micrprocesadores, uno de *8* bits, el *MC 6809*, y otro de *16*, el *MC 68000*. Con el primero estudiaremos la forma de multiplicar una línea de interrupción autovectorizada en varias, también autovectorizadas, utilizando un dispositivo hardware. Con el segundo veremos un sistema de interrupciones más complejo en el que coexisten las interrupciones autovectorizadas y vectorizadas.

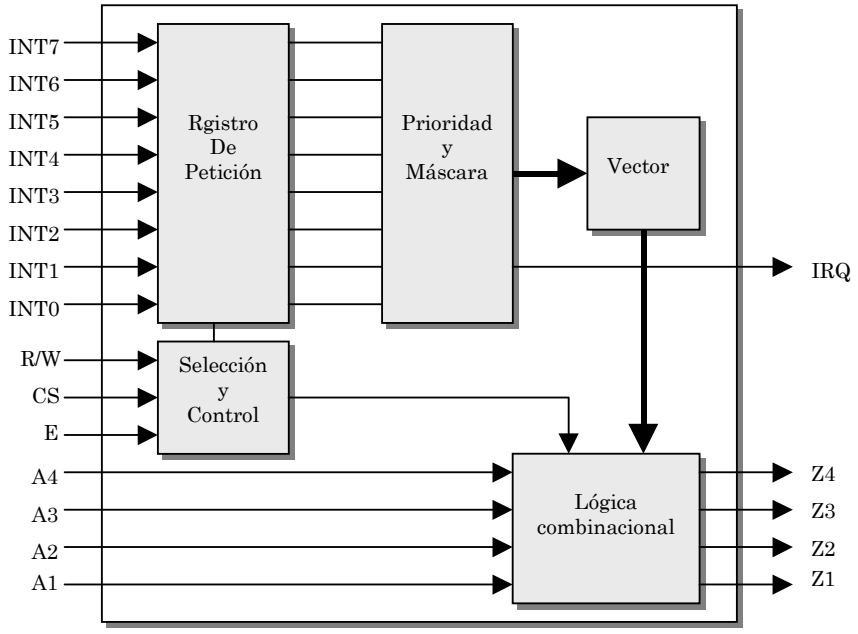
6.1 Sistema de interrupciones del microprocesador MC 6809

El *MC 6809* dispone de las siguientes líneas de interrupción:

Línea	Prioridad	Dirección Rutina	Bit de máscara en RE	Comportamiento
IRQ	Baja	FFF8 – FFF9	I	Salva todos los registros
FIRQ	|	FFF6 – FFF7	F	Salva CP y RE
NMI	↓	FFFC – FFFD	No enmascarable	Salva todos los registro
RESET	Alta	FFFE - FFFF	No enmascarable	No salva registros

Veremos como se puede transformar la línea *IRQ* en 8 líneas de interrupción diferentes. Para ello se aprovecha el que cuando se produce una interrupción por esta línea y es atendida por la *CPU*, ésta accede a su vector de interrupción que se encuentra en las direcciones *FFF8* y *FFF9*, lo que significa que en el bus de direcciones aparecerán estas direcciones. Sólo en respuesta a una interrupción por IRQ se generan estas direcciones en el bus, por lo que podemos identificarla por hardware y descomponerla en 8 pares diferentes, una por cada línea nueva de interrupción.

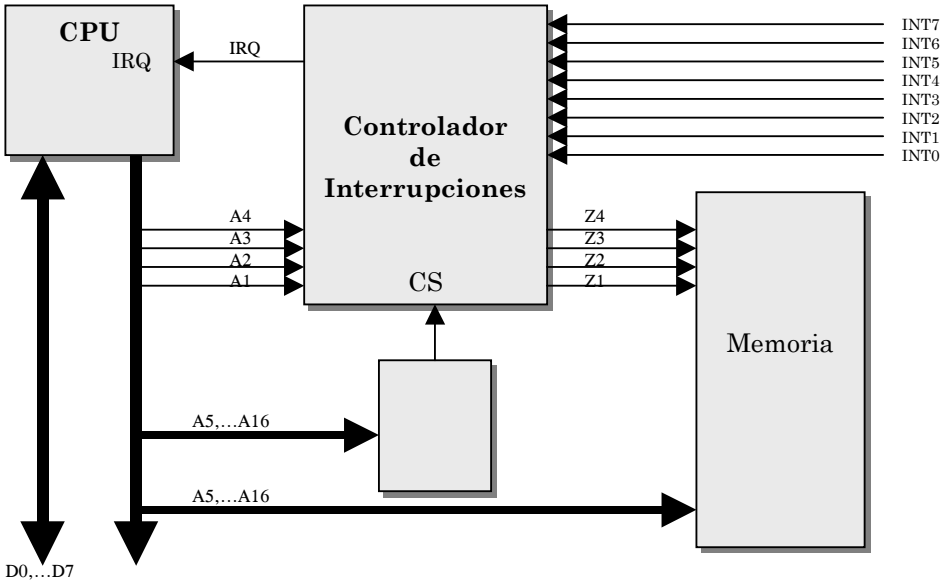
El esquema completo de un dispositivo que realiza esta conversión lo hemos representado en la siguiente figura. El dispositivo se selecciona cuando se activa la señal *CS*, por lo que esta señal deberá generarse cuando en el bus aparezcan las direcciones *FFF8* y *FFF9*. Cuando eso ocurra, el dispositivo transformará las 4 líneas menos significativas del bus de direcciones (*A4,A3,A2,A1*) en otras 4 (*Z4,Z3,Z2.Z1*) de manera que en las *Zi* se genere uno de los 8 pares de las nuevas direcciones, el correspondiente a la línea *INTi* activada de mayor prioridad.



La tabla de transformación de direcciones sería, pués, la siguiente:

Dirección entrada	Línea	Dirección salida
A16....A4 A3 A2 A1	1	A16....Z4 Z3 Z2 Z1
FFF8 – FFF9	INT7	FFE6 – FFE7
	INT6	FFE8 – FFE9
	INT5	FFEA – FFEB
	INT4	FFEC – FFED
	INT3	FFEE – FFEF
	INT2	FFF0 – FFF1
	INT1	FFF2 – FFF3
	INT0	FFF4 – FFF5

La disposición del controlador de interrupciones respecto a la *CPU* y la memoria se muestra en el siguiente esquema:

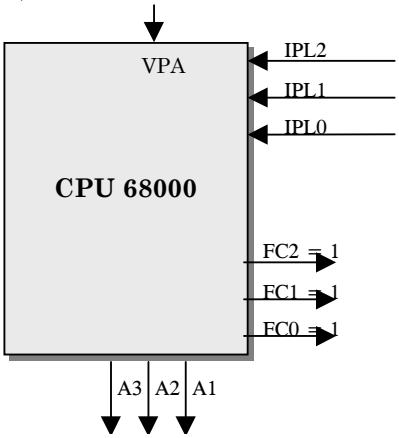


6.2 Sistema de interrupciones del microprocesador MC 68000

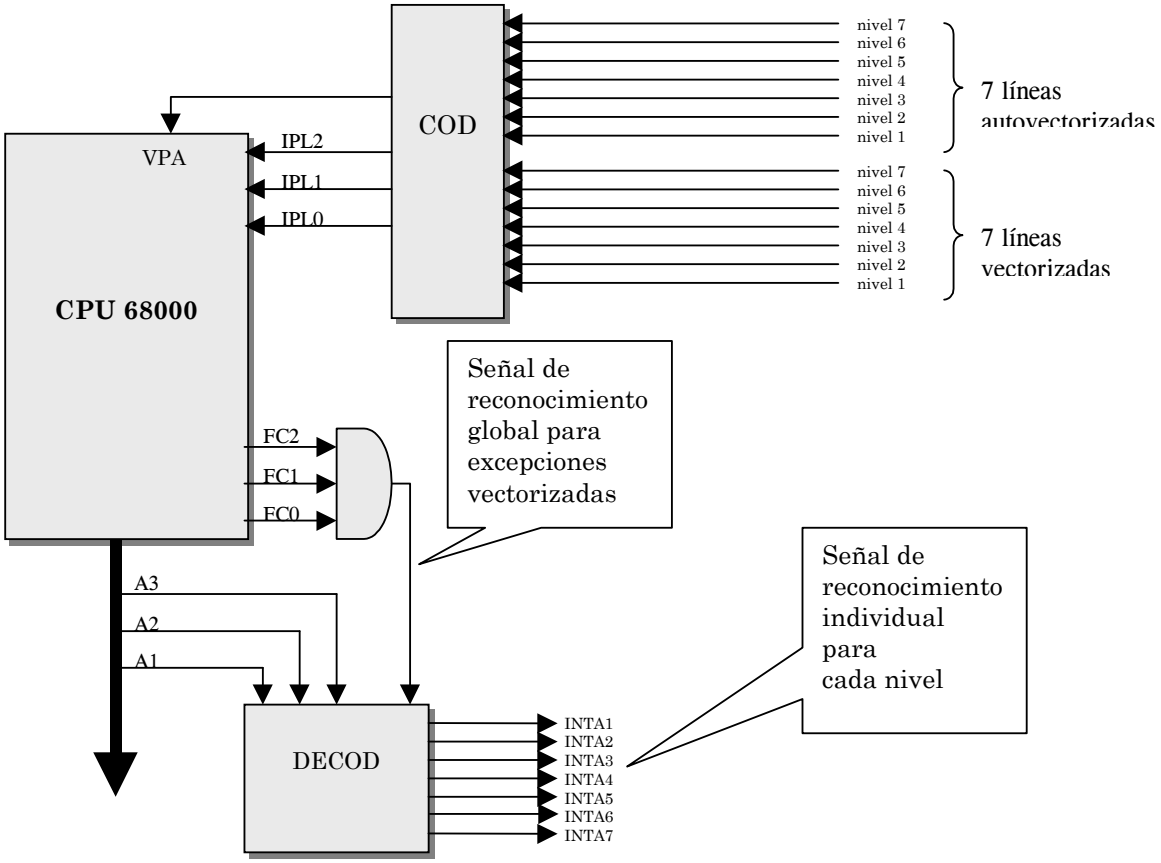
En el *MC 68000* las interrupciones se generalizan al concepto de *excepción*, que incluye las interrupciones hardware de *E/S* propiamente dichas, las producidas por eventos internos en la *CPU*, y las producidas por software.

- **Líneas de interrupción**

Las excepciones de E/S llegan a la *CPU* en forma codificada a través de tres líneas externas: *IPL2*, *IPL1*, *IPL0*, que codifican la ausencia de excepción (*111*), y *7* niveles con prioridad (*000* el más prioritario y *110* el menos prioritario).



Una cuarta línea, *VPA*, determina si los siete niveles de las líneas *IPLi* codifican una excepción autovectorizada (*VPA = 0*) o vectorizada (*VPA = 1*). En el primer caso el vector de excepción es fijo para cada nivel, y en el segundo lo determina el periférico y lo transmite por las *8* líneas menos significativas del bus de datos cuando le llega la señal de reconocimiento procedente de la *CPU*, que corresponde al valor *111* de las señales de estado *FC2, FC1, FC0*. El número de la línea vectorizada reconocida lo genera la *CPU* a través de las líneas del bus de direcciones *A3, A2, A1*. Por ello, es necesario complementar con circuitería lógica externa el sistema, tal como aparece en el siguiente esquema:



Enmascaramiento

Los bits I2,I1,I0 del registro de estado codifican los niveles inhibidos por la CPU, de acuerdo a la siguiente tabla:

I2 I1 I0	Niveles inhibidos
0 0 0	Ninguno
0 0 1	Nivel 1
0 1 0	Niveles 1 al 2
0 1 1	Niveles 1 al 3
1 0 0	Niveles 1 al 4
1 0 1	Niveles 1 al 5
1 1 0	Niveles 1 al 6
1 1 1	Niveles 1 al 6
El nivel 7 no es enmascarable	

T		S		I_2	I_1	I_0		**X**	**N**	**Z**	V	**C**

Vectores de excepción

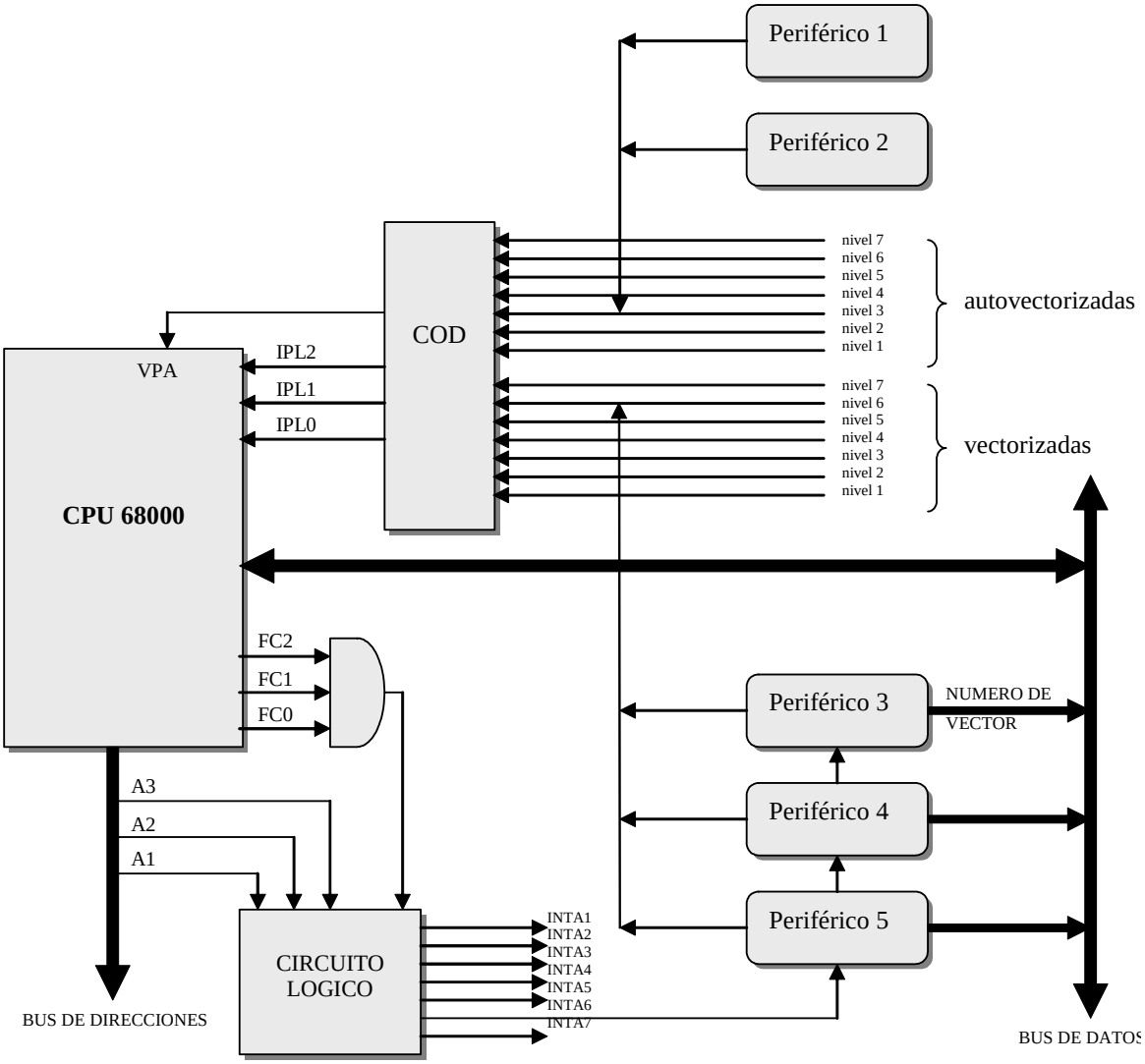
Los vectores de excepción son posiciones de memoria donde la CPU obtiene la dirección de la rutina de tratamiento de la excepción. En el MC 68000 existen 255 vectores que se agrupan en la tabla de vectores de excepción ubicada en la zona más baja de memoria (desde la dirección 0 a la 1020 en decimal). Cada vector ocupa 2 palabras (4 bytes) excepto el vector de RESET que ocupa 4 palabras, 2 para el CP y las otras 2 para el SP, y se identifica con un número de 8 bits (en decimal del 0 al 255, excepto el 1, que corresponde a la segunda mitad del 0, que es el de RESET). La dirección real se obtendrá multiplicando por 4 el número de vector. El número de vector lo genera la CPU (excepciones autovectorizadas) o el periférico (excepciones vectorizadas).

Tabla de vectores de excepción	
Nº de vector	Asignación
0 1 1 3 4 5 6 . . .	RESET (CP inicial) RESET (SP inicial) error del bus dirección no existente instrucción ilegal división por cero instrucción CHK (entero fuera de rango)
25 26 27 28 29 30 31	autovector nivel 1 autovector nivel 2 autovector nivel 3 autovector nivel 4 autovector nivel 5 autovector nivel 6 autovector nivel 7
. . .	
64 . . . 255	192 vectores de excepción de usuario

Ejemplo de conexión de periféricos

Para finalizar veamos el esquema completo de conexión de periféricos al sistema de excepción del MC 68000.

Se han conectado tres periférico de forma encadenada (*daisy chain*) a la línea vectorizada de nivel 3.


Periférico 1
Periférico 2
COD
nivel 7
nivel 6
nivel 5
nivel 4
nivel 3
nivel 2
nivel 1
autovectorizadas
nivel 7
nivel 6
nivel 5
nivel 4
nivel 3
nivel 2
nivel 1
vectorizadas
VPA
IPL2
IPL1
IPL0
CPU 68000
FC2
FC1
FC0
Periférico 3
NUMERO DE
VECTOR
Periférico 4
Periférico 5
A3
A2
A1
CIRCUITO
LOGICO
INTA1
INTA2
INTA3
INTA4
INTA5
INTA6
INTA7
BUS DE DIRECCIONES
BUS DE DATOS