Índice

Interrupciones: Mecanismo de funcionamiento

Área de Arquitectura y Tecnología de Computadores – Versión 1.0.428, 27/03/2020

Objetivos

En esta sesión se pretende que el alumno conozca y comprenda claramente el mecanismo de generación y aceptación de interrupciones.

En una primera fase se estudiarán los cambios que una interrupción produce en el estado del procesador para, seguidamente, observar paso a paso la secuencia de señales que motivan estos cambios. Finalmente, se manipulará la memoria y la tabla de vectores de interrupción para lograr la ejecución y retorno con éxito de una rutina de servicio mínima.

Conocimientos y materiales necesarios

Para poder realizar esta sesión el alumno debe:

- Saber escribir programas en el lenguaje ensamblador de la CPU elemental y utilizar el programa ensamblador para obtener archivos `.eje`.
- Repasar en los apuntes de teoría el concepto de interrupción y el mecanismo que la unidad de control pone en marcha cuando detecta una interrupción.

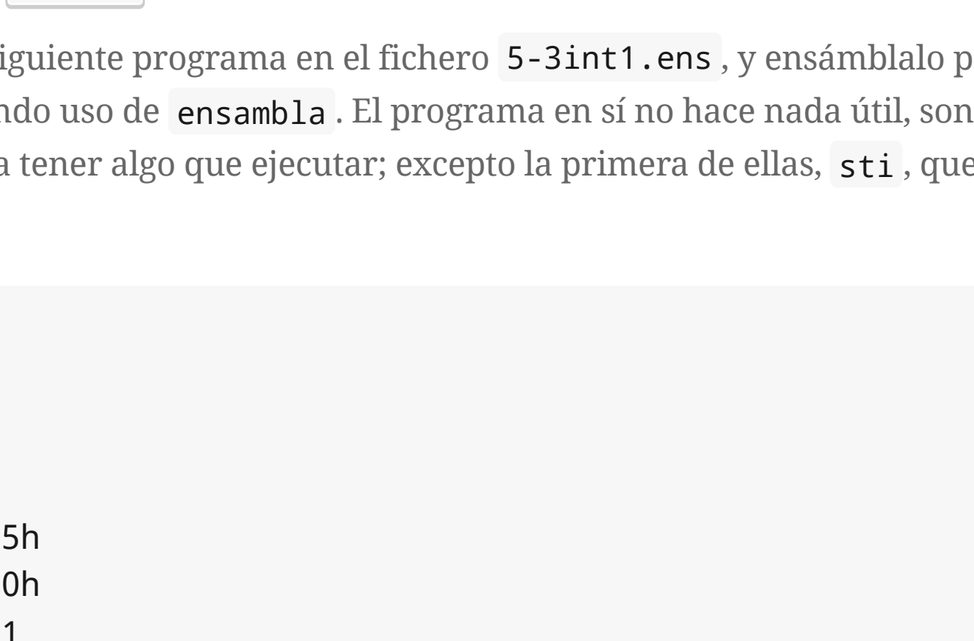
Durante la sesión se plantearán una serie de preguntas que deben responderse en el correspondiente cuestionario del Campus Virtual. El cuestionario se puede abrir en otra pestaña del navegador pinchando en el enlace mientras se mantiene pulsada la tecla `Ctrl`.

1. Primeros experimentos

1.1. Configuración inicial del computador

Para poder estudiar qué ocurre cuando la CPU recibe una interrupción, necesitaremos conectar a ésta un dispositivo capaz de generar interrupciones. Para esta práctica usaremos el dispositivo llamado `Luces`. Realiza los siguientes pasos:

- Inicia el simulador de la CPU. En el menú de la Memoria, elige la opción `Configurar`.
- Elimina el módulo de 32K que cubre las direcciones altas de la memoria (de `8000h` a `FFFFh`), y pon uno nuevo de 16K que ocupe las direcciones bajas de este hueco que has creado. Por tanto, los últimos 16K del mapa de direcciones quedarán vacíos (sin memoria), y este rango estará disponible para interfaces de E/S. Cierra la ventana de configuración de la memoria.
- Pulsando sobre la `Entrada/Salida` selecciona la opción `Conectar Luces`.
- Elige como nombre para el dispositivo `Luces`, como dirección base `E000h`, como vector el `3` y como prioridad `1`. Marca la casilla `Generar Int.` para indicar que este periférico tiene capacidad de generar interrupciones.
- Cuando hayas finalizado la configuración se mostrará un dispositivo como el de la siguiente figura:



Verifica que en los `Datos del periférico` aparecen correctamente la dirección base, el vector, la prioridad y sobre todo que el círculo `INT` está marcado. Si no fuera así, debes eliminar este periférico pulsando con el ratón en `Entrada/Salida` y eligiendo `Desconectar periférico` para crearlo de nuevo con los datos correctos.

- Presiona el botón `Minimizar` en esa ventana.
- Ahora escribe el siguiente programa en el fichero `5-3int1.ens`, y ensámblalo para obtener `5-3int1.eje` haciendo uso de `ensambla`. El programa en sí no hace nada útil, son sólo unas pocas instrucciones para tener algo que ejecutar; excepto la primera de ellas, `sti`, que es fundamental para esta práctica.

```
ORIGEN 300h
.PILA  20h
.CODIGO
   STI
   MOVL R1, 05h
   MOVH R1, 80h
   MOV  R0, R1
FIN
```

- Finalmente, carga `5-3int1.eje` en el simulador. Todo está listo ahora para iniciar la práctica.
- Guarda el estado de la CPU en este momento en el archivo `5-3int1.sim`, pues necesitaremos volver a este mismo estado más adelante.

1.2. Ejecución normal del programa (sin interferencias)

Observa que en el registro `PC` se ha cargado la dirección en la que comienza nuestro programa ¿Cuál es? Responde en el cuestionario: pregunta 1. Nuestro programa tiene 4 instrucciones, y en la dirección siguiente a la que se almacena la última instrucción, comienza la pila. Ya que hemos indicado un tamaño de `20h` mediante el uso de `.PILA`, ¿en qué dirección debería terminar la zona de pila? Responde en el cuestionario: pregunta 2. Observa que el registro `r7` se inicializa con el valor correspondiente a la siguiente dirección donde finaliza la pila.

Si ahora pulsáramos varias veces `F8`, las diferentes instrucciones se irían ejecutando en secuencia. Ya que ninguna de ellas es una instrucción de salto, el valor del registro `PC` simplemente se irá incrementando. Por otra parte, como tampoco hay ninguna instrucción `push`, `pop`, `call` ni `ret`, la pila no se toca nunca y por tanto el registro `r7` no debería cambiar de valor.

- Pulsa `F8` cuatro veces para ejecutar el programa y comprueba que efectivamente es así.

Ahora vamos a repetir la ejecución del programa, pero causando desde el periférico una interrupción, para ver qué ocurre.

1.3. Ejecución interrumpida por el periférico

- Carga de nuevo el estado `5-3int1.sim` para volver a la situación inicial. Comprueba los valores en los registros `PC`, `SR` y `r7`.
- Pulsa `F8`. Se ejecuta la instrucción `sti`. ¿Ha cambiado algún registro en el procesador, además de `PC`? ¿Cuál? Responde en el cuestionario: pregunta 3.
- Comprueba que el bit `IF` del registro de estado ahora está a 1. Esto significa que el procesador admite ser interrumpido.
- Pulsa `F8` para ejecutar la siguiente instrucción, `movl r1, 05h`.

Ahora vamos a solicitar una interrupción desde el periférico `Luces`:

- Haz reaparecer la ventana del periférico `Luces`.
- Pulsa el botón `Generar Interrupción` `INT`.
- Vuelve a minimizar esta ventana y observa el dibujo de la CPU. ¿Ves una línea llamada `INT` que ahora aparece en color rojo? Esto indica que un dispositivo de E/S ha solicitado una interrupción.

Por el momento la CPU aún no se ha dado cuenta de que la línea `INT` está activada, ya que sólo examina esa línea cuando la unidad de control genera la señal `FIN`, pero ya había generado esa señal antes de que nosotros activáramos `INT`, por tanto `no se dará cuenta` hasta la próxima activación de `FIN`.

Si ahora pulsáramos `F8` (no lo hagas), se ejecutaría la próxima instrucción `movh r1, 80h` y cuando esta ejecución llegara a su `FIN`, entonces sería cuando la unidad de control comprobaría el estado de `INT` y la encontraría activada. Antes de pulsar `F8`, responde a las siguientes preguntas:

- ¿Qué valor esperarías en el registro `PC` si la ejecución fuese `normal`, es decir, sin interrupciones? Responde en el cuestionario: pregunta 4.
- Observa el valor que hay en el registro `r7`.

Ahora pulsa `F8`. En un breve instante ocurren muchas cosas, puesto que se ejecuta la instrucción `movh` y a continuación se detecta `INT`, lo que causa unos importantes cambios en varios registros del procesador. Tratemos de localizar esos cambios:

- ¿Qué valor aparece en el registro `PC`? Responde en el cuestionario: pregunta 5. Después veremos de dónde ha salido este valor.
- Fíjate en el registro de estado, ¿ha cambiado de valor? Responde en el cuestionario: pregunta 6.
- ¿Qué valor hay ahora en `r7`? Responde en el cuestionario: pregunta 7. ¿En cuánto se diferencia del valor que tenía anteriormente? Responde en el cuestionario: pregunta 8.
- La respuesta anterior implica que algo ha sido introducido en la pila. Si utilizas el editor hexadecimal para mirar lo que hay en la dirección apuntada por el registro `r7` encontrarás qué es lo que se ha guardado en la pila. Fíjate que coincide con la respuesta del bloque anterior, la dirección que esperábamos que tuviera `PC` ante una ejecución "normal".
- Y ¿cuál es el número que hay en la pila justo a continuación? Responde en el cuestionario: pregunta 9. ¡Este es el antiguo valor del registro de estado!
- Observa que la línea `INT` ya está de nuevo de color negro. Esto implica que nuestro periférico `Luces` la ha desactivado.
- Observando el nuevo valor del registro `PC` y utilizando el desensamblador de la memoria puedes saber cuál será la próxima instrucción que va a ejecutar la CPU. ¿Cuál sería? Responde en el cuestionario: pregunta 10.
- Teniendo en cuenta que `INT` ya no está activa (y además `IF` es cero), ¿cuál sería el siguiente valor de `PC` si pulsamos `F8`? Responde en el cuestionario: pregunta 11.
- Pulsa `F8` y comprueba tus dos respuestas anteriores.

Observa que a partir de este instante, todo irá mal en nuestra CPU. El control ha saltado a la dirección de memoria `0000h`, donde no hay nada para ejecutar salvo instrucciones `nop`. Nuestro programa ha sido abandonado a la mitad y nunca se regresará a él. La información que se guardó en la pila nunca será extraída.

Todas estas "calamidades" se deben a que se produjo una interrupción, pero el sistema no tenía correctamente preparada la tabla de vectores de interrupción ni la rutina de servicio.

2. Instalando una rutina de servicio

Recibe el nombre de rutina de servicio o también, rutina de interrupción, el código o programa que queremos que se ejecute cuando se produzca la interrupción. Este código se organiza como un procedimiento con una característica especial. A los pasos a seguir para conseguir que el proceso funcione se le denomina `instalación de la rutina de servicio o interrupción`.

`Instalar` una rutina de servicio consiste en suministrar al procesador la dirección de memoria donde está almacenada la primera instrucción del código de la rutina de servicio. Esto requiere dos pasos:

- Obtener la dirección. Ahora la obtendremos de forma manual, pero en la siguiente sesión se explicará como obtenerla utilizando directivas del ensamblador.
- Suministrar esa dirección al procesador. Para ello, se establece un `convenio` que consiste en guardar la dirección de memoria (dónde comienza el código de la rutina), en una posición de memoria concreta, fijada dependiendo de un número de identificación del periférico. Este número de identificación del periférico recibe el nombre de `número de vector de interrupción`.

Así, un periférico cuyo identificador sea el número 5 indicará que su número de vector es el 5, y la dirección de la primera instrucción de su rutina de servicio estará guardada en la dirección de memoria `0005h`. En nuestro caso es posible utilizar los identificadores `0` a `255` (`FFh`), lo que implica que las direcciones de memoria `0000h` a `00FFh` se podrían utilizar para guardar las direcciones de comienzo de las rutinas. Al conjunto de estas direcciones se le denomina `tabla de vectores de interrupción`.

Los pasos anteriores habría que repetirlos para cada posible periférico, puesto que cada uno de ellos generará un número de vector diferente y probablemente requerirá también un programa diferente que se ocupe de él.

3. La rutina de servicio más simple posible

La rutina de servicio más simple posible sería una rutina vacía, es decir, que no haga absolutamente nada útil, salvo retornar para que, una vez finalizada su ejecución, vuelva a ejecutarse el programa que fue interrumpido. Como hemos indicado, el código de la rutina de servicio se organiza como un procedimiento con una característica especial; esta característica especial es que finaliza con la instrucción `iret` en lugar de `ret`. Por tanto, la rutina de servicio más simple posible constará de una sola instrucción: `iret`. Usando las tablas de codificación, vemos que el código máquina de `iret` es `B800h`. Vamos ahora a poner ese código en nuestro computador.

- Carga de nuevo en el simulador el estado `5-3int1.sim`. Comprueba que los registros `PC` y `r7` están correctamente inicializados.
- Vamos a colocar nuestra `mini-rutina de servicio` en la dirección de memoria `50C0h` (un valor cualquiera elegido al azar, con la precaución de que no sea una zona utilizada por nuestro programa de prueba ni su pila).
- Abre el editor hexadecimal de la memoria y ve a la posición `50C0h`.
- Introduce allí el código máquina de `iret`. Cierra el editor hexadecimal y comprueba con el desensamblador que la codificación ha sido correcta.

La rutina ya está en memoria. Ahora tenemos que relacionarla con el periférico `Luces`, de modo que cada vez que `Luces` genere una interrupción se ejecute nuestra rutina. Para ello, hay que instalar la rutina, es decir, modificar la tabla de vectores de interrupción. La tabla de vectores de la CPU elemental comienza en la dirección de memoria `0000h`. Cada vector ocupa una posición, por tanto para modificar un vector de la tabla basta modificar la dirección de memoria igual al número del vector.

- ¿Cuál es el número de vector que hemos asignado al periférico `Luces`? Responde en el cuestionario: pregunta 12.
- Esa será la dirección de memoria que tenemos que modificar ahora. Abre el editor hexadecimal y ve a esa posición de memoria.
- Modifica el contenido de esa dirección y escribe `50C0h`. Este número es la dirección donde hemos puesto nuestra rutina de servicio. Ese será el valor que tome el registro `PC` al aceptar una interrupción generada por el periférico que tenga asociado ese número de vector de interrupción.
- Cierra el editor hexadecimal de la memoria.
- Guarda el estado del simulador en el archivo `5-3int2.sim`, pues lo necesitaremos más adelante.

¡Nuestra rutina ya está `instalada`! A partir de este momento, cuando `Luces` genere una interrupción, el registro `PC` tomará el valor `50C0h` y ejecutará la instrucción `iret` que hemos colocado allí. ¡Y por tanto la ejecución volverá a nuestro programa que podrá seguir funcionando!

Comprobemos todo esto:

- Pulsa `F8`. Se ejecuta la instrucción `sti` y el bit `IF` del registro de estado se pone a 1, con lo que el procesador está listo para aceptar interrupciones.
- Pulsa `F8`. Se ejecuta la instrucción `movl r1,05h`.
- Abre la ventana del periférico `Luces` y pulsa sobre `Generar Interrupción` `INT`. Observa cómo la línea `INT` se pone de color rojo en la CPU.
- Si ahora pulsáramos `F8`, tendría lugar la ejecución de la instrucción actual (`movh`), e inmediatamente, al estar la línea `INT` activada, tendrían lugar los cambios que ya conocemos (apilación de los registros `PC` y `SR`, y modificación de los registros `r7` y `PC`). Pulsa `F8` y comprueba si has comprendido los cambios que ocurren.
- En la situación actual, ¿qué instrucción será la próxima en ejecutarse? Responde en el cuestionario: pregunta 13.
- La instrucción `iret` simplemente saca dos datos de la pila (por lo que el registro `r7` se incrementa en 2 unidades), y guarda el primero de ellos en el registro `PC` y el segundo en el registro de estado. Por tanto, al ejecutar esta instrucción:
  - ¿Qué valor aparecerá en el registro `r7`? Responde en el cuestionario: pregunta 14.
  - ¿Qué valor aparecerá en el registro `PC`? Responde en el cuestionario: pregunta 15.
  - ¿Qué valor tomará el bit `IF` del registro de estado? Responde en el cuestionario: pregunta 16.
- Pulsa `F8` y verifica tus respuestas anteriores. En la situación actual ¿cuál sería la próxima instrucción en ejecutarse? Responde en el cuestionario: pregunta 17.

Observa que en esta ocasión la interrupción ha causado la ejecución de una rutina `ajena` a nuestro programa, tras lo cual la ejecución prosigue en el punto en que nuestro programa fue interrumpido. Para ello, le hemos indicado al procesador la dirección de comienzo de la rutina, `50C0h`, y la hemos dejado en el lugar convenido para el periférico, número de vector 3, dirección de memoria `0003h`. Al finalizar, la instrucción `iret` permite que todo vuelva al estado anterior a la interrupción. En esta situación podríamos generar una nueva interrupción, que sería atendida tan pronto como la CPU terminara de ejecutar la instrucción en curso.

4. Ejecución paso a paso de la aceptación de interrupción

Ya sabemos qué ocurre cuando la CPU acepta una interrupción. Veremos ahora qué señales va activando la unidad de control para lograr llevar a cabo todas las acciones necesarias (apilar los registros `SR` y `PC`, consultar la tabla de vectores de interrupción, asignar al registro `PC` el nuevo valor obtenido de la tabla y desactivar el bit `IF` del registro de estado).

- Carga de nuevo el fichero `5-3int2.sim` en el simulador. Tenemos de nuevo nuestro programa a punto de empezar y la rutina `instalada` en la dirección `50C0h`.
- Pulsa `F8` para ejecutar la instrucción `sti`. La próxima instrucción sería `movl r1,05h`, pero ésta vamos a ejecutarla paso a paso (`F7`) para ver qué ocurre exactamente en el instante en que la interrupción se acepta.
- Pulsa `F7` dos veces para ejecutar los dos primeros pasos de obtención del código máquina de la instrucción a ejecutar e incremento del registro `PC`.
- Abre la ventana del periférico `Luces` y pulsa sobre `Generar Interrupción` `INT`. Observa cómo la línea `INT` en la CPU se pone de color rojo. Hemos activado `INT` mientras la CPU estaba `en medio` de la ejecución de una instrucción. Esta situación es muy normal ya que cuando un periférico activa `INT` no sabe qué está haciendo la CPU en ese instante.
- Pulsa `F7` para pasar al siguiente ciclo de ejecución. Observa que la unidad de control prosigue con su secuencia de pasos `normal`, ignorando la línea `INT` activada. Sólo examinará esta línea cuando llegue a la señalFIN.`Por tanto, ahora se finaliza el paso 3 que completa la lectura del código máquina de la instrucción. Observa cómo en el registro `IR` aparece la instrucción `movl r1,05h`.
- Pulsa de nuevo `F7` para ejecutar el cuarto (y último) paso de esta instrucción. Observa que la unidad de control activa la señal `FIN`. Esto implica que en este momento, también acaba de `darse cuenta` de que la línea `INT` está activa. Y, puesto que el bit `IF` también está activo, la unidad de control ya ha decidido `aceptar` la interrupción.
- Observando el simulador no encontramos indicios de que la línea `INT` haya sido reconocida, pero sí lo ha sido. Pulsa `F7` y observa la unidad de control. Fíjate en el número del paso, ya no es `1` de la siguiente instrucción del programa, sino `I-1` (la `I` indica que se halla en una secuencia de señales especial para el tratamiento de Interrupciones).
- Las señales que la unidad de control está generando tienen por objetivo decrementar el valor del registro `r7`. ¿Qué valor hay en el registro `TMPS`? Observa que es una unidad menos que el valor de `r7`. El valor que has obtenido en el registro `TMPS` es justamente el lugar de la pila donde hay que almacenar una copia del registro de estado.
- Las señales siguientes que generará la UC, paso `I-2`, irán preparando una copia del registro de estado en el registro `MDR`. ¿Qué señales serán necesarias para realizar esta operación? Pulsa `F7` y verifica tu respuesta. Responde en el cuestionario: pregunta 18.
- Ya tenemos el dato en el registro `MDR`, falta escribir la dirección en el registro `MAR`. Como hemos dicho, esta dirección está en `TMPS`. ¿Qué señales debes activar en el paso `I-3` para copiarla a `MAR`? Responde en el cuestionario: pregunta 19. Pulsa `F7` y observa las señales que genera la UC. Las señales que acabas de responder tienen que aparecer allí, pero aparecen algunas más. Fíjate que la UC aprovecha el dato que hay en el bus interno para cargarlo también en el registro `r7`, de modo que `r7` queda así decrementado. Además, se activa la señal `WRITE` para que la memoria guarde el dato en la dirección que se le indica.
- Mientras la memoria guarda ese dato, la UC se prepara para repetir la operación, es decir, decrementar el contenido del registro `r7` en una unidad para enviar a esa dirección de memoria una copia del registro `PC`. Pulsa `F7` y comprueba cómo el ciclo `I-4` es idéntico al ciclo `I-1`.
- Ahora hay que copiar el contenido del registro `PC` en el registro `MDR` para enviarlo a la memoria, como se hizo antes con `SR`. Pulsa `F7` dos veces y observa cómo lo hace la UC.
- Observa un detalle en este último ciclo (`I-6`). Además de las señales necesarias para escribir en la memoria, la UC activa otra señal llamada `INTA`. Si te fijas en la zona de `Entrada/Salida` ves que esa línea `INTA` llega hasta allí, y está de color rojo. Esta línea es recibida por todos los periféricos que tengamos conectados en nuestro computador. Con esta línea la CPU está preguntando ¿qué periférico ha activado `INT`?

El periférico concreto que haya sido deberá responder poniendo en el bus de datos su número de identificación. En nuestro caso, será `Luces` quien responda, escribiendo el valor 3 en el bus de datos. En preparar su respuesta tardará un ciclo de reloj.

- Pulsa `F7`, paso `I-7`. Observa que la UC no genera ninguna señal. Es un ciclo de espera mientras el periférico que causó la interrupción prepara su respuesta en el bus de datos.
- Pulsa de nuevo `F7`, paso `I-8`. ¿Qué aparece en el bus de datos del sistema (SDB)? Responde en el cuestionario: pregunta 20. Este es el vector que le hemos asignado al periférico `Luces` cuando lo conectamos.

Observa que la unidad de control simplemente toma ese dato (que obtiene en el registro `MDR`) y lo transfiere al registro `MAR`, activando la señal `READ`.

- ¿Qué dirección de memoria tratamos de leer? ¿Qué valor hay allí? Responde en el cuestionario: pregunta 21. (Puedes usar el editor hexadecimal de la memoria para consultarlo). Es decir, la unidad de control está usando el número de vector para obtener la dirección de la rutina a la que debe saltar.
- Pulsa `F7`, paso `I-9`, y observa cómo la UC está esperando por la respuesta de la memoria.
- En el siguiente ciclo, paso `I-10`, la memoria habrá respondido, y su respuesta es el nuevo valor que debe ser asignado al registro `PC`. ¿Qué señales crees que activará ahora la unidad de control? Pulsa `F7` y verifica tu respuesta.
- Además de las señales que habías respondido, la UC genera la señal `CLI` cuya función es borrar el bit `IF`, y la señal `FIN`, con lo que finaliza la fase de aceptación de la interrupción. El próximo paso que genere la UC será el paso 1 de la siguiente instrucción a ejecutar, que ahora corresponderá al código de la rutina de servicio.
- Pulsa `F7` y comprueba cómo, efectivamente, la ejecución vuelve a la normalidad comenzando por el paso 1.
- ¿A qué instrucción corresponderá el código máquina que se reciba en el paso 3? Pulsa `F7` dos veces y comprueba tu respuesta.
- Puedes seguir pulsando `F7` para ver cómo se lleva a cabo la instrucción `iret` paso a paso. La función de la instrucción `iret`, como hemos visto, es recuperar de la pila lo que se guardó allí durante los ciclos especiales `I-1` a `I-10`, y de este modo causar el retorno al punto en que el programa fue interrumpido.

5. Ejercicios adicionales

Estos ejercicios adicionales permiten al alumno reforzar los conocimientos adquiridos durante la sesión práctica.

- Conectaremos otro periférico a la CPU e instalaremos una rutina para servir a este nuevo periférico. Para ello:
  - Carga en el simulador el archivo `5-3int2.sim`.
  - Conecta un periférico de tipo Luces, con los siguientes parámetros: nombre `Más Luces`, dirección base `F010h`, número de vector `7`, prioridad `2`, `Generar int.` permitido (marca la última casilla).
  - Guarda el estado actual de la simulación en el archivo `5-3int3.sim`.
  - Si este nuevo periférico generara una interrupción ¿a qué dirección de memoria saltaría el procesador? Responde en el cuestionario: pregunta 22. Compruébalo pulsando `F8` para ejecutar la istrucción `sti` y seguidamente genera una interrupción en el periférico `Más luces`. Pulsa `F8` otra vez y comprueba el nuevo valor del registro `PC`.
  - Carga en el simulador el fichero `5-3int3.sim` para recuperar el estado que tenías en el punto 3.
  - Instala una rutina que contenga una sola instrucción (`iret`) en la dirección de memoria `6000h` y modifica el vector `7` para que apunte a ella.
  - Guarda el estado de nuevo en el archivo `5-3int4.sim`.
  - Repite el punto 4 y comprueba que ahora se salta a la dirección `6000h`.
  - Carga otra vez el estado `5-3int4.sim` y repite de nuevo el paso 4, pero esta vez genera la interrupción desde el periférico `Luces`, en lugar de `Más luces`. Como verás, según cuál sea el periférico que causa la interrupción se salta a una dirección u otra de la memoria.
- Carga el estado `5-3int2.sim`. Pulsa `F8`. Ahora genera una interrupción desde el periférico `Luces`. Pulsa `F8` de nuevo. Si todo va bien, el registro `PC` debe tener el valor `50C0h` y la CPU está a punto de ejecutar la instrucción `iret`. Si ahora `Luces` generase otra interrupción, ¿qué crees que ocurriría? ¿En qué momento la UC "se dará cuenta" de esta nueva interrupción? Comprueba con el simulador lo que ocurre.