ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

ORGANIZACIÓN DE COMPUTADORES

GRADO EN INGENIERÍA DE COMPUTADORES

ACTIVIDADES PRÁCTICAS

ACTIVIDAD 2.3:
TRATAMIENTO DE INTERRUPCIONES DE UN TEMPORIZADOR EN MIPS32

CURSO 2019-20

Luis Rincón Córcoles
luis.rincon@urjc.es

Prólogo

En este cuaderno de prácticas se proponen varias actividades sobre el tratamiento de un temporizador sencillo incorporado en la herramienta **Digital Lab Sim** de **MARS**. Se realizarán dos versiones del ejercicio:

- Una primera versión, en la que un programa activa directamente las interrupciones del temporizador actuando sobre su registro de control.
- Una segunda versión, en la que la escritura sobre el registro de control se realiza mediante una llamada a sistema operativo con la instrucción **syscall**.

Junto con estas actividades se adjuntan varios cuestionarios que se responderán a través del aula virtual. La nota obtenida en los mismos computa para el cálculo de la nota final de la asignatura, dentro del apartado de tests sobre actividades de laboratorio. Esta actividad es **optativa**.

2.3.1. LECTURA PREVIA

TRATAMIENTO DE INTERRUPCIONES EN MIPS

Las interrupciones son un tipo particular de excepciones. La diferencia con las excepciones estudiadas hasta el momento estriba en que las interrupciones son producidas por los controladores de dispositivos cuando éstos requieren atención por parte del procesador.

Cuando una interrupción es admitida, el procesador detiene el programa en curso y ejecuta una **rutina de tratamiento de interrupción** (RTI), que realiza las acciones necesarias para atender al dispositivo que la ha generado.

Las interrupciones se organizan por niveles. Cada interrupción está asociada a un nivel de interrupción, de modo que el procesador puede considerar que unos niveles son prioritarios frente a otros, y así puede decidir atender antes a las interrupciones más prioritarias.

En los procesadores existe la opción de inhibir el tratamiento de las interrupciones. En **MIPS**, el bit 0 del registro **Status** (registro $12 del coprocesador 0) habilita las interrupciones cuando vale 1, y las deshabilita cuando vale 0.

Además, los procesadores incorporan una máscara de interrupciones que permite habilitar o deshabilitarlas. Cada bit de la máscara se encuentra asociado a un nivel de interrupción. Cuando una interrupción está deshabilitada (“enmascarada”), el procesador no la atenderá inmediatamente, sino que lo hará en un momento posterior. Las interrupciones que no están enmascaradas pueden ser atendidas de forma inmediata. En **MIPS**, la máscara de interrupciones se encuentra en los bits 8-15 del registro **Status**. Cuando un cierto bit de la máscara está a 1, las interrupciones de ese nivel están habilitadas. Por el contrario, si el bit está a 0, las interrupciones de ese nivel están deshabilitadas.

En MIPS el registro Cause (registro 13 del coprocesador 0) también es relevante cuando sucede una interrupción. En concreto, son importantes los campos **ExcCode** (bits 2 a 6) y **Pending interrupts** (interrupciones pendientes, bits 8 a 15). Cada bit del campo de interrupciones pendientes está asociado a un nivel de interrupción, y se pone a 1 cuando se produce una interrupción de dicho nivel. Por su parte, cuando hay una interrupción, el campo **ExcCode** toma el valor 0 (todos sus bits a 0).

Cuando la interrupción es atendida, salta el manejador de excepciones que, una vez identificada la causa, ejecuta la RTI. Cuando ésta finaliza, el manejador debe devolver el control al programa en curso, que deberá continuar su ejecución normalmente.

INTERRUPCIONES DEL CONTADOR/TEMPORIZADOR EN MARS

La herramienta **Digital Lab Sim** de **MARS** contiene un contador capaz de generar interrupciones de forma periódica. Por tanto, se comporta como un temporizador, aunque no es programable, porque, cuando está activado, genera una interrupción cada 30 instrucciones ejecutadas, sin opción a modificar dicho número.

El contador tiene asociado un único *byte* en el mapa de direcciones, que se encuentra en la posición 0xFFFF0013, y corresponde con su registro de control. Si en dicha posición escribimos un 0, el contador queda desactivado y no generará interrupciones. En cambio, si en ella escribimos un 1, el contador generará una interrupción cada 30 instrucciones ejecutadas.

En el registro **Cause** del coprocesador 0, el bit del campo de interrupciones pendientes asociado a las interrupciones del contador es el bit 10.

MANEJADOR DE EXCEPCIONES CON ATENCIÓN A INTERRUPCIONES

Como se ha mencionado anteriormente, al producirse una interrupción debe ejecutarse el manejador de excepciones, y éste, una vez identificada la causa de la excepción, debe dar paso a una rutina de tratamiento de interrupciones, que se encargará de atender debidamente al dispositivo que la generó.

Al terminar el tratamiento de la interrupción, el manejador debe retornar, y el programa en curso continuará su ejecución como si no hubiera sucedido nada.

CUESTIONARIO 2.3.1

En preparación

2.3.2. GENERACIÓN DE INTERRUPCIONES DEL CONTADOR DE FORMA DIRECTA

INTRODUCCIÓN

En este ejercicio se escribirá un fragmento de código que escribirá directamente sobre el registro de control del contador/temporizador de la herramienta **Digital Lab Sim** de **MARS** para activar y desactivar las interrupciones generadas por el mismo.

Se parte del fichero **caso09.asm**, que pide al usuario que teclee un dato numérico positivo, para a continuación realizar una serie de iteraciones sobre un bucle. Las interrupciones se irán generando a medida que se ejecute el programa anterior.

Para realizar el tratamiento de la interrupción, se proporciona un manejador de excepciones nuevo, que incluye una rutina específica para el tratamiento de las excepciones por interrupción, y una rutina para tratar la interrupción del contador/temporizador. Este manejador está en el fichero **excepciones_rti.asm**, y, antes de utilizarlo en esta actividad, el alumno deberá modificar su contenido, incluyendo en el mismo el código que añadió al manejador al realizar el apartado 2.2.3 de la actividad 2.2.

Junto con este ejercicio se planteará un cuestionario con una serie de preguntas que se responderán a través del Aula Virtual.

ANÁLISIS DEL CÓDIGO FUENTE DEL PROGRAMA

El programa comienza presentando un mensaje genérico por pantalla indicando el número del caso de prueba ejecutado. A continuación, solicita que se introduzca un número entero positivo por la consola. Para ello, se crea una cadena de caracteres que contiene el mensaje escrito en la consola para solicitar el dato:

```
            .data
str_pedir:  .asciiz     "Introduzca un dato numérico positivo: "
```

La solicitud de introducción del dato se realiza mediante un bucle que comprueba que el mismo se encuentra dentro del rango especificado:

```
# Pedir el dato de entrada
pedir_dato:
            li    $v0,4       # Escribir tira de caracteres
            la    $a0,str_pedir
            syscall
            li    $v0,5       # Leer entero
            syscall
            ble   $v0,$zero,pedir_dato
```

Mediante este código, en el byte menos significativo de **$v0** queda escrito un dato que será utilizado para ver cuántas veces se ejecuta un cierto bucle que presenta un mensaje por pantalla. La idea es que las

interrupciones del contador/temporizador se produzcan durante la ejecución de este bucle, de modo que los mensajes que presenta el programa principal en la consola se entremezclen con los mensajes presentados por la rutina de tratamiento de interrupciones.

Para ello, es preciso activar las interrupciones del contador/temporizador antes del bucle. Esta es tarea del alumno, que deberá reemplazar los comentarios situados justo después de la etiqueta **activar** con código ensamblador que habilite dichas interrupciones.

Tras ejecutar el bucle, será preciso desactivar las interrupciones del contador. Para ello, el alumno reemplazará los comentarios situados después de la etiqueta **desactivar** con código ensamblador que realice dicha tarea.

A continuación, el programa presenta un mensaje final y termina.

DESCRIPCIÓN DEL MANEJADOR DE EXCEPCIONES

La estructura del manejador propuesto es similar a la del manejador de la actividad 2.2. Para este apartado se ha modificado lo siguiente:

- En la tabla de vectores de excepción, dentro de la sección de datos, se ha modificado la entrada correspondiente a las excepciones por interrupción, poniendo la etiqueta del punto de entrada de la rutina genérica de tratamiento de interrupciones.
- En la sección de datos se incluye también una constante que identifica el nivel de interrupción de las interrupciones del contador/temporizador dentro del registro de causa.
- Además, en la sección de datos se ha añadido también una cadena de caracteres que se imprimirá por pantalla cuando se produzca una interrupción del contador.
- Se incluye una rutina genérica de tratamiento de interrupciones, que primero deberá identificar la interrupción producida mediante el código anterior, y después realizar su tratamiento; en caso de que la interrupción no se pueda identificar, se presenta un mensaje genérico por pantalla. Después, se retorna al manejador general de excepciones.
- Se incluye también una rutina de tratamiento de la interrupción del contador, que presenta un mensaje específico por pantalla y retorna al manejador general de excepciones.

Tabla de vectores de excepción

En la tabla de vectores de excepción se modifica únicamente su primera entrada:

```
EXV_TABLE:  .word INT_RTE, GEN_RTE, GEN_RTE, GEN_RTE       # Excepciones 0-3
            ...
```

Código de la interrupción

Aquí se muestra la definición de la constante que identifica el código de la interrupción producida:

```
.eqv  INT_COUNTER_CODE,0x00000400
```

Esta constante sirve para aplicar una máscara al contenido del registro **Cause** y comprobar si la interrupción ha sido generada por el contador/temporizador.

Rutina de tratamiento de la excepción por interrupción

Esta rutina, que está incompleta, debería identificar la interrupción solicitada y, en caso afirmativo, saltar a la rutina específica de tratamiento de dicha interrupción. Si la interrupción no ha podido ser identificada, entonces se presentan varios mensajes informativos en la consola. En todo caso, al final se retornará al manejador de excepciones.

```
INT_RTE:
   # Chequear el registro de Causa $13 para identificar
   # la interrupción producida y así saltar a la RTI correspondiente
##### EL ALUMNO SUSTITUIRÁ ESTOS COMENTARIOS POR CÓDIGO ENSAMBLADOR
##### QUE SALTE A LA RUTINA DE TRATAMIENTO CONCRETA

# Si la RTI no está implementada, presentar mensajes
      # Salvar copia de $v0 en memoria
            sw    $v0,sv_v0
      # Salvar copia de $a0 en memoria
            sw    $a0,sv_a0
      # Salvar copia de $ra en memoria
            sw    $ra,sv_ra
      # Imprimir mensajes
            jal   MSG_PRINT
      # Restaurar copia de $v0 en memoria
            lw    $v0,sv_v0
      # Restaurar copia de $a0 en memoria
            lw    $a0,sv_a0
      # Restaurar copia de $ra en memoria
            lw    $ra,sv_ra
# Volver al cuerpo principal del manejador
            j     continue_exception
```

En el cuestionario de esta sección hay preguntas relacionadas con el código que el alumno debe añadir para identificar la interrupción producida.

Rutina de tratamiento de la interrupción del contador/temporizador

Esta rutina simplemente presenta un mensaje informativo por pantalla y vuelve al cuerpo principal del manejador de excepciones.

```
COUNTER_RTE:
      # Salvar registros
            sw    $a0,sv_a0   # Salvar $a0
            sw    $v0,sv_v0   # Salvar $v0
      # Escribir mensaje informativo
            li    $v0,4       # syscall 4 (print_str)
```

```
            la    $a0,COUNTER_MSG
            syscall
      # Restaurar registros
            lw    $a0,sv_a0   # Restaurar $a0
            lw    $v0,sv_v0   # Restaurar $v0
# Volver al cuerpo principal del manejador
            j     continue_exception
```

REALIZACIÓN DEL CUESTIONARIO 2.3.2

Abrir **MARS**, y entrar en **Exception Handler ...** dentro del menú **Settings**, activar la opción **Include this exception handler file in all assemble operations** y usar los botones **Browse** y **OK** para cargar el manejador incompleto del fichero **excepciones_rti.asm**.

Arrancar la herramienta **Digital Lab Sim**, que está en el menú **Tools**. Una vez arrancada, conectarla pulsando **Connect to MIPS**. Recordar que, después de cada prueba, para reiniciar la herramienta será preciso pulsar el botón **Reset** en la parte inferior de la misma.

A continuación, cargar el fichero **caso09.asm** con la opción **File -> Open**.

Llegados a este punto, es necesario editar el fichero **caso09.asm** para rellenar las operaciones que faltan, y que corresponden con la activación y desactivación de las interrupciones del contador/temporizador respectivamente.

1. **Supongamos que vamos a escribir en el registro $t0 un código mediante el cual posteriormente querremos activar las interrupciones del contador/temporizador. ¿Cuál de las siguientes operaciones serviría para ello?**
   a) li $t0,1
   b) li $t0,0
   c) lb $t0,0xffff0013
   d) Ninguna de las restantes respuestas es cierta.
2. **¿Cuál de las siguientes operaciones sirve para escribir el contenido de $t0 en el registro de control del contador/temporizador?**
   a) sb $t0,0xffff0010
   b) sb $t0,0xffff0011
   c) sb $t0,0xffff0012
   d) sb $t0,0xffff0013
   e) Ninguna de las restantes respuestas es cierta.

Escribir las dos operaciones anteriores en orden justo tras la etiqueta **activar**.

Guardar el fichero con el nombre **caso09-solucion.asm**. A continuación, ensamblarlo con **Run -> Assemble**, y hacer **Run -> Go** para ejecutarlo. Cuando se nos pida introducir un dato numérico por teclado, escribir **1** y pulsar **INTRO**.

3. **¿Cuántas interrupciones ha generado el contador?**
   a) Ninguna, porque no se han ejecutado suficientes instrucciones.
   b) Una.
   c) Cuatro.
   d) Ninguna de las restantes respuestas es cierta.

Pulsar **Clear** en la zona de mensajes y de la consola. Ejecutar el programa de nuevo, pero ahora introduciendo un **2** cuando se nos pida el dato numérico por pantalla.

4. **¿Cuántas interrupciones ha generado ahora el contador?**
   a) Ninguna, porque no se han ejecutado suficientes instrucciones.
   b) Una.
   c) Cuatro.
   d) Ninguna de las restantes respuestas es cierta.
5. **¿Cuál es el contenido del registro del coprocesador 0 BadVaddr ($8) en hexadecimal? (escribirlo con el prefijo 0x, tal como aparece en MARS)**
6. **¿Cuál es el contenido del registro del coprocesador 0 Status ($12) en hexadecimal? (escribirlo con el prefijo 0x, tal como aparece en MARS)**
7. **¿Cuál es el contenido del registro del coprocesador 0 Cause ($13) en hexadecimal? (escribirlo con el prefijo 0x, tal como aparece en MARS)**
8. **¿Cuál es el contenido del registro del coprocesador 0 EPC ($14) en hexadecimal? (escribirlo con el prefijo 0x, tal como aparece en MARS)**
9. **¿Cuál es el código de la causa de la excepción, expresado en base 10?**
10. **¿Cuál es el bit de interrupción pendiente que se ha activado en el registro Status? (escribir un número entre 8 y 15)**

Pulsar **Clear** en la zona de mensajes y de la consola. Ejecutar el programa de nuevo, pero ahora introduciendo un **10** cuando se nos pida el dato numérico por pantalla.

11. **¿Cuántas interrupciones ha generado ahora el contador?**
    a) Ninguna, porque no se han ejecutado suficientes instrucciones.
    b) Una.
    c) Cuatro.
    d) Ninguna de las restantes respuestas es cierta.

Cargar el fichero **excepciones_rti.asm** con la opción **File -> Open**, para editarlo rellenando el código faltante. Lo primero será incluir el código añadido en la actividad 2.2 en lo tocante al tratamiento de la excepción **syscall** y la correspondiente rutina de servicio. Será preciso poner atención en las definiciones incorporadas en la sección de datos, y también en el nuevo código (con comentarios y huecos) para la rutina genérica de tratamiento de interrupciones (a partir de la etiqueta **INT_RTE**) y la rutina

de tratamiento de la interrupción del contador/temporizador (a partir de la etiqueta COUNTER_RTE).

12. **En la rutina genérica de tratamiento de interrupciones, ¿cómo copiaríamos el contenido del registro Cause en el registro $k0?**
    a) mtc0 $k0,$12
    b) mfc0 $k0,$12
    c) mtc0 $k0,$13
    d) mfc0 $k0,$13
13. **Una vez copiado el contenido del registro Cause en $k0, ¿con qué operación averiguaríamos si se ha producido una interrupción del contador/temporizador?**
    a) andi $k0,$k0,0x00000100
    b) andi $k0,$k0,0x00000200
    c) andi $k0,$k0,0x00000400
    d) andi $k0,$k0,0x00000800
14. **¿Cuál sería el contenido de $k0 tras la operación anterior?**
    a) Tendría un único bit a 1 indicando la interrupción pendiente.
    b) Tendría escrito un número que en decimal indicaría el nivel de interrupción producido.
    c) Tendría el escrito un número que en hexadecimal indicaría el nivel de interrupción producido.
    d) Ninguna de las restantes respuestas es cierta.
15. **¿Cómo saltaríamos a la rutina específica de tratamiento de la interrupción del contador?**
    a) bne $k0,0x00000100,COUNTER_RTE
    b) bne $k0,0x00000200,COUNTER_RTE
    c) bne $k0,0x00000400,COUNTER_RTE
    d) bne $k0,0x00000800,COUNTER_RTE
    e) bne $k0,0,COUNTER_RTE

Modificar el código de la rutina de tratamiento de interrupciones con las operaciones elegidas en las preguntas 12, 13 y 15. Guardar el manejador en el fichero **excepciones_rti-solucion.asm**, cerrarlo con **File -> Close**.

Ahora, entrar en **Exception Handler ...** dentro del menú **Settings**, activar la opción **Include this exception handler file in all assemble operations** y usar los botones **Browse** y **OK** para cargar el manejador modificado del fichero **excepciones_rti-solucion.asm**.

Manteniendo abierto únicamente el fichero **caso09-solucion.asm**, ensamblarlo con **Run -> Assemble**. Poner un punto de ruptura en la instrucción etiquetada con **INT_RTE**. Recordar que es preciso mantener abierta y conectada a **MIPS** la herramienta Digital Lab Sim. Hacer un Reset sobre la misma pinchando el botón de la parte inferior. Ahora, hacer **Run -> Go** para ejecutar el programa. Cuando se nos pida introducir un dato numérico por teclado,

escribir **2** y pulsar **INTRO**. La ejecución se detendrá en el punto de entrada del manejador de interrupciones.

16. **¿Cuál es el contenido del registro del coprocesador 0 Status ($12) en hexadecimal? (escribirlo con el prefijo 0x, tal como aparece en MARS)**
17. **¿Cuál es el contenido del registro del coprocesador 0 Cause ($13) en hexadecimal? (escribirlo con el prefijo 0x, tal como aparece en MARS)**

Ejecuta una única instrucción.

18. **¿Cuál es el contenido del registro $k0 después de ejecutar la primera instrucción de la rutina de tratamiento de excepción? (escribirlo con el prefijo 0x, tal como aparece en MARS)**

Ejecuta únicamente la siguiente instrucción.

19. **¿Cuál es el contenido del registro $k0 después de ejecutar la segunda instrucción de la rutina de tratamiento de excepción? (escribirlo con el prefijo 0x, tal como aparece en MARS)**

Pon un punto de ruptura en la instrucción etiquetada con **COUNTER_RTE**. Ahora ejecuta hasta que el programa se detenga en ella. Si esto no sucede, corrige la respuesta a la pregunta 15, vuelve a editar el fichero **excepciones_rti-solucion.asm** de acuerdo con la nueva opción elegida y ejecuta de nuevo. Si la instrucción se detiene en **COUNTER_RTE**, ejecuta el programa hasta el final.

20. **Indicar cuáles de las siguientes respuestas son ciertas tras ejecutar el programa hasta el final:**
    a) Se ha presentado un mensaje diciendo que se ha producido una interrupción del contador/temporizador.
    b) El programa en curso ha presentado un mensaje de error y ha terminado su ejecución incorrectamente.
    c) El programa en curso ha continuado su ejecución normal tras salir del manejador de excepciones.
    d) El manejador ha abortado la ejecución del programa.
    e) Se han producido nuevas interrupciones que han requerido de la entrada en acción del manejador.

Pulsar el botón **Clear** de la zona de mensajes y de la consola, cerrar todos los ficheros abiertos mediante la opción **File -> Close All**, y cerrar **MARS**.

2.3.3. ESCRIBIR SOBRE EL REGISTRO DE CONTROL DEL CONTADOR/TEMPORIZADOR CON UN SERVICIO DEL SISTEMA OPERATIVO

INTRODUCCIÓN

Como la dirección del registro de control del contador/temporizador pertenece al espacio de direcciones de E/S, la manipulación directa del mismo debe hacerse desde código del sistema operativo, y no desde un programa de usuario. Para ello, implementaremos un servicio de sistema operativo operativo que se encargará de escribir sobre dicho registro de control y así activar o desactivar las interrupciones del contador/temporizador.

Junto con este ejercicio se planteará un cuestionario con una serie de preguntas que se responderán a través del Aula Virtual.

DEFINICIÓN DEL SERVICIO DE ACTIVACIÓN/DESACTIVACIÓN DE LAS INTERRUPCIONES DEL CONTADOR/TEMPORIZADOR

Los datos del servicio implementado son los siguientes:

- **Nombre del servicio**: set_counter.
- **Código del servicio** (en **$v0**): 1002.
- **Argumentos**:
  - **$a0**: dato que se quiere escribir en el registro de control. Si es 1, las interrupciones se activarán. Si es 0, las interrupciones se desactivarán. En otro caso, el servicio no escribirá sobre el registro de control.
- **Valor de retorno**: ninguno.

Para invocar este servicio, el programa principal:

- Escribirá en **$v0** el número del servicio (1002).
- Escribirá en **$a0** un 1 si se pretende activar las interrupciones del contador/temporizador, ó un 0 si se pretende desactivarlas.
- Ejecutará la instrucción **syscall**.

El fichero **caso10.asm** contiene un programa que activa y desactiva las interrupciones del contador/temporizador mediante la instrucción **syscall**.

La rutina de tratamiento de llamada a sistema, codificada en espacio del *kernel*, se encargará de leer el código del servicio y sus argumentos, tras lo que realizará el servicio pedido. Si hubiera algún valor de retorno, la rutina de tratamiento de **syscall** lo pondría en el registro **$v0**.

Por tanto, en este ejercicio el alumno deberá modificar la rutina de tratamiento de la excepción **syscall** para que mire el valor de **$v0**, y si es igual a 1002, salte a la rutina del servicio **set_counter**, que actuará sobre el registro de control del contador/temporizador tal como se ha descrito.

DESCRIPCIÓN DEL MANEJADOR DE EXCEPCIONES

La estructura del manejador no cambia. Sin embargo, se ha añadido lo siguiente:

- En la sección de datos se incluye una definición de constante para dar nombre al servicio de activación y desactivación de interrupciones en el contador/temporizador.
- En la sección de código se ha incluido una rutina para el tratamiento del servicio de activación y desactivación de interrupciones del contador/temporizador. Esta rutina está incompleta, y deberá ser completada por el alumno, que, además, deberá realizar la identificación del servicio en la rutina de tratamiento del servicio **syscall**.

Código del servicio

Aquí se muestra la definición de la constante que identifica el código del servicio:

```
.eqv  SET_COUNTER,1002
```

Esta constante sirve para identificar el servicio solicitado dentro de la rutina de tratamiento de la excepción **syscall**.

Rutina de tratamiento de la excepción syscall

En esta rutina se añadirá el código que la permitirá reconocer si se ha solicitado el servicio de activación/desactivación de interrupciones del contador/temporizador, y en caso afirmativo saltará a la correspondiente rutina de servicio, cuyo punto de entrada está en la etiqueta **SET_COUNTER_RTE**. Este código se añadirá al realizado en la actividad 2.2 en cuanto al servicio de escritura en los visualizadores de 7 segmentos.

Rutina de servicio para activar/desactivar las interrupciones del contador/temporizador

Esta rutina comprueba el valor del parámetro **$a0**, y si es menor o igual que 1 escribe su valor sobre el registro de control del contador/temporizador, activando o desactivando las interrupciones del mismo. Si el contenido de **$a0** es mayor que 1, entonces no escribirá nada en el registro de control. En todo caso, al terminar su tarea la rutina de servicio saltará a la rutina de tratamiento de la excepción **syscall** con **j continue_syscall**.

```
SET_COUNTER_RTE:
##### EL ALUMNO ESCRIBIRÁ AQUÍ EL CÓDIGO ENSAMBLADOR
##### QUE IMPLEMENTE LA RUTINA DE SERVICIO
# Si $a0=1 ó $a0=0, escribir sobre el registro de control del contador
# para activar o desactivar las interrupciones y saltar a continue_syscall
# Si $a0 no es ni 0 ni 1, saltar directamente a continue_syscall
# sin escribir en el registro de control del contador
            j     continue_syscall
```

Esta rutina debe ser completada por el alumno.

REALIZACIÓN DEL CUESTIONARIO 2.3.3

Abrir **MARS**, y cargar el fichero **excepciones_rti-solucion.asm** con la opción **File -> Open**, para editarlo de nuevo rellenando el código faltante en cuanto al tratamiento de la excepción por syscall y la correspondiente rutina de servicio. Es preciso poner atención otra vez en las nuevas constantes definidas en la sección de datos, y también en el nuevo código (con comentarios y huecos) para las rutinas de tratamiento de **syscall** y servicio de escritura en los visualizadores.

1. **En la rutina de tratamiento de syscall, ¿cuál de las siguientes operaciones sirve para chequear el número de servicio de activación y desactivación de las interrupciones del contador/temporizador, y saltar en su caso a la rutina de servicio correspondiente?**
   a) j continue_syscall
   b) beq $v0, SET_COUNTER,SET_COUNTER_RTE
   c) bne $v0, SET_COUNTER,SET_COUNTER_RTE
   d) Ninguna de las restantes respuestas es cierta.
2. **En la rutina del servicio de escritura sobre los visualizadores, ¿qué operación serviría para salir correctamente de la rutina de servicio sin escribir en el registro de control del contador/temporizador cuando $a0 no vale ni 0 ni 1?**
   a) bne $a0,0,continue_syscall
   b) bne $a0,1,continue_syscall
   c) rte
   d) jr $ra
   e) bgeu $a0,2,continue_syscall
   f) Ninguna de las restantes respuestas es cierta.
3. **¿Cuál de las siguientes operaciones sirve para escribir el contenido de $a0 en el registro de control del contador/temporizador?**
   a) sb $a0,0xffff0010
   b) sb $a0,0xffff0011
   c) sb $a0,0xffff0012
   d) sb $a0,0xffff0013
   e) Ninguna de las restantes respuestas es cierta.

4. **En la rutina del servicio de activación/desactivación de las interrupciones del contador/temporizador, ¿para qué puede servir la operación j continue_syscall?**
   a) Para saltar a **continue_syscall** una vez hecho el trabajo requerido.
   b) Para saltar a **continue_syscall** cuando se ha cometido un error de escritura en el registro de control.
   c) Esa instrucción está mal.Para hacerlo bien hay que cambiarla por la instrucción **j continue_exception**.
   d) Ninguna de las restantes respuestas es cierta.

Es preciso rellenar el código que falta en la rutina de **syscall** y en la rutina de servicio, y grabar el fichero resultante con el nombre **excepciones_contador-solucion.asm**. Después, entrar en **Exception Handler ...** dentro del menú **Settings**, activar la opción **Include this exception handler file in all assemble operations** y usar los botones **Browse** y **OK** para cargar el manejador del fichero recién grabado. Cerrar todos los ficheros abiertos con **File -> Close All**.

A continuación, cargar el fichero **caso10.asm** con la opción **File -> Open**, y ensamblarlo con la opción **Run -> Assemble**. Después, ejecutar el programa mediante la opción **Run -> Go**. Cuando se pida un dato numérico por teclado, escribir un **2**. Probar que el comportamiento del programa es el mismo que el del caso 9. Si no, abrir el fichero **excepciones_contador-solucion.asm** y editarlo de nuevo. No olvidar grabarlo y cerrarlo antes de realizar pruebas adicionales.

Pulsar el botón **Clear** de la zona de mensajes y de la consola, cerrar todos los ficheros abiertos mediante la opción **File -> Close All**, y cerrar **MARS**.