Proyecto de Ingeniería de Computadores (PEC)

Subetapa 7.2: Implementación de las excepciones

Tal y como ya se mostró en el anexo de la etapa 7.1, aparte de las interrupciones, en el procesador SISA también pueden producirse excepciones. Las excepciones son un suceso no esperado ("excepcional") que se produce internamente en el procesador.

Las excepciones son fundamentalmente para que un Sistema Operativo pueda funcionar correctamente. Por ejemplo, con la excepción del *miss* de la TLB se podría implementar el "*copy on write*" que tienen la mayoría de sistema operativos, incluyendo el Linux.

Los principales sucesos que provocan una excepción en nuestro procesador son los siguientes:

- Ejecución de una instrucción ilegal
- Acceso no alineado a memoria
- Desbordamiento (overflow) en una operación de coma flotante
- División por cero en coma flotante
- División por cero en enteros o naturales
- Acceso a memoria protegida
- Instrucción protegida
- Instrucción **`CALLS`**

Las excepciones son tratadas por el procesador casi de la misma forma que las interrupciones. Cuando se produce una excepción se realiza una ruptura del secuenciamiento al igual que una interrupción y se llama a la rutina RSG. La rutina RSG detectará si se trata de una interrupción o de una excepción y en este último caso de que excepción se trata. Una vez identificada se llama a la RSE (Rutina de Servicio de Excepción) de la excepción que corresponda con el mismo modo de funcionamiento que las interrupciones.

Las excepciones pueden ser enmascarables o no. Las enmascarables son aquellas que el programador puede impedir que se produzcan (o puede hacer que el procesador las ignore). Según la documentación del SISA la única excepción enmascarable es la excepción de overflow para operaciones de coma flotante. Para indicar si están enmascaradas o no el procesador dispone de un bit en la palabra de estado (PSW<2>). El resto de excepciones en el SISA no pueden ser enmascaradas.

Para que la RSG pueda identificar si se trata de una interrupción o de una excepción, y en este caso de que excepción se trata, debe consultar el valor del registro S2. Ahora el registro S2 no sólo contiene el valor 15 si no que contiene el identificador de excepción. Si su valor es 15 se trata de una interrupción y si su valor es menor entonces es una excepción y este valor indica la excepción concreta de que se trata.

Al igual que las interrupciones, tendremos un vector de excepciones y un comportamiento por defecto. Los comportamientos por defecto habituales pueden ser no hacer nada o parar el procesador. Igualmente que con el vector de interrupciones, el contenido del vector de interrupciones se puede cambiar en cualquier momento reprogramando las direcciones de inicio de las RSE.

Listado de excepciones

Las excepciones que generará esta implementación del SISA son:

- Excepción 0: Instrucción ilegal. Esta excepción se produce cuando el decodificador del procesador se encuentra con un código de instrucción que no tiene implementado.
- Excepción 1: Alineación impar. Se produce cuando se hace un fetch de una instrucción que sea en una dirección de memoria impar o se produzca un load o store de word con una dirección impar.
- Excepción 2: Overflow en operación de coma flotante.

- Excepción 3: División por cero en coma flotante. Esta excepción se da cuando la operación que debe realizar la ALU es una división de coma flotante y el segundo operando es un cero.
- Excepción 4: División por cero. Esta excepción se da cuando la operación que debe realizar la ALU es una división de enteros o naturales y el segundo operando es un cero.
- Excepción 6: Miss en TLB de instrucciones. Se lanza esta excepción cuando se hace un fetch de una dirección, la página de la que no está al TLB y por tanto no puede traducirse en dirección física.
- Excepción 7: Miss en TLB de datos. Esta excepción se produce cuando se intenta hacer un load o store en una dirección, la página de la que no está al TLB de datos y no se puede traducir en dirección física.
- Excepción 8: Página inválida al TLB de instrucciones. Se produce cuando se intenta hacer un fetch de una dirección que está en una página con el bit de inválida al TLB.
- Excepción 9: Página inválida al TLB de datos. Igual que el anterior pero cuando se realiza en una página referenciada con un load o store.
- Excepción 10: Página protegida al TLB de instrucciones. Se hace un fetch en modo usuario de una página que pertenece al sistema operativo (tiene el bit de protección activado) y, por tanto, sólo él puede ejecutar código de esta página.
- Excepción 11: Página protegida al TLB de datos. De manera similar a la anterior, el programa de usuario está ejecutándose en modo no privilegiado e intenta leer (load) o escribir (store) en una página que pertenece al sistema operativo (tiene el bit de protección activado).
- Excepción 12: Página de sólo lectura. Esta excepción se lanza cuando una instrucción de store hace un acceso a una página que tiene el bit de sólo lectura activado el TLB de datos.
- Excepción 13: Excepción de protección. Se lanza cuando se está ejecutando una instrucción de entrada/salida o de sistema y en cambio el procesador está en modo usuario.
- Excepción 14: Llamada al sistema. Esta no es propiamente una excepción sino que se produce cuando se realiza el **CALLS**.
- Excepción 15: Interrupción. Esta excepción no es tampoco una excepción sino que es cuando se produce una interrupción hardware, tal y como ya hemos implementado en la etapa anterior.

De las excepciones de TLB y de coma flotante de momento podemos olvidarnos ya que aún no están implementadas. La excepción 14 corresponde a la llamada al sistema con el **CALLS** que implementaremos más adelante y la excepción 15 ya la tenemos implementada porque corresponden a las interrupciones hardware respectivamente.

Uso de los registros de sistema

El funcionamiento de les excepciones es muy parecido a las interrupciones, por tanto el uso de los registros será prácticamente igual.

Los registros de sistema son 8 registros de 16 bits: S0, S1,...,S7. Su uso es el siguiente:

S0. Contiene la palabra de estado que tenía el sistema cuando se produjo una excepción (interna) o interrupción (externa). Cuando se produce uno de estos eventos, se salva en S0 la palabra de estado actual, que se encuentra en S7, ya que acto seguido se cambia la palabra de estado actual, para inhibir las interrupciones (externas).

S1. Contiene la dirección de retorno después de una interrupción (externa), una excepción (interna). Cuando se produce uno de estos eventos, se salva en S1 el PC actualizado, ya que acto seguido se copia en el PC el registro S5, que contiene la dirección del código de entrada al sistema operativo.

S2. Contiene, en sus cuatro bits de menor peso, un código binario que puede tomar valores de 0 a 15 que identifica el tipo de evento que se ha producido: tipo concreto de excepción (interna), llamada a sistema (**CALLS**) o interrupción (externa). El resto de bits de S1 están siempre a 0. El código es el descrito en el listado de excepciones.

S3. Contiene la dirección efectiva usada por una instrucción de acceso a memoria, cuando ésta provoca una excepción de acceso a memoria mal alineado.

S4. Puede ser usado por el programador como variable temporal.

S5. Contiene la dirección de memoria donde se encuentra la siguiente instrucción a ejecutar después de producirse una excepción (interna) o interrupción (externa). En esta dirección de memoria comienza el código de la rutina de servicio genérica (RSG) encargado de identificar la causa de excepción o interrupción y en cada caso llamar a la rutina de servicio de excepción (RSE) o a la rutina de servicio de interrupción (RSI) específica correspondiente.

S6. Su uso se reserva para ampliaciones futuras del lenguaje.

S7. Es el PSW (Processor Status Word). Contiene el estado del procesador en todo momento. Cada bit tiene el siguiente significado:

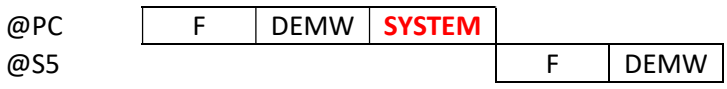
Bit M (PSW<0>): Se reserva para ampliaciones futuras del lenguaje.

Bit I (PSW<1>): Bit que indica si las interrupciones están permitidas (en cuyo caso vale 1) o si están inhibidas (en cuyo caso vale 0). Este bit se modifica con las instrucciones **`EI`** (Enable Interrupt) y **`DI`** (Disable Interrupt), que lo ponen a 1 o a 0 respectivamente y la instrucción **`RETI`** que restaura la palabra de estado y retorna de una excepción o interrupción. También, se pone a 0 cuando se produce una excepción o interrupción.

Bit V (PSW<2>): Bit que indica si la excepción de overflow en operación de coma flotante está permitida (en cuyo caso vale 1) o si está inhibida (en cuyo caso vale 0).

Tratamiento de la excepción

Se tratará casi igual que las interrupciones. Se usará el mismo estado extra (SYSTEM) que ya se ha implementado en las interrupciones. La secuencia de estados es la siguiente.



Acciones en una excepción

Son muy parecidas a las de las interrupciones. El ciclo nuevo al que hemos llamado SYSTEM que se encarga de realizar todas las tareas necesarias para poder ejecutar la primera instrucción correspondiente a la rutina de servicio genérica (RSG). Dicha dirección estaba almacenada en S5. Las principales diferencias con las interrupciones son que ahora el registro S2 guarda el código del evento que se ha producido (excepción o interrupción), para informar a la rutina de servicio (RSG) de lo ocurrido y que el registro S3 almacena, en el caso de una excepción de acceso mal alineado a memoria, la dirección efectiva impar utilizada por la instrucción que provocó la excepción.

En resumen, debe realizar las siguientes tareas:

- S0 ← S7 (se copia la palabra de estado PSW)
- S1 ← PCup (hace un backup del PC de retorno)
- S2 ← #codigo (indica el código del tipo de evento ocurrido)
- S3 ← @efectiva (indica la dirección que ha provocado el acceso no alineado)
- PC ← S5 (rutina de sistema a la que se llama)
- S7<1> ← 0 (se cambia la PSW inhibiendo las interrupciones)

Finalmente, el procesador debe inicializarse con las interrupciones inhibidas ya que hasta que no se haya almacenado en el registro S5 la dirección de la RSG, si llegase alguna interrupción el comportamiento del procesador seria impredecible.

Modificaciones en los componentes de procesador

Primero analizaremos los cambios que hay que hacer en los módulos de ejecución del procesador como hemos hecho habitualmente hasta ahora.

ALU

En primer lugar, habrá que cambiar la ALU para que pueda informar cuando se produce la situación de división por cero. Entonces se producirá una excepción de tipo 4 (división por cero). Lo habitual es añadir una salida (típicamente llamada div_zero) que detecta esta situación cuando la función a realizar es una división.

Banco de registros

También tenemos que cambiar ligeramente el banco de registros. Cuando se produzca una excepción de tipo 1 (memoria mal alineada) el banco de registros debe almacenar en S3 la dirección que ha provocado el fallo.

Por lo tanto, habrá que llevar el valor de la dirección (bus addr_m) hasta el banco de registros y la una señal que indique que se ha producido una excepción del tipo acceso no alineado desde el lugar donde hayáis implementado esa detección (controlador de memoria, TLB, …)

Además, el registro S2 deberá almacenar el código correspondiente al tipo de excepción producido o si es una interrupción. Puede ser interesante añadir una única entrada de 4 bits al banco de registros por donde ya venga codificado el valor adecuado.

Cambios en el controlador de memoria

Ampliaremos el controlador de memoria para detectar la situación de alineación incorrecta en memoria (no es el único lugar donde puede implementarse esta función). Esto se produce cuando se hace un acceso de tipo word y la dirección es impar. El acceso entonces debería hacerse en dos ciclos y debido a ello se lanza una excepción. Este acceso puede suceder en la etapa de fetch o en la de ejecución, aunque nosotros no distinguiremos estas dos situaciones. En ambos casos se lanzará la excepción del tipo 1 (alineación impar).

Módulo de control

Sólo tenemos que añadir la lógica de activación para que se produzca una excepción del tipo 0 (instrucción ilegal)

Módulo de excepciones

De momento sólo tenemos que procesar las excepciones de tipo 0, 1, 4 y 15, pero puede ser práctico crear un nuevo módulo para gestionar las excepciones ya que más adelante tendremos bastantes más.

La función principal de este seria recibir todas las señales que indican que se ha producido una excepción i generar el identificador de excepción que se enviaría al banco de registros para que lo almacenase en S2.

Juegos de pruebas

Realizad un juego de pruebas que provoque y trate todas estas excepciones.