**Privilegios e Interrupciones RISC-V**

**Autor: Bruno Burgos Kosmalski**

**Indice**

* **Aclaración Inicial**
* **Introducción**
* **Niveles de Privilegios**
* **Registros de Control y Estado (CSR)**
* **Instrucciones Privilegiadas Adicionales**
* **Resumen**
* **Información Adicional (Controladores) (Por hacer)**
* **Listado de Algunos de los Registros de Control y Estado**
* **Terminología Empleada**
* **Referencias (Por hacer)**

**Aclaración Inicial:**

En este documento intenta dar una visión inicial del comportamiento del estándar frente a los niveles de privilegios sobre los que ejecuta, junto a las interrupciones y excepciones. Para esto se hace referencia en todo momento al estándar oficial (versión actual 2024).

**Introducción:**

Para entender el comportamiento de las interrupciones y excepciones dentro del estándar, es necesario entender primero el concepto sobre el que se fundamenta. RISC-V es un estándar que parece más orientado en un primer momento al tratamiento de las traps a distintos niveles de ejecución, que al propio tratamiento de las interrupciones que han podido generar dispositivos externos. Esto es algo curioso teniendo en cuenta que uno de los campos en los que más destaca actualmente es en el de los sistemas empotrados, donde aparentemente el control de las interrupciones externas es más prioritario que la propia gestión de privilegios.

El estándar justifica este punto de vista como un intento de dar soporte a distintos formatos de software, desde sistemas con un único nivel de ejecución, pasando por sistemas de tipo Unix, hasta sistemas de multiprocesamiento más complejos que están gestionados por distintos sistemas operativos. Está claro que para dar soporte a estos sistemas se necesita una jerarquía de privilegios, lo que nos lleva al siguiente apartado.

Diagrama, Texto

El contenido generado por IA puede ser incorrecto.La figura 1.1 muestra tres ejemplos de sistemas que están soportados por el estándar.

Figura 1.1

**Niveles de Privilegios:**

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.El estándar tiene definidos 3 modos de ejecución, sin embargo, dependiendo del fabricante se podrían implementar otros a demanda de la necesidad. Estos niveles de privilegios son: Usuario/Aplicación, Supervisor y, Máquina; donde el modo máquina es aquel que tiene mayor nivel de privilegios, seguido por el supervisor y llegando al usuario como el que tiene la menor cantidad. Es precisamente por esto que cualquier implementación del estándar tiene que tener definido el nivel de máquina, junto a los sus registros de control, siendo el resto de los modos de operación completamente opcionales.

Figura 1.2

El estándar está planteado para que cuando se trate de ejecutar una instrucción sin los permisos suficientes, se genere una excepción, entrando en una de las traps del entorno. Normalmente el código ejecuta en nivel de aplicación hasta que entra en una trap, que puede ser una llamada al sistema, o una interrupción. El tratamiento de estas traps puede ser vertical, escalado a un nivel superior de ejecución, u horizontal, manteniendo el nivel de ejecución. Esto no está definido de una manera rigurosa, sino que el propio estándar permite cierta flexibilidad en el enrutamiento de las traps.

Adicionalmente y, dependiendo del sistema que se plantee implementar, como se ha mencionado al comienzo del apartado, puede ser que se necesiten modos de ejecución adicionales. Siendo los más típicos el hypervisor para algunas implementaciones con jerarquías más complejas, y el de debug, siendo este un modo de ejecución que puede tener más privilegios que el propio modo máquina, donde puede reservar algunos de los registros de control e incluso partes del espacio de memoria de la máquina.

Imagen que contiene Tabla

El contenido generado por IA puede ser incorrecto.

Figura 1.3

La figura 1.3 muestra algunos ejemplos de los modos que se necesitarían implementar para algunos de los sistemas más típicos.

**Registros de Control y Estado (CSR):**

Cada nivel de privilegios tiene una serie de registros de control y estado con los que tiene que contar para poder trabajar de manera correcta. Donde los registros asociados a un nivel de privilegios son solo visibles por un nivel igual o superior. Estos registros de control y estado comprenden un espectro bastante amplio de funcionalidades, como puede ser: el manejo de interrupciones y excepciones, la delegación del tratamiento de las mismas, un registro de los ciclos, paginación, etc.

Para trabajar con estos registros de control y estado, sobre todo con los registros internos (hw) tenemos básicamente dos tipos de instrucciones: las que trabajan de forma atómica y nacen de la extensión “Zicsr” (las instrucciones csr), y las que son derivadas de las mismas. Antes de explicar el funcionamiento de las principales instrucciones del tipo “csr” para el trabajo con los registros, cabe mencionar que en este documento se adjunta un listado con algunos de los registros de control y estado contemplados por el estándar.

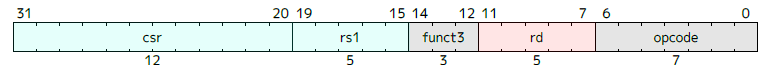
****Las instrucciones csr son instrucciones que normalmente garantizan la atomicidad de la operación, donde escriben, leen o modifican un registro de control o estado. Si se intenta acceder a un registro inexistente o del que no se tienen privilegios suficientes se levantará una excepción. Estas instrucciones pueden tomar uno de dos formatos: csrXX rd, csr, rs1, para trabajar con un registro fuente, o csrXX rd, csr, imm para trabajar con datos inmediatos.

Figura 1.4

Solo hay tres tipos de instrucciones csr: la de lectura y escritura (csrrw), la de lectura y clear (csrrc), y la de lectura y set (csrrs), tambien se cuenta con las opciones con datos inmediatos (csrrwi, csrrci, csrrsi). Para el formato de escritura se recupera sobre el registro destino (rd) el valor del registro de control (csr), y se carga sobre el mismo el valor fuente (rs1/imm). Los formatos clear y set son muy parecidos al anterior en el sentido de que la lectura se realiza de la misma manera, siendo el único cambio el no escribir sobre todo el registro de control, sino solo los bits que sea necesario: ponerlos a uno (set) o a cero (clear).

Por el código de operación que toman las instrucciones csr podríamos tener diferenciados hasta 4096 registros de control y estado posibles. Sin embargo, en el estándar hay especificados a penas unas decenas, aunque, si permite la definición de nuevos registros de control por parte del fabricante, siempre que no sea incompatible con la implementación del estándar ya descrita. Añadiendo a este último punto, con el tiempo han surgido algunas implementaciones no estándar para un manejo más complejo de interrupciones, que cuentan con la declaración de nuevos registros, aunque muchos de ellos mapeados a memoria, se verá en más detalles en el apartado de controladores.

Entendiendo que cada modo de ejecución tiene una serie de registros de control muy parecidos en funcionamiento al del resto de modos, solo diferenciándose en la identificación numérica y la primera letra en el nombre para identificar al modo al que pertenecen. Parece conveniente solo explicar algunos de los registros de estado y control, pero solo para el modo máquina, que son los que tienen que estar siempre implementados. Añadiendo a esto, se intentará focalizar el centro de atención en aquellos que forman parte del control de las interrupciones y excepciones.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.Empezando por el principal registro de estado y control, mstatus (machine status), este registro nos da información del sistema, donde los primeros veintidós bits tienen un propósito específico en el estándar, sin embargo, los que nos interesan son aquellos que habilitan las interrupciones y excepciones, que en este caso son los XIE donde X es el identificador del modo de ejecución.

Figura 1.5

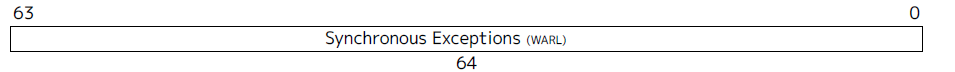
En la figura 1.5 podemos ver como este es el bit número 3, o el bit MIE (machine interrupt enable) del registro. Por otra parte, si queremos interrupciones anidadas tendremos que activar el bit MPIE (machine previous interrupt enable), donde el valor del nivel de privilegios anterior se almacena en los dos bits MPP (machine previous privilage).

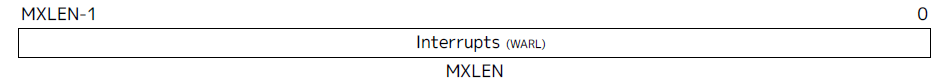
El registro mtcev ( machine trap vector interrupt) es el principal encargado de la vectorización de las interrupciones y excepciones. En este registro se guarda la dirección base sobre la que se calculará el salto cuando se produzca una interrupción o excepción. Como la dirección tiene que estar alineada a palabra ( 4 bytes ) se usan los dos bits menos significativos para indicar el modo de operación: cuando están los dos puestos a cero entonces el salto es directo y no se calcula ni diferencia nada, mientras que cuando el valor que representan es 1, al generarse una interrupción se añade sobre esta dirección el producto por cuatro del valor numérico que representa la causa de la interrupción, se verá en más detalle con el registro mcause.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto. Figura 1.6

Por defecto las interrupciones y excepciones se tratan en el modo máquina, sin embargo, como se mencionaba al principio del apartado, se pueden delegar a niveles inferiores. Para esto es necesario implementar dos registros de control que permiten llevar a cabo esta tarea: mideleg ( machine interruption delegataion register ) y, medeleg ( machine exception delegation register ). Nótese que para las implementaciones con modo supervisor, estos registros se tendrán que implementar de manera obligatoria.

 Figura 1.7 Registro mdeleg

**** Figura 1.8 Registro mideleg

Para el manejo de las interrupciones tenemos los registros de interrupción: mie (machine interrupt enable register ), y mip ( machine interrupt pending register ). Donde los 16 bits menos significativos de estos dos registros están asociados a valores específicos del estándar, de forma completamente simétrica. Podemos ver por ejemplo las interrupciones del timer y las interrupciones software, donde los16 bits más significativos están preparados para que el fabricante pueda particularizar la implementación a necesidad. **Interfaz de usuario gráfica, Tabla, Escala de tiempo

El contenido generado por IA puede ser incorrecto.**Figura 1.9 Imagen sacada del estándar oficial

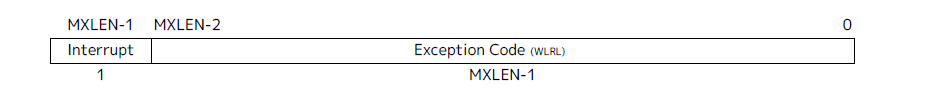
Como se había mencionado al comienzo del documento, este estándar está más pensado en el tratamiento de excepciones a distintos niveles de privilegios que en las propias interrupciones, siendo este registro el claro ejemplo. Podemos seccionar las interrupciones externas en dos, las interrupciones de timer, y el resto, como vemos en la imagen las interrupciones de timer tienen dos bits dedicados en el registro: MTIE (machine timer interrupt enable ) y, STIE ( supervisor timer interrupt enable ), siendo la misma cantidad de bits dedicados para todo el resto de interrupciones externas: MEIE (machine external interruption enable) y, SEIE (supervisor external interruption enable), lo que para muchos sistemas es claramente insuficiente. Es decir, para los periféricos externos tenemos una única línea de interrupción, donde la prioridad se trata de manera externa, por ejemplo, con un controlador.

Una vez que se interrumpe, ya sea una interrupción externa o una excepción, se almacena el contador de programa en un registro específico diseñado para esto: mepc (machine execution program counter), siendo que se usará este registro en las instrucciones de retorno de interrupción (mret, sret, uret, …), si por ejemplo, se ha generado una excepción y queremos tratarla saltando a la siguiente instrucción, entonces tendremos que sumarle cuatro al valor del registro, almacenarlo, y retornar de la interrupción.

Como se ha mencionado con el registro de vectorización, las interrupciones y excepciones tienen un valor específico que las identifica, o al menos la causa que las ha generado. Para este propósito tenemos el registro mcause (machine cause register), en el que, cuando se genera una interrupción, se almacena el valor numérico asociado a la causa de la misma, lo que se puede usar para identificar el tipo de la misma.

Las figuras 1.10 y 1.11 muestran el formato del registro y las codificaciones de las causas definidas por el estándar.

**Tabla

El contenido generado por IA puede ser incorrecto.**Figura 1.10 Estructura del registro mcause

**Tabla

El contenido generado por IA puede ser incorrecto.**

**Tabla

El contenido generado por IA puede ser incorrecto.**

Figura 1.11 Tabla con las causas definidas en el estándar.

Dentro del estándar también se ha definido la prioridad de ejecución de las excepciones síncronas, esto se puede ver en la figura 1.12 sacada directamente del estándar.**Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.**Figura 1.12

Para identificar además de la causa de la interrupción, información adicional, se implementa el registro mtval (machine trap value), que está pensado como asistente para el tratamiento de las interrupciones y excepciones. Cuando se genera una interrupción o excepción este registro o bien tiene el valor 0, o bien contiene información específica de la interrupción o excepción. Por ejemplo, cuando se hace un acceso desalineado, este registro puede contener una parte de la dirección virtual que ha generado la interrupción, mientras que de otra manera solo tendríamos el punto del programa en que se generó.

Como se había mencionado antes, las interrupciones externas se desglosan en dos tipos principales, las interrupciones de timer y el resto. Entonces vemos una definición bastante completa del funcionamiento del timer en el estándar, donde también nos define dos registros de control específicos, mapeados a memoria, para el funcionamiento correcto de este dispositivo: mtime, y mtimecmp.

Los registros mtime y mtimecmp son ambos registros de 64 bits, donde el registro mtime marca el valor actual del timer, siendo un registro que por definición se tiene que ir incrementado al mismo ritmo que el del procesador. Mientras que mtimecmp contiene un valor estático que una vez alcanzado por mtime generará una interrupción de tipo timer. Estos dos registros se pueden modificar, tanto para reiniciar el timer, como para especificar tras cuanto tiempo se debe interrumpir.

**Instrucciones privilegiadas adicionales:**

En este apartado se van a mencionar dos instrucciones que pueden ser útiles en el manejo de las interrupciones o excepciones: mret (Xret) y WFI. Una vez se he generado una excepción o interrupción para poder retornar de la rutina de tratamiento de forma sencilla, podemos ejecutar una de las instrucciones de la serie xret, donde x corresponde al modo de ejecución (mret para modo máquina), esta instrucción restaura el modo de ejecución y salta al punto en el que se encontraba cuando se generó la interrupción. Esta es básicamente la instrucción de retorno de interrupción, aunque también puede tener otros usos, como para el cambio de modo de ejecución.

La siguiente instrucción que puede ser de interés cuando se está trabajando con interrupciones es la instrucción wfi (wait for interrupt), que en este caso es directamente una instrucción que por definición se queda a la espera de una interrupción, gastando la menor cantidad de recursos posible.

**Resumen**

Para resumir el trabajo del estándar que se ha descrito en los anteriores apartados, a continuación, se describirá el proceso de un proyecto de entrada salida ejemplo. Por mantener un carácter genérico no se van a dar detalles específicos de la implementación.

Lo primero que vamos a necesitar son los modos de operación que se van a implementar, o sobre los que se van a trabajar. Por simplicidad, se va a plantea un sistema con solo dos modos de operación, el principal y obligatorio, el modo máquina (M), y un modo secundario para trabajar con menos privilegios, modo supervisor (S).

Lo siguiente será seleccionar las excepciones que se van a cubrir junto con los periféricos con los que se va a trabajar. Se tendrá que reservar un espacio de memoria para la vectorización y el tratamiento de las excepciones, para esto último, se necesitará conocer el valor de causa y calcular la vectorización sobre cada una de ellas.

Luego, suponiendo que las interrupciones se manejan en el nivel superior (M), mientras que el resto del trabajo se hace en el nivel inferior (S), y entendiendo que se inicia en el modo superior (M), se inicializarán los registros pertinentes, tanto internos del procesador como los del periférico. Dentro del esquema de los registros de control internos se tendría que seguir un orden parecido al siguiente: inicializar el registro de vectorización al valor de la dirección base + 1 (mtvec), se tendría que actualizar el registro de interrupciones para habilitar las interrupciones del periférico (mie), se actualizaría el registro que contine la dirección de salto al del inicio del programa (mepc), se inicializaría el registro de estado mstatus con el valor actualizado en los bits mpp y mpie, en este caso 1 para ambas regiones, básicamente cargar sobre este registro el valor 0x880. Y una vez inicializados los registros, y lo que se crea conveniente para el manejo del periférico, se retornará con la instrucción mret para cambiar de modo de ejecución al especificado en los registros.

En la figura 1.13 se presenta un código ejemplo que puede servir para la inicialización de los registros internos y del sistema.

Texto

El contenido generado por IA puede ser incorrecto.Figura 1.13 Código ejemplo y conceptual

A partir de este punto podríamos continuar con la ejecución del programa controlando solo aquellos registros que son internos del periférico, al menos, hasta que se genere una interrupción o excepción, donde se saltará a la dirección especificada, se cambiará automáticamente de modo de ejecución y se podrá proceder con el tratamiento de la interrupción. Normalmente, en la rutina de tratamiento tendremos que usar aquellos registros relacionados, tanto para el control de las interrupciones (mstatus, mtvec, mepc, mie, … ), como para obtener información de la misma (mcause, mtval, mip, … ).

El tratamiento tanto de interrupciones como de excepciones dependerá tanto del programador como de los propios periféricos y el uso que se les quiera dar. De hecho, este mismo ejemplo se podría extrapolar a sistemas más sencillos, con un único nivel de privilegios (M), o a sistemas más complejos, con varios niveles de privilegios (M, S, U), donde se tendrían que manejar los registros pertinentes al modo de operación que se vaya a encargar de las interrupciones.

**Información Adicional (Controladores):**

Para una mayor compatibilidad con las interrupciones externas, se han desarrollado una serie de controladores completamente compatibles con el estándar. Aunque, hay que tener en cuenta que no son completamente oficiales, a pesar de estar comprendidos en el repositorio oficial. Estos controladores son: PLIC (Plataform-Level Interrupt Controller) y CLIC, el actualmente ACLIC (Advanced Core Local Interrupt Controller).

Donde están planteados para una gestión más eficaz de las interrupciones, como añadir una jerarquía de prioridad, un mejor control de algunas de las interrupciones interprocesador, como cuando se tienen que manejar varios dispositivos tipo timer al mismo tiempo, y controlar el flujo de las interrupciones decidiendo en que contexto se va a interrumpir, por ejemplo, para multiprocesadores.

**Listado de Algunos de los Registros de Control y Estado del Estándar:**

Las siguientes imágenes hacen referencia al apartado 2.2 dentro del estándar, manual de instrucciones segundo volumen, donde se listan los registros de control descritos en el mismo.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Imagen de la pantalla de un celular de un mensaje en letras blancas

El contenido generado por IA puede ser incorrecto.

Imagen de la pantalla de un celular de un mensaje en letras blancas

El contenido generado por IA puede ser incorrecto.

Un periódico con texto

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

**Terminología Usada:**

Hart Hardware threads

OS Operating System

ABI Aplication Binary Interface

AEE Aplication Execution Environment

SBI Supervisor Binary Interface

SEE Supervisor Execution Environment

HBI Hypervisor Binary Interface

HEE Hypervisor Execution Environment

WARL Write Any / Read Legal

PMA Physical Memory Attributes

**Referencias:**

[X] PLIC : <https://github.com/riscv/riscv-plic-spec/blob/master/riscv-plic.adoc>

[X+1] ACLIC : <https://github.com/riscvarchive/riscv-aclint/blob/main/riscv-aclint.adoc>