**Enlaces:**

- What is the difference between mret and ret instruction in machine mode?:

<https://stackoverflow.com/questions/73103031/what-is-the-difference-between-mret-and-ret-instruction-in-machine-mode>

- Gestión de interrupciones en microcontroladores RISC-V:

<https://avelinoherrera.com/blog/comments.php?y=21&m=06&entry=entry210608-211213>

- URET or ERET:

<https://github.com/TheThirdOne/rars/issues/46>

- Estándar oficial:

<https://lf-riscv.atlassian.net/wiki/spaces/HOME/pages/16154769/RISC-V+Technical+Specifications>

- Antiguo 1.9:

<https://people.eecs.berkeley.edu/~krste/papers/riscv-privileged-v1.9.pdf>

- Para sistemas empotrados:

<https://five-embeddev.com/riscv-priv-isa-manual/Priv-v1.12/priv-intro.html>

**What is the difference between mret and ret instruction in machine mode?:**

El enlace corresponde a un foro de discusión donde se preguntaba por la diferencia entre las instrucciones “xRET” y “ret”. Se explica que la principal y más notoria diferencia entre “ret” y cualquiera de las “xRET” (mret, uret, sret) es que, mientras que ret es una pseudoinstrucción que se traduce de manera directa como: “jalr zero, 0(ra)” usando registros de propósito general. Las instrucciones “xRET” son directamente instrucciones que cambian los registros de control y permiten retornar de “traps” desde un modo de privilegio al que corresponda, por ejemplo, de supervisor a usuario, si está implementado.

**Gestión de interrupciones en microcontroladores RISC-V:**

<https://avelinoherrera.com/blog/comments.php?y=21&m=06&entry=entry210608-211213>

Artículo que explica algunas partes de las interrupciones del estándar y del funcionamiento en base a los distintos tipos o modos de ejecución. Cabe mencionar que hace referencia al estándar de 2019, por lo que algunas partes podrían estar obsoletas.

En el artículo:

- Introducción -> las arquitectura RISC-V está más planteada para la captura entre distintos niveles de privilegios, de ejecución, que, para la captura general de eventos externos, entrada salida. EL autor menciona que se echa de menos en microcontroladores y que por eso va a explicar la tecnología CLIC en este artículo, que esá relacionado a ello.

- Controlador básico de interrupciones -> explica, al igual que en el apartado anterior, que la arquitectura está más pensada para la captura en un nivel superior de eventos generados en un nivel inferior. Como, por ejemplo, desde modo máquina a usuario, y desde supervisor a máquina. También menciona la problemática que encontramos con la definición de interrupciones, que así como se las conoce solo hay de dos tipos: las de timer, y las “generales”. Menciona que esto es completamente ineficiente en el mundo de los microcontroladores donde a penas de cambia de nivel de ejecución pero si se trabaja bastante con las interrupciones. Por otro lado, para solo el timer es una solución ideal puesto que está explicado directamente en el estándar.

- Los CSR (registros de control y estado) -> muchos de los parámetros de funcionamiento se configuran en los registros de control, donde hay algunos definidos en el estándar que tienen que estar y otros que pueden añadir los fabricantes. Estos registros solo se pueden modificar con instrucciones específicas definidas en el estándar. Muestra como hay programas que hacen referencia directa a registros de control del estándar, pero que hay que tener cuidado con los que se añaden, como por ejemplo con el controlador avanzado de interrupciones (CLIC) que añade algunos registros de control que hay que definir.

- Ejemplo 1 - > En este ejemplo enseña el funcionamiento de algunos de los registros de control para trabajar con un dispositivo del tipo timer, que está el mejor explicado en el estándar. Para empezar habla del registro mtvec ( resgistro de vectorización ), registro que debe estar siempre presente, en este registro se guarda la dirección base sobre la que encuentra la rutina de tratamiento, donde se puede saltar directamente a la dirección o se puede aplicar vectorización sobre la misma, usando la causa de la interrupción, esto de declara en los dos bits menos significativos del registro puesto que no hacen falta ya que la direccion siempre tendrá que estar alineada a 4 bytes y por tanto siempre estarán puestos a cero. El segundo registro que hay que tocar es el mie (interrupt-enable , resgitro habilita interrupciones) en el que se usan principalmente sus 12 bits menos significativos para habilitar distintos tipos de interrupción, en este ejemplo se activa el bit 7 que es el que corresponde al timer en el estándar. Luego tendremos que tocal el registro mstatus (registro de estado) y habilitar las interrupciones de nivel máquina, lo que se hace activando el bit 3. Según está descrito en el estándar los dispositivos de tipo timer tendrán que tener dos registros de 64 bits mapeados a memoria, uno que esté constantemente auntoincrmentandose a la misma velocidad que la propia máquina, y otro que tenga un valor que cuando sea igual al que se autoincrementa interrumpa, las direcciones son de libre elección por el fabricante. Luego da un ejemplo escrito en cpp donde con ayuda de un linker define los principales valores de los registros de control como se ha dicho antes y se mete en un bucle infinito a esperar que interrumpa para reiniciar el timer y llamar al led para que se ilumine.

- Ejemplo 2 -> En este caso es el mismo ejemplo de antes pero esta vez el autor quiere usar CLIC como se había mencionado al principio.

Empieza describiendo CLIC (Core Local Interrupt Controler) diciendo que ya varios fabricantes importantes lo están implementado pero que todavía están a la espera de que se añada de forma oficial al estándar, donde CLIC además de añadir más registros de control lo que permite es generar hasta 4096 interrupciones, separarlaspor niveles de prioridad, etc. Ya está declarado en RISCV international (<https://github.com/pulp-platform/clic/blob/master/doc/clic.adoc> ) (<https://raw.githubusercontent.com/riscv/riscv-fast-interrupt/0b0083ee0af0cd88d59cdcf81e89cd3f9859e9ad/clic.pdf>)

Este formato es relativamente más complejo que el tratamiento estándar, a parte de manejar los registros de control normales, tener que reiniciar los 4096 registros de control mapeados en memoria de la herramienta, declarar el nivel de prioridad, habilitar las interrupciones, declarar todas las secciones, etc. Todo el código subido a en el enlace (<https://avelinoherrera.com/soft/>) en la sección de RISCV, para estos ejemplo son las secciones de más abajo del apartado, es decir, las 3 últimas secciones.

**URET or ERET:**

Es una contestación del creador de rars que dice que ha actualizado las llamadas de los registros porque ahora los llaman de una manera distinta. ;

Sin embargo, la mención que hace no aparece en la documentación oficial actual, al menos no en el manual de instrucciones privilegiadas.

Instrucción xRET para regresar de una instrucción si esta está en el mismo nivel, poe ejemplo U para usuario.

**Gestión de interrupciones en microcontroladores RISC-V:**

<https://avelinoherrera.com/blog/comments.php?y=21&m=06&entry=entry210608-211213>

Este es un enlace a una página web donde se ha seccionado el estándar oficial y se ha puesto en secciones resumidas. Se muestran los cambios junto a algunos ejemplos de código.

Arquitectura con privilegios según el estándar :

- Introducción:

Hay distintos formatos software que soportados por el estándar, desde sistemas simples que corren solo desde el nivel de aplicación, sistemas de multiprocesamiento en los que se ejecutan varias aplicaciones al mismo tiempo, y sistemas más complejos en lo que se ejecutan ya no solo varias aplicaciones al mismo tiempo sino que se tienen distintos superusuarios o supervisores. Esto se muestra resumido en la siguiente imagen.

Diagrama, Texto

El contenido generado por IA puede ser incorrecto.

Normalmente las implementaciones Hardware del estándar necesitan características adicionales para soportar distintos niveles de ejecución.

Niveles de privilegios: El estándar tiene definidos cuatro niveles de privilegios en su codificación que impactarán en los registros de estado y control. Esta implementación se muestra resumida en la siguiente tabla.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

El intento de ejecutar una instrucción que no está permitida en ese nivel de ejecución generará una excepción. Normalmente este tipo de excepciones entraran en las trapas del entorno de ejecución.

El modo máquina (M) es el único modo que es de obligada implementación en el estándar, este modo es el que tiene mayor nivel de privilegios. Mientras que los modos de usuario y supervisor están pensados para ser ejecutados a nivel de aplicación/usuario o sistema operativo donde sus privilegios pueden variar, todo depende del entorno de ejecución definido.

Hay varias extensiones y variantes para los distintos niveles de ejecución, como por ejemplo el uso del modo supervisor para una implementación de hypervisor tipo-2.

La siguiente imagen muestra un ejemplo de los modos de ejecución que suelen implementar los sistemas del tipo.

Imagen que contiene Tabla

El contenido generado por IA puede ser incorrecto.

El código ejecuta en el nivel de ejecución en el que se encuentre, generalmente usuario, hasta que entra en una trap (por una llamada al supervisor o una interrupción). Esto puede ejecutarse en un nivel superior de privilegios (traps verticales) o en el mismo nivel (traps horizontales). EL estándar permite un enrutamiento flexible de los traps.

Adicionalmente, las implementaciones pueden añadir también el modo de debug (D), que podría tener incluso más control que el propio modo máquina, este puede reservar algunos de los registros de control, e incluso puede reservar porciones de los propios espacios de memoria de la propia máquina.

- Registros de Control y Estado (CSR):

Para poder trabajar con los registros de control hay dos tipos de instrucciones con privilegios, aquellas que trabajan atómicamente, y todo el resto. Las que trabajan atómicamente son aquellas implementadas en la extensión Zicsr mientras que el resto dependen de esta extensión, es decir para implementar distintos niveles de privilegios se necesita esta extensión.

Las instrucciones csr están mapeadas de tal forma que se podría programar hasta 4096 registros de control. Por otra parte, si se intenta acceder a un registro de control inexistente o se intenta acceder a un registro de control sin los privilegios adecuados entonces se genera una excepción de instrucción ilegal.

Abajo se adjunta un listado con los registros de control del estándar y una breve descripción de cada uno.

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.

Un periódico con texto

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

- Nivel de Máquina:

El modo máquina puede acceder a todos los registros de estado y control de los niveles inferiores, que en este caso son todos menos el de debbbug.

misa -> registro en el que se describen las distintas extensiones e implementaciones que se han añadido sobre la máquina.

mvendrorid -> registro con el id del fabricante (machine vendor ID register).

Marchid -> contiene el tipo de microarquitectura implementada (machine architecture ID register).

mimpid -> contiene la versión de la implementación (machine implementation ID register).

mhartid -> información del hardware thread (Hart ID Register).

Registros de estado (mstatus y mstatush):

En el registro de control mstatus guardamos la información de estado del estado de operación actual. Desde este registro podemos habilitar también excepciones e interrupciones.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.

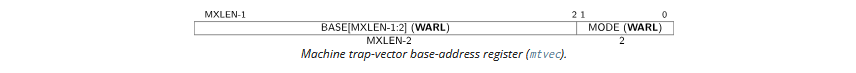
Dentro del registro de control mstatus los bits que habilitan las interrupciones son XIE, con X siendo el nivel de ejecución, en este caso MIE (Iterrupt-Enable), y si queremos interrupciones anidadas hay otros dos bits que continen información del anterior modo de ejecución junto al la anterior información de habilitación de interrupciones: xPIE (previous interrupt-enable), xPP (previous privilage).

Luego para retornar la rutina de tratamiento tenemos las instrucciones xRET, que retornarán al nivel de privilegios que se estaba ejecutando, viene dado por los bits xPP y xPIE. Fundamentalmente recuperan el estado anterior antes de haberse visto interrumpidas.

mtvec -> (machine trap vector base address register) este es el registro que almacena la dirección base sobre la que se puede vectorizar la rutina de tratamiento. Este registro tiene que estar siempre implementado para las excepciones. Sus dos primeros bits indican el modo de funcionamiento de este registro, puesto que la dirección de comienzo tendrá que estar siempre alineada a 4 bytes. Si este modo está puesto a cero entonces toma un enfoque directo donde la propia dirección alamacenada en el registro es la que se usa como comienzo de la rutina de tratamiento, si está puesto a 1, entonces activamos la vectorización donde se saltará a la dirección base más el producto de la causa de la interrupción por 4. Se puede ver en la siguiente tabla.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.



Por lo general las interrupciones y excepciones son tratadas en modod máquina, sin embargo, se puede platear el delegar estas a un nivel inferior. Para esto se implementan dos registros de control que permiten llevar esta tarea a cabo, como son el midelegate y el medelegate, nótese que las interrupciones y excepciones solo se podrán anidar de esta manera horizontalmente no verticlamente.

Registros de interrupción (mip y mie) -> mip (interruption pending), mie (interruption anable). En este caso el registro mip muestra si hay alguna interrupción pendiente de ser tratada, mientras que el registro mie puede habilitar distintos tipos de interrupciones, entre ellos el timer, y a partir de bit numero 16 son particularizables por el propietario. Se adjunta una imagen con los nombres que le da el estándar a cada uno de estos bits de este registro.

Diagrama

El contenido generado por IA puede ser incorrecto.

El bit número 11 (MEIE) es el que se usa para las interrupciones externas, es decir interrupciones generales. Sin embargo el bit numero 7 (MTIE) es para las interrupciones de un timer, si es que está implementado.

El bit 3 (MSIE) es para las interrupciones de tipo software, y si se tiene implementado el modo supervisor se pueden usar los bits 9 (SEIE), 5 (STIE) y 1 (SSIE).

Cuando se ha producido una excepción tenemos el registro mepc (machine execution program counter) que almacena la dirección de la instrucción que se estaba ejecutando (alineada) cuando se generó la excepcion.

Cuando se genera una excepcion o interrupción se almacena la causa de la excepcion o interrupción en el registro de control mcause (machine cause register).

Adjunto una imagen con las posibles con las cusas seleccionadas por el estándar.

Tabla

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Hay una tabla de prioridad de excepciones.

Hay un registro que contiene el valor asociado a la interrupción (trap), en este caso mtval (machine trap value register). Básicamente para ayudar en el manejo de la excepción o interrupción. También se puede usar con otros valores asociados, por ejemplo con los bits que fallan en una instrucción ilegal o, para las excepciones de paginación puede contener la dirección virtual que causó la excepción.

Registros de control mapeados a memoria: Para el estándar oficial solo se contemplan dos registros de control mapeados a memoria. Es este caso son aquellos referentes a la descripción propia del timer, estos son mtime y mtimecpy. Ambos son registros de 64 bits donde el primero es un registro que se actualiza e incremente al mismo tiempo que los propios ciclos de la máquina. Mientras que el segundo es un registro que tiene el valor referente al umbral de tiempo sobre el que el timer generará una interrupción.

Siendo estos los únicos q¡registros de control mapeados en memoria que entiende el propio estándar oficial. Para implementaciones como CLIC tenemos otra serie de registros mapeados en memoria.

Instrucciones privilegiadas: Ecall para llamadas al sistema desde el modo de ejecución en el que se encuentren. Suelen usar registros para el paso de parámetros. Ebreak para el debbuggin, genera un punto de ruptura que le devuelve el control al sistema de depuración. Instrucciones xRET para el retorno de rutina que rehabilitan las interrupciones y retornan al punto desde donde se generó la interrupción. WFI (wait for interrupt) es una instrucción que se usa cuando se espera que la máquina espere a que se genere una interrupción, se puede programar directamente como un nop. Reset para las situaciones que lo apremien, que pondrá todos los registros en un estado base, volviendo al modo máquina, necesario por ejemplo para el apagado de una máquina. Iterrupciones no enmascarables, solo para errores del hardware, a diferencia del reset se puede seguir usando la máquina por ejemplo para análisis.

- PLataform Level Instrucction Controler (PLIC):

Para prioridades en la entrada salida por interrupciones, añadiendo DMA.

Terminología:

Hart -> Hardware threads

ABI -> Aplication Binary Interface

OS -> Operating System

SEE -> Supervisor Execution Environment

AEE -> Aplication Execution Environment

SBI -> Supervisor Binary Interface

HEE -> Hypervisor Execution Environment

HBI -> Hypervisor Binary Interface

WARL -> Write Any / Read Legal

PMA -> Physical Memory Attributes

Este es un documento que compacta las especificaciones oficiales junto a la información más usual del estándar, y las deja en formato web que es más fácil de atajar. Por otro lado podemos encontrar varios ejemplos muy útiles que nos pueden servir para las diferentes implementeciones que se busquen hacer.

T