(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 301 309**

(21) Número de solicitud: 200502884

(51) Int. Cl.:
***G06F 12/14*** (2006.01)

(12) PATENTE DE INVENCIÓN CON EXAMEN PREVIO B2

(22) Fecha de presentación: **23.11.2005**

(43) Fecha de publicación de la solicitud: **16.06.2008**

Fecha de la concesión: **24.11.2008**

(45) Fecha de anuncio de la concesión: **16.12.2008**

(45) Fecha de publicación del folleto de la patente: **16.12.2008**

(73) Titular/es: **Universidad de Alcalá**
**Plaza de San Diego, s/n**
**28801 Alcalá de Henares, Madrid, ES**

(72) Inventor/es: **Sánchez Prieto, Sebastián y**
**Meziat Luna, Daniel**

(74) Agente: **No consta**

(54) Título: **Diseño de un mecanismo hardware que mejora el procedimiento de llamada al sistema operativo.**

(57) Resumen:
Diseño de un mecanismo hardware que mejora el procedimiento de llamada al sistema operativo.
Esta invención para la que se solicita la patente consiste en el diseño de un mecanismo hardware, que aplicado a los procesadores actuales que soportan un modo dual de ejecución de instrucciones (modo usuario y modo supervisor), reduce considerablemente los tiempos necesarios para conmutar entre ambos modos. Todo ello es aplicable siempre que el sistema operativo se proyecte en el espacio de direccionamiento virtual de cada proceso, cosa que suele ser habitual. Con este mecanismo, que se basa en el empleo de un bit adicional ubicado en las tablas de páginas o de segmentos, se consigue que una llamada al sistema se pueda implementar de un modo similar a las llamadas típicas a función con instrucciones del tipo call/ret, que en este caso denominaremos syscall/sysret, asegurando al mismo tiempo que al sistema operativo sólo se accede a través de puntos de entrada bien definidos.

ES 2 301 309 B2

## DESCRIPCIÓN

Diseño de un mecanismo hardware que mejora el procedimiento de llamada al sistema operativo.

### Sector de la técnica

La presente invención se refiere a los añadidos y modificaciones que pueden aplicarse a la arquitectura de un computador para mejorar considerablemente el procedimiento de llamadas al sistema empleado por las aplicaciones para acceder a los servicios ofrecidos por el sistema operativo o *kernel*.

### Estado de la técnica

Todas las arquitecturas de computadores actuales que soportan modo dual de ejecución de instrucciones (modo supervisor y modo usuario) y protección de memoria, basado generalmente en paginación o segmentación, establecen mecanismos específicos para que las aplicaciones puedan acceder a los servicios del sistema operativo. Estos mecanismos de conmutación de modo usuario a modo supervisor y viceversa sirven para garantizar la protección entre la aplicación que hace la llamada y el sistema operativo. De este modo, la aplicación sólo puede acceder a los servicios del sistema operativo a través de una interfaz bien definida que sirve para garantizar la integridad del sistema.

El mecanismo de llamada al sistema provoca que el procesador conmute a modo supervisor, se guarde el estado del proceso que realizó la llamada, se conmute de pila y se transfiera el control a una zona de memoria diferente del espacio de direccionamiento virtual de la aplicación, dicha zona pertenece al sistema operativo. Una vez que el servicio finaliza, se devuelve el control a la aplicación utilizando una instrucción específica que retorna el procesador nuevamente a modo usuario.

La realidad es que, aunque el mecanismo ofrecido por la arquitectura permita conmutar a un espacio de direccionamiento virtual no visible por la aplicación, la mayoría de los sistemas operativos (sino todos) no hacen uso de esta posibilidad y proyectan el propio sistema operativo dentro del espacio de direccionamiento de cada aplicación. Como ejemplo de lo comentado podemos citar los casos de UNIX, Windows, MacOSX o Linux. Este hecho es el que se puede aprovechar para mejorar considerablemente el mecanismo de llamada al sistema, utilizando la técnica objeto de la presente invención.

### Descripción de la invención

Basándose en el hecho de que la mayoría de los sistemas operativos proyectan el espacio de memoria utilizado por ellos mismos en el espacio de direccionamiento virtual de cada aplicación, podemos cambiar la instrucción específica de llamada al sistema por una instrucción similar a la empleada para llamar a un procedimiento o función (instrucción CALL) que tiene una sobrecarga mucho menor. También el procedimiento de retorno puede simplificarse utilizando una instrucción similar al RET empleado en procedimientos y funciones. De este modo, la invocación de un servicio del sistema operativo es prácticamente idéntico en su modo de operación a la invocación de una función en espacio de usuario.

Las diferencias entre la instrucción de llamada al sistema propuesta, que denominaremos SYSCALL, con la instrucción de llamada a procedimiento CALL, son básicamente tres. La primera es que la instrucción SYSCALL provoca una conmutación a modo supervisor y CALL no. La segunda es que la instrucción CALL permite llamar a cualquier posición de memoria dentro del código de usuario, mientras que la instrucción SYSCALL sólo permite la llamada a las posiciones de memoria del sistema operativo que sean marcadas como puntos de entrada al *kernel*. La tercera es que la dirección de retorno, en el caso de SYSCALL, debe almacenarse siempre en un registro del procesador y no en la pila, con objeto de evitar posibles manipulaciones de la misma por otros hilos de la aplicación. Un modo alternativo de evitar el problema de la pila planteado, consiste en emplear dos registros punteros de pila, uno en modo usuario y otro en modo supervisor como hacen determinadas arquitecturas. En este caso el puntero de pila de usuario se emplea cuando el procesador ejecuta código en modo usuario y el puntero de pila de supervisor se emplea cuando el procesador ejecuta código en modo supervisor.

Dado que las diferencias entre ambas instrucciones (CALL/SYSCALL) es mínima, la eficiencia de las mismas será similar y por lo tanto mucho mejor que la eficiencia de las actuales llamadas al sistema.

Para retornar de una llamada al sistema se propone el empleo de una instrucción SYSRET que será similar a la instrucción RET empleada para retornar de una función ordinaria. La diferencia entre ambas, es que SYSRET retorna el procesador desde modo supervisor a modo usuario.

El principal problema de implementación deriva de la definición de los puntos de entrada al *kernel*, cuyo objetivo será el de garantizar que la llamada desde modo usuario no se realiza a una dirección de memoria incontrolada. Para ello se propone añadir un bit adicional a la entrada de la tabla de páginas o de segmentos que determine si la página o segmento correspondiente contiene un punto de entrada al *kernel*. A dicho bit lo denominaremos "bit de punto de entrada al *kernel*" (PEK). Un intento de llamada al sistema a una dirección que no esté definida como un punto de entrada al *kernel* provocará un error de direccionamiento. Dentro de cada página o segmento sólo se permitirá, en

principio, un punto de entrada al *kernel* que sin pérdida de generalidad, podría ser la primera posición de memoria de la página o del segmento.

En ciertas arquitecturas, las RISC por ejemplo, si una página tiene en su entrada de la tabla de páginas su PEK activo, esta página podría contener múltiples puntos de entrada al *kernel*. Para ello bastaría con que el código de la página contuviese únicamente instrucciones de salto (de longitud fija en RISC) a múltiples posiciones dentro del núcleo. Dicho de otro modo, la página sólo debe contener código trampolín a distintas posiciones del núcleo.

**Breve descripción de los dibujos**

El primer dibujo (Figura 1) incluido muestra cómo se puede implementar el mecanismo descrito en un sistema paginado. En la parte izquierda aparece la tabla de páginas de un proceso con una entrada que contiene su PEK activo. Esa entrada contiene el número de marco donde se proyecta la página y que sus primeras posiciones contienen un punto de entrada al *kernel* válido.

El segundo dibujo (Figura 2) es similar al primero, pero en este caso, la página contiene código trampolín a múltiples posiciones del *kernel*. Este dibujo sólo es aplicable en arquitecturas donde las instrucciones tienen longitud fija, típicamente cuatro bytes, y están alineadas, como por ejemplo las arquitecturas RISC.

**Modo de realización**

La parte que supone una modificación de la implementación tradicional de una instrucción de llamada a función ordinaria CALL, respecto a la instrucción SYSCALL, es la de conmutar a modo supervisor y establecer los puntos de entrada al *kernel*. Por este motivo, es la parte que vamos a describir. La conmutación a modo supervisor por parte de la instrucción SYSCALL y el retorno a modo usuario por parte de la instrucción SYSRET es sencilla, basta con modificar el bit de la palabra de estado del procesador (PSW o *flags*) cuando se ejecuten estas dos instrucciones, la primera para activarlo y la segunda para desactivarlo.

Para establecer un punto de entrada al *kernel* hay que añadir un bit a cada entrada de la tabla de páginas o de segmentos, correspondientes a las páginas o segmentos donde se localice el punto de entrada.

Analicemos cómo se desarrolla una llamada al sistema en un sistema paginado (en un sistema segmentado sería similar) haciendo uso del bit PEK y del bit que determina que la página pertenece al *kernel* (este bit existe en las arquitecturas que soportan paginación y modo dual de ejecución de instrucciones). Supondremos que la llamada se realiza con una instrucción del tipo SYSCALL *_ADDRESS_*. Los pasos llevados a cabo por la arquitectura al ejecutar esta instrucción deben de ser los siguientes:

1. Se verifica que la dirección de memoria *_ADDRESS_* pertenece al espacio de direccionamiento del *kernel*, consultando para ello el correspondiente bit de la tabla de páginas. Si no es así, se producirá un *trap*.

2. Se comprueba que la página referenciada en la llamada contiene su correspondiente bit PEK de la tabla de páginas activado. Si no es así, se producirá un *trap* porque la página no contiene ningún punto de entrada al *kernel*.

3. Se analiza si la dirección de llamada *_ADDRESS_* corresponde a la primera posición de la página que actúa como punto de entrada. Como alternativa también se podría obligar a que las llamadas al sistema operativo estuvieran siempre alineadas a página, bastaría con poner a cero todos los bits de la dirección correspondientes al desplazamiento dentro de la página. Sea como fuere, o bien se produce un *trap* si la dirección de llamada no está alineada, o se obliga a la alineación explícitamente por hardware.

4. Se guarda la dirección de retorno en un registro. Si se emplea un sistema con pila de usuario y de núcleo, la dirección de retorno se almacena en la pila del núcleo.

5. Se conmuta a modo supervisor.

6. Se ejecuta el servicio de la llamada.

7. Se finaliza con la instrucción SYSRET que devuelve el control a modo usuario y se continúa con la ejecución en el punto determinado por la dirección de retorno guardada en el paso 4.

**Aplicación industrial**

La presente invención es susceptible de ser empleada, introduciendo pequeñas modificaciones, en todos los procesadores que soportan segmentación o paginación y un modo dual de ejecución de instrucciones. Para que la invención pueda aplicarse, el *kernel* debe proyectarse en el espacio de direccionamiento de las aplicaciones. Las modificaciones necesarias consisten en el empleo de dos instrucciones específicas, SYSCALL y SYSRET, para realizar y retornar de la llamada respectivamente y un bit adicional en la tabla de páginas o de segmentos.

## REIVINDICACIONES

1. Diseño de un mecanismo hardware que mejora el procedimiento de llamada al sistema operativo cuando se proyecta el *kernel* en el espacio de direccionamiento virtual de cada aplicación. Se introducen para ello dos nuevas instrucciones (o se modifica la semántica de las instrucciones existentes) para invocar y retornar respectivamente de las funciones ofrecidas por el sistema operativo. Se introduce un bit adicional en cada entrada de la tabla de páginas para establecer puntos de entrada al *kernel* (PEK) que serán utilizados por las nuevas instrucciones, garantizando la protección en el acceso al sistema operativo.

2. Mecanismo que según la reivindicación 1 se **caracteriza** por utilizar dos nuevas instrucciones denominadas SYSCALL y SYSRET para invocar servicios del sistema operativo y retornar de los mismos respectivamente de modo seguro, SYSCALL recibe como argumento la dirección de entrada al kernel y comprueba que la entrada de la tabla de páginas o de segmentos correspondiente a la dirección de entrada tiene activo el PEK. Si es así, permite el acceso a la llamada y guarda la dirección de retorno en la pila, si no, provoca un error de direccionamiento, SYSRET retorna de la llamada recuperando la dirección almacenada previamente en la pila por SYSCALL.

3. Mecanismo que según la reivindicación 2 introduce un bit (modificación hardware) en cada entrada de la tabla de páginas o de segmentos para la implementación de la semántica de la instrucción SYSCALL.

4. Mecanismo que según las reivindicaciones 2 y 3 se **caracteriza** por el empleo de una instrucción SYSCALL, que hace uso del bit PEK, para garantizar la entrada al sistema operativo a través de puntos bien definidos. Esos puntos de entrada se obtiene a partir de la dirección pasada como argumento a SYSCALL, bien alineando la dirección al comienzo de la página (véase Figura 1) o bien directamente a partir de esa dirección (véase Figura 2). En el caso de la figura 1, la dirección efectiva de entrada se obtiene a partir de la dirección pasada como argumento a SYSCALL poniendo a ceros los bits correspondientes al desplazamiento de página y dejando los bits de mayor peso sin modificar. En el caso de la figura 2 la dirección de entrada contiene el código trampolín para saltar a la dirección efectiva del kernel.

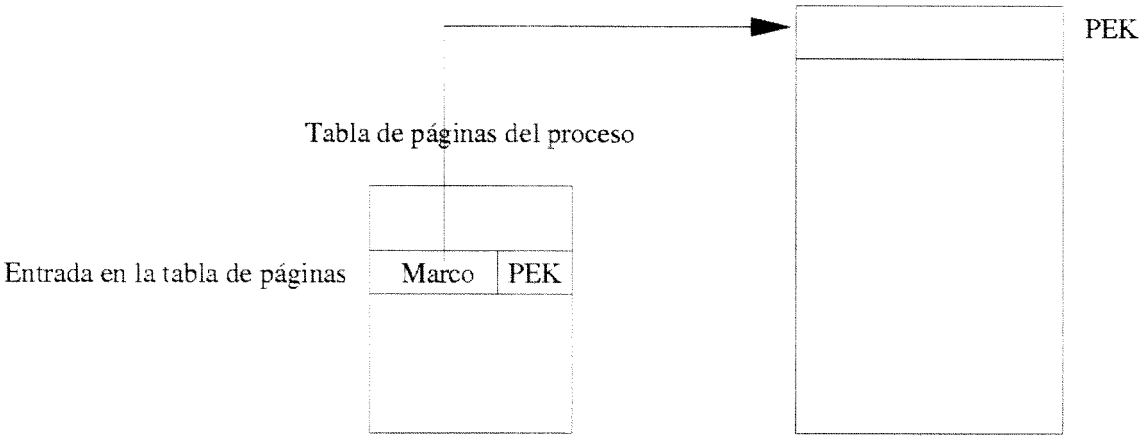


**Figura 1**

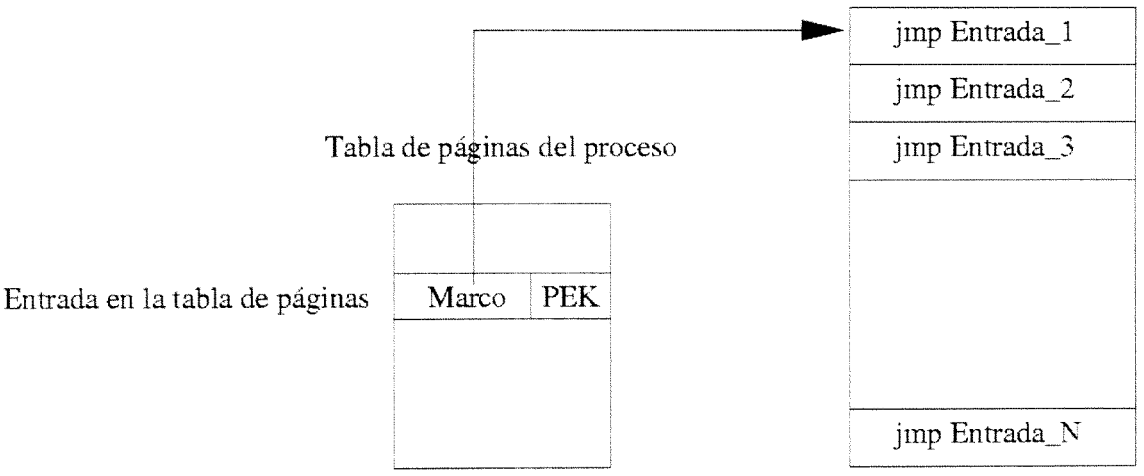


**Figura 2**

OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) ES 2 301 309

(21) Nº de solicitud: 200502884

(22) Fecha de presentación de la solicitud: **23.11.2005**

(32) Fecha de prioridad:

# INFORME SOBRE EL ESTADO DE LA TÉCNICA

(51) **Int. Cl.:** ***G06F 12/14*** (2006.01)

## DOCUMENTOS RELEVANTES

| Categoría | Documentos citados | Reivindicaciones afectadas |
|---|---|---|
| A | AMD, "SYSCALL and SYSRET Instruction Specification", Application Note, páginas 1-10. 20.09.2004 [recuperado el 16.05.2008]. Recuperado de Internet: <URL:http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/21086.pdf>; <URL:http//web.archive.org/web/*/http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/21086.pdf> | 1-4 |
| A | Intel, "IA-32 Intel Architecture Software Developer's Manual, Vol. 2 Instruction Set Reference", páginas 3-752-3-758. 07.12.2004 [recuperado el 16.05.2008]. Recuperado de Internet: <URL:http://homepages.cae.wisc.edu/~ece734/mmx/24547106.pdf>; <URL:http://web.archive.org/web/*/http://homepages.cae.wisc.edu/~ece734/mmx/24547106.pdf> | 1-4 |
| A | US 6968446 B1 (MCGRATH KEVIN J) 22.11.2005, columna 10, línea 37 - columna 14, línea 48; columna 15, línea 58 - columna 16, línea 62; figuras 1,5-6,8,9. | 1-4 |

**Categoría de los documentos citados**

X: de particular relevancia
Y: de particular relevancia combinado con otro/s de la misma categoría
A: refleja el estado de la técnica
O: referido a divulgación no escrita
P: publicado entre la fecha de prioridad y la de presentación de la solicitud
E: documento anterior, pero publicado después de la fecha de presentación de la solicitud

**El presente informe ha sido realizado**

☒ para todas las reivindicaciones ☐ para las reivindicaciones nº:

| Fecha de realización del informe | Examinador | Página |
|---|---|---|
| 19.05.2008 | Mª J. Lloris Meseguer | 1/1 |