(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 357 308**

(51) Int. Cl.:
G06F 9/38 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **06743880 .4**
(96) Fecha de presentación : **05.05.2006**
(97) Número de publicación de la solicitud: **1880276**
(97) Fecha de publicación de la solicitud: **23.01.2008**

(54) Título: **Almacenamiento en memoria caché en el procesado de datos.**

(30) Prioridad: **09.05.2005 GB 0509420**

(45) Fecha de publicación de la mención BOPI:
25.04.2011

(45) Fecha de la publicación del folleto de la patente:
25.04.2011

(73) Titular/es:
SONY COMPUTER ENTERTAINMENT Inc.
2-6-21, Minami-Aoyama, Minato-ku
Tokyo 107-0062, JP

(72) Inventor/es: **Ezra, Rabin**

(74) Agente: **Curell Aguilá, Marcelino**

DESCRIPCIÓN

La presente invención se refiere al almacenamiento en memoria caché en el procesado de datos.

A continuación se describirá un problema a título de ejemplo con respecto a la emulación de un microprocesador particular, aunque se observará que la invención presenta una aplicabilidad mucho más amplia.

Los núcleos de microprocesador se usan en varias aplicaciones tales como terminales particulares del televisor y el sistema de entretenimiento informático familiar Playstation 2™ (PS2) de Sony[RTM]. En el procesador de entrada/salida (IOP) de la PS2, el núcleo está provisto de 2 Megabytes de memoria principal y una memoria caché muy pequeña. Hace uso de la denominada “escritura directa” de la memoria caché, en la que cualquier información escrita por el procesador en una posición de memoria caché se escribe también en la memoria principal subyacente. Esto significa que la información nueva se escribe en la memoria caché en el caso de que sea necesario leerla de nuevo en un futuro próximo, pero la propia operación de escritura no se almacena en memoria caché ya que sigue siendo necesario cada vez un acceso a la memoria principal. Una ventaja de esta disposición es que permite implementar más fácilmente un código de programa automodificable. La técnica anterior es el documento US 2002/0010837, que da a conocer un sistema y un método de memoria caché para controlar esta última.

Se ha propuesto que el IOP se emule mediante un procesador de emulación que tiene una memoria interna demasiado pequeña para proporcionar los 2 MB de la memoria del IOP. Se puede acceder a una memoria externa, aunque esto se realiza únicamente a través de un controlador de DMA. Para permitir que la emulación funcione a una velocidad útil, se requiere por lo tanto una estrategia de almacenamiento en memoria caché, ya que los accesos a una memoria externa en el sistema de emulación que usa el controlador de DMA son lentos. Para reducir el número de accesos a la memoria externa que son necesarios, la estrategia de almacenamiento en memoria caché debería incluir el almacenamiento en memoria caché de escrituras de datos así como de lecturas de datos. No obstante, esto significaría que el código automodificable no se puede emular fácilmente.

La presente invención proporciona un procesador de datos según la reivindicación 1 adjunta a la presente memoria.

La invención proporciona una forma eficaz de acceder a datos e instrucciones al mismo tiempo que se reduce la necesidad de acceder a una memoria principal.

Vista desde un segundo aspecto, la presente invención proporciona un método de procesado de datos según la reivindicación 10 adjunta a la presente memoria.

Otros aspectos de la invención incluyen software de ordenador que comprende un código de programa para llevar a cabo este método; y un soporte (por ejemplo, un medio de transmisión o un soporte de almacenamiento) por medio del cual se proporciona dicho código de programa.

En las reivindicaciones adjuntas se definen otros diversos aspectos y características de la invención. A continuación se describirán formas de realización de la invención, únicamente a título de ejemplo, haciendo referencia a los dibujos adjuntos, en los que:

la Figura 1 ilustra esquemáticamente un sistema de procesado de datos;

la Figura 2 ilustra esquemáticamente un sistema de procesado de datos que usa memorias caché de datos y de instrucciones;

la Figura 3 es un diagrama de flujo esquemático referente a una operación para leer una instrucción;

la Figura 4 es un diagrama de flujo esquemático referente a una operación para escribir un valor de datos; y

la Figura 5 es un diagrama de flujo esquemático referente a una operación para leer un valor de datos.

La Figura 1 ilustra esquemáticamente un sistema de procesado de datos del que se va a realizar una emulación. El sistema comprende un procesador 10 que lee datos e instrucciones desde, y escribe datos e instrucciones modificadas en, una memoria principal 20.

La siguiente descripción se refiere a una técnica para la emulación del funcionamiento del sistema de la Figura 1 que usa un procesador cuya memoria local es demasiado pequeña para contener una imagen de la memoria principal 20 del sistema del que se va a realizar la emulación. Debido a esta restricción, se debe utilizar una estrategia de memoria caché.

La Figura 2 ilustra esquemáticamente la disposición de emulación. Las técnicas de emulación son en general bien conocidas, y, por claridad, se omiten las características que no son relevantes directamente para la presente forma de realización. La emulación conlleva un procesador de emulación que ejecuta un software de emulación escrito en un lenguaje nativo para el procesador de emulación, de manera que se ejecuta un grupo de dichas instrucciones nativas para emular el tratamiento de una instrucción en el sistema emulado. En la descripción que se ofrece a continuación, el

término “instrucción” se referirá a una instrucción en el sistema emulado, y no a una instrucción nativa del software de emulación.

Haciendo referencia a la Figura 2, un procesador 110 que ejecuta un software 120 de emulación accede a una memoria principal 130 a través de una memoria caché (I) 140 de instrucciones y una memoria caché (D) 150 de datos. La razón de usar la memoria caché I y la memoria caché D es que la memoria local para el procesador 110 es demasiado pequeña para contener una imagen de la memoria principal 20 del sistema emulado, y se debe acceder a la memoria principal 130 asociada al procesador 110 a través de un acceso DMA caro (es decir, que consume mucho tiempo).

La memoria caché I 140 es de correspondencia directa para obtener más velocidad de acceso y contiene 8 páginas de memoria de 4 kilobytes cada una de ellas (es decir, cada página contiene diversas líneas de memoria caché). En esta forma de realización se usa un número reducido de páginas de memoria de gran tamaño para conseguir que el proceso de comprobación de un acierto de memoria caché resulte más eficaz. Las páginas de memoria de gran tamaño amortizan los accesos lentos a la memoria. Se pueden leer páginas de memoria desde la memoria principal 130 hacia la memoria caché I 140, y el procesador puede leer instrucciones de la memoria caché I 140. No obstante, los valores almacenados en la memoria caché I 140 no se vuelven a escribir nunca en la memoria principal 130.

Las transferencias hacia y desde memorias caché se realizan página a página. De modo similar, la búsqueda de una memoria caché, para detectar si la misma contiene un elemento de datos requerido, se lleva a cabo detectando si la página que contiene ese elemento está contenida en la memoria caché.

La memoria caché D 150 es totalmente asociativa para reducir la denominada “hiperpaginación” – es decir, un cambio rápido de las páginas de la memoria caché – y contiene nuevamente 8 páginas de 4 kilobytes cada una de ellas. Cuando se va a leer una página nueva hacia la memoria caché D desde la memoria principal 130, una página a la que se ha accedido menos recientemente, almacenada en la memoria caché D, se vuelve a escribir en la memoria principal (si la misma se ha cambiado desde que se leyó de la memoria principal). Por lo tanto, si el procesado modifica cualquier dato almacenado en la memoria caché D, la modificación queda contenida en la memoria caché D 150 hasta que esa página se vuelva a escribir en la memoria principal 130.

A continuación se describirá la interacción de la memoria caché D y la memoria caché I haciendo referencia a diagramas de flujo mostrados en las figuras 3 a 5.

La Figura 3 es un diagrama de flujo esquemático referente a una operación para leer una instrucción.

En una etapa 200, el procesador 110 intenta acceder a la instrucción requerida desde la memoria caché I 140. Si la instrucción requerida está presente en la memoria caché I 140, el control se traslada a una etapa 210 en la que se lee la instrucción desde la memoria caché I y la misma se traslada al procesador 110 para su tratamiento según la manera habitual. A continuación, el proceso finaliza.

No obstante, si la instrucción requerida no estaba en la memoria caché I 140, se realiza a continuación una comprobación en relación con si la instrucción requerida es parte de la información almacenada en la memoria caché D 150. Esta prueba se representa mediante una etapa 220. Si la instrucción requerida está realmente en la memoria caché D, entonces la página completa se copia desde la memoria caché D a la memoria caché I en una etapa 230. Obsérvese que esto simplemente puede sobrescribir una página en la memoria caché I, ya que los datos de la memoria caché I no se vuelven a escribir nunca en la memoria principal 130. Desde la etapa 230, el control se traslada nuevamente a la etapa 210, en la que la instrucción requerida se lee de la memoria caché I y el procesado finaliza.

No obstante, si la instrucción requerida no está ni en la memoria caché I (etapa 200) ni en la memoria caché D (etapa 220), entonces, en una etapa 240, la página que contiene la instrucción requerida se lee desde la memoria principal hacia la memoria caché I 140, sobrescribiendo una página de la memoria caché I. El control se traslada nuevamente a la etapa 210, y el proceso finaliza.

La Figura 4 es un diagrama de flujo esquemático referente a una operación para escribir un valor de datos.

En una etapa 310, el procesador 110 escribe un valor de datos en la memoria caché D 150. Tal como se ha descrito anteriormente, esto se usará finalmente para actualizar la memoria principal 130, aunque puede que ello no ocurra hasta que la página pertinente se deba sobrescribir en la memoria caché D.

En una etapa 320, se realiza una detección en relación con si la página que contiene el valor de datos recién escrito está también contenida en la memoria caché I. En caso afirmativo, entonces, en una etapa 330, el nuevo valor de datos se escribe también en la posición pertinente de la memoria caché I, y el proceso finaliza. No obstante, si la página pertinente no estaba contenida en la memoria caché I, el proceso simplemente finalizaría en ese momento.

La Figura 5 es un diagrama de flujo esquemático referente a una operación para leer un valor de datos. En una etapa 400, el procesador 110 intenta acceder al valor de datos desde la memoria caché D 150. Si la dirección requerida está almacenada en memoria caché en la D 150, entonces el valor requerido se lee de la memoria caché D en una etapa 410, y el proceso finaliza. No obstante, si la página necesaria no está en la memoria caché D, entonces, en una

etapa 420, la página menos usada recientemente se escribe de nuevo (si fuera necesario, es decir, si la misma ha sido modificada) desde la memoria caché D hacia la memoria principal 130, y, en una etapa 430, la página que contiene la dirección de memoria requerida se lee desde la memoria principal 130 hacia la memoria caché D 150. A continuación, el control se traslada nuevamente a la etapa 410, y el proceso finaliza.

Usando la estrategia anterior, aun cuando la información almacenada en la memoria caché I 140 no se escribe de nuevo nunca directamente en la memoria principal 130, y en la presente forma de realización no se proporciona ninguna instrucción o función de invalidación de la memoria caché I, si el procesador 110 ejecuta el código denominado automodificable de manera que instrucciones almacenadas en la memoria principal 130 se van a sobrescribir con instrucciones generadas por medio de la ejecución de las propias instrucciones, no existe ninguna posibilidad de que se produzca una inconsistencia en el código a ejecutar. Cuando se va a sobrescribir una instrucción, el valor nuevo se escribe en la memoria caché D y, si esa página está almacenada también en la memoria caché I, se escribe en la memoria caché I. Por lo tanto, la memoria caché I se mantiene actualizada con todos los cambios. En el momento en el que se va a acceder a una instrucción, si la misma no está en la memoria caché I, se busca en la memoria caché D antes que la memoria principal. Por lo tanto, cualquier cambio que no se haya escrito de nuevo todavía en la memoria principal 130 seguirá estando en la memoria caché D, y las instrucciones modificadas se transferirán desde la memoria caché D a la memoria caché I.

REIVINDICACIONES

1. Procesador de datos que comprende:

una memoria principal (130);

una memoria caché (140) de instrucciones y una memoria caché (150) de datos dispuestas, cada una de ellas, como una pluralidad de páginas de memoria caché;

una lógica (110) de recuperación de instrucciones que se puede hacer funcionar para buscar en la memoria caché (140) de instrucciones una instrucción requerida; y si la instrucción requerida no está presente en la memoria caché (140) de instrucciones, para buscar en la memoria caché (150) de datos; si la instrucción requerida está presente en la memoria caché de datos para recuperar la página que comprende la instrucción requerida desde la memoria caché de datos hacia la memoria caché de instrucciones, y si la instrucción requerida no está presente en la memoria caché (150) de datos, para recuperar la página que comprende la instrucción requerida desde la memoria principal (130) hacia la memoria caché (140) de instrucciones;

una lógica (110) de escritura de datos que se puede hacer funcionar para escribir un valor de datos en la página pertinente de la memoria caché (150) de datos en una dirección de datos y, si la misma página está representada también en la memoria caché (140) de instrucciones, para escribir ese valor de datos en la dirección correspondiente de esa página en la memoria caché (140) de instrucciones; y

una lógica (110) de control de memorias caché que se puede hacer funcionar para transferir páginas desde la memoria caché (150) de datos hacia la memoria principal (130),

en el que valores de datos almacenados en la memoria caché (140) de instrucciones no se vuelven a escribir nunca en la memoria principal.

2. Procesador de datos según la reivindicación 1, en el que la memoria caché de instrucciones es de correspondencia directa y la memoria caché de datos es totalmente asociativa.

3. Procesador de datos según la reivindicación 1 ó 2, en el que transferencias hacia o desde una memoria caché se llevan a cabo basándose en un esquema de página a página.

4. Procesador de datos según la reivindicación 3, en el que la lógica de recuperación de instrucciones se puede hacer funcionar para buscar en la memoria caché de instrucciones y en la memoria caché de datos detectando si una página requerida está contenida en la memoria caché respectiva.

5. Procesador de datos según la reivindicación 3 ó 4, en el que la lógica de control de memorias caché está dispuesta para transferir una página de datos a la que menos se ha accedido recientemente, desde la memoria caché de datos hacia la memoria principal cuando se va a escribir una página recién requerida en la memoria caché de datos y esa página a la que menos se ha accedido recientemente contiene datos que han sido modificados.

6. Procesador de datos según cualquiera de las reivindicaciones anteriores, en el que cada una de las memorias caché contiene 32 kilobytes dispuestos en forma de 8 páginas de 4 kilobytes cada una de ellas.

7. Procesador de datos según cualquiera de las reivindicaciones anteriores, que comprende una lógica de ejecución de instrucciones para ejecutar instrucciones.

8. Procesador de datos según la reivindicación 7, en el que la lógica de ejecución de instrucciones comprende una disposición de procesado de datos que ejecutan software para la emulación de la ejecución de las instrucciones.

9. Procesador de datos según cualquiera de las reivindicaciones anteriores, no proporcionando el procesador de datos una función de invalidación de entradas en la memoria caché de instrucciones.

10. Método de procesado de datos en un sistema que tiene una memoria principal (130), y una memoria caché (140) de instrucciones y una memoria caché (150) de datos dispuestas, cada una de ellas, como una pluralidad de páginas de memoria caché;

comprendiendo el método las etapas siguientes:

buscar (200) en la memoria caché de instrucciones una instrucción requerida;

si la instrucción requerida no está presente en la memoria caché (140) de instrucciones, buscar (220) en la memoria caché (150) de datos; si la instrucción requerida está presente en la memoria caché de datos, recuperar la página que comprende la instrucción requerida desde la memoria caché de datos hacia la memoria caché de instrucciones;

si la instrucción requerida no está presente en la memoria caché de datos, recuperar (240) la página que comprende la instrucción requerida desde la memoria principal (130) hacia la memoria caché (140) de instrucciones;

cuando se va a escribir un valor de datos, escribir (310) ese valor de datos en la página pertinente de la memoria caché (150) de datos en una dirección de datos y, si la misma página está representada también en la memoria caché (140) de instrucciones, escribir (330) ese valor de datos en la dirección correspondiente de esa página en la memoria caché (140) de instrucciones; y

transferir páginas desde la memoria caché (150) de datos hacia la memoria principal (130),

en el que valores de datos almacenados en la memoria caché (140) de instrucciones no se vuelven a escribir nunca en la memoria principal.

11. Software de ordenador que comprende código de programa para llevar a cabo un método según la reivindicación 10.

12. Soporte mediante el cual se proporciona código de programa según la reivindicación 11.

13. Soporte según la reivindicación 12, siendo el soporte un medio de transmisión.

14. Soporte según la reivindicación 13, siendo el soporte un soporte de almacenamiento.

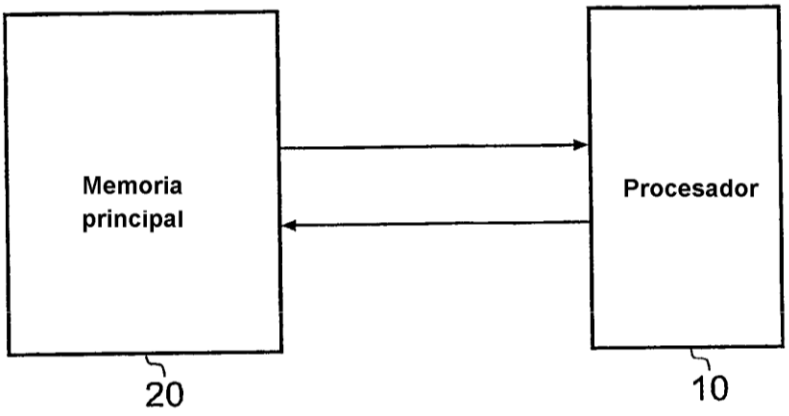
Memoria
principal
Procesador
20
10

Fig. 1

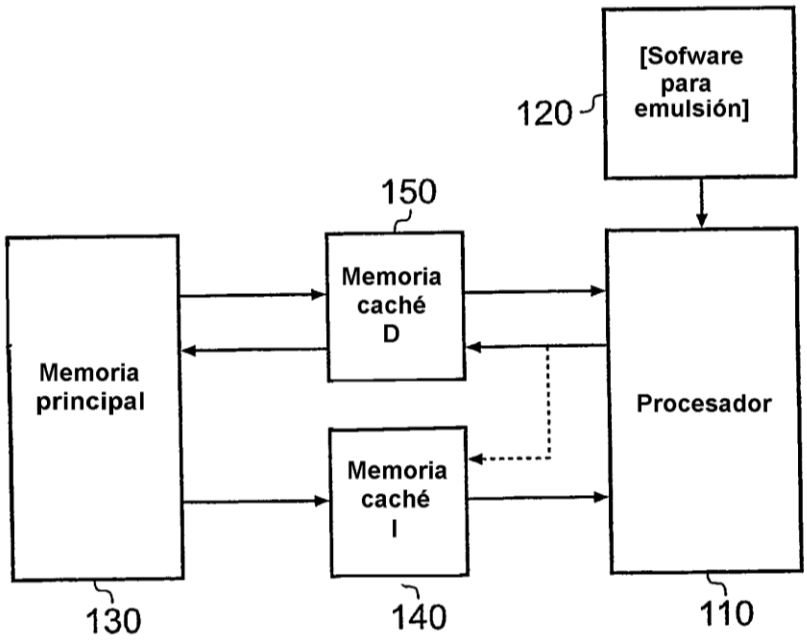
[Sofware
para
emulsión]
120
150
Memoria
caché
D
Memoria
principal
Procesador
Memoria
caché
I
130
140
110

Fig. 2

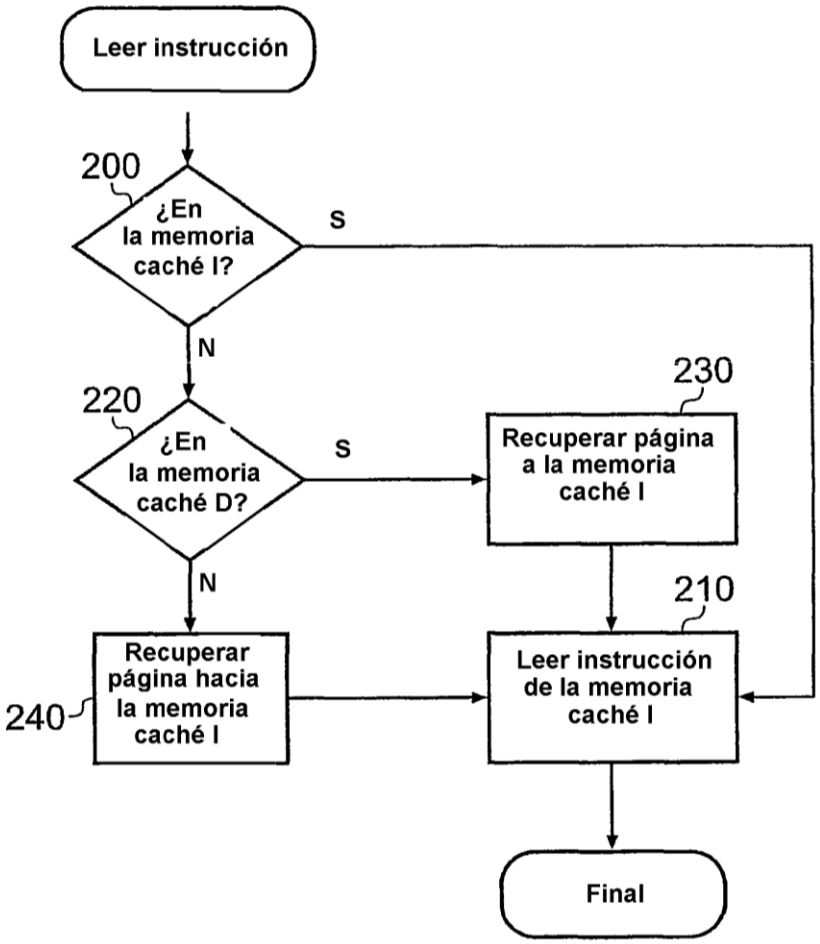
Leer instrucción
200
¿En la memoria caché I?
S
N
220
¿En la memoria caché D?
S
230
Recuperar página a la memoria caché I
N
210
240
Recuperar página hacia la memoria caché I
Leer instrucción de la memoria caché I
Final

Fig. 3

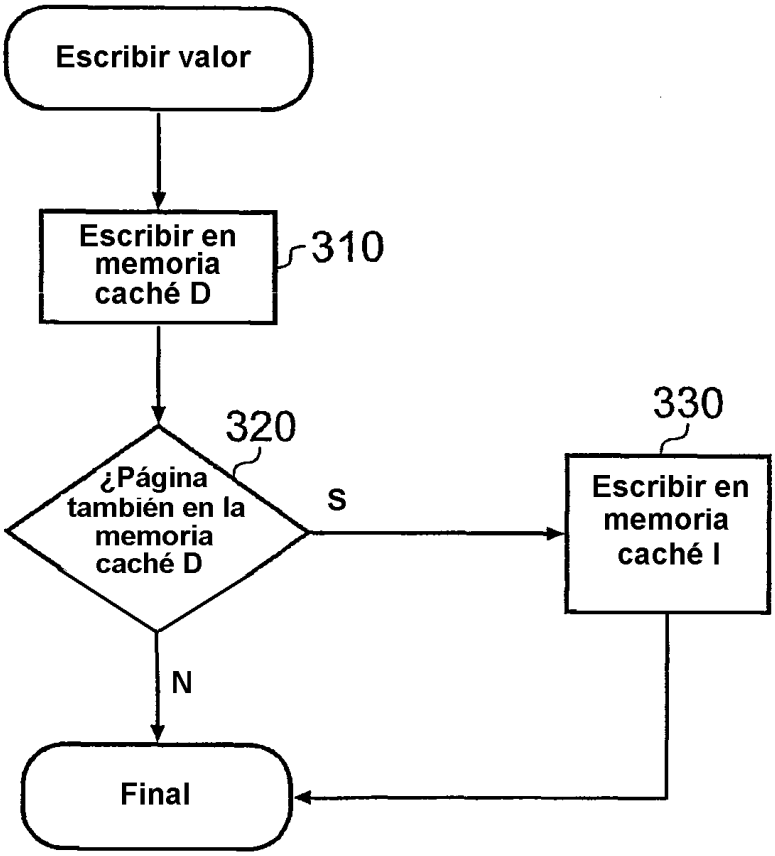
Escribir valor
Escribir en memoria caché D
310
320
¿Página también en la memoria caché D
S
330
Escribir en memoria caché I
N
Final

Fig. 4

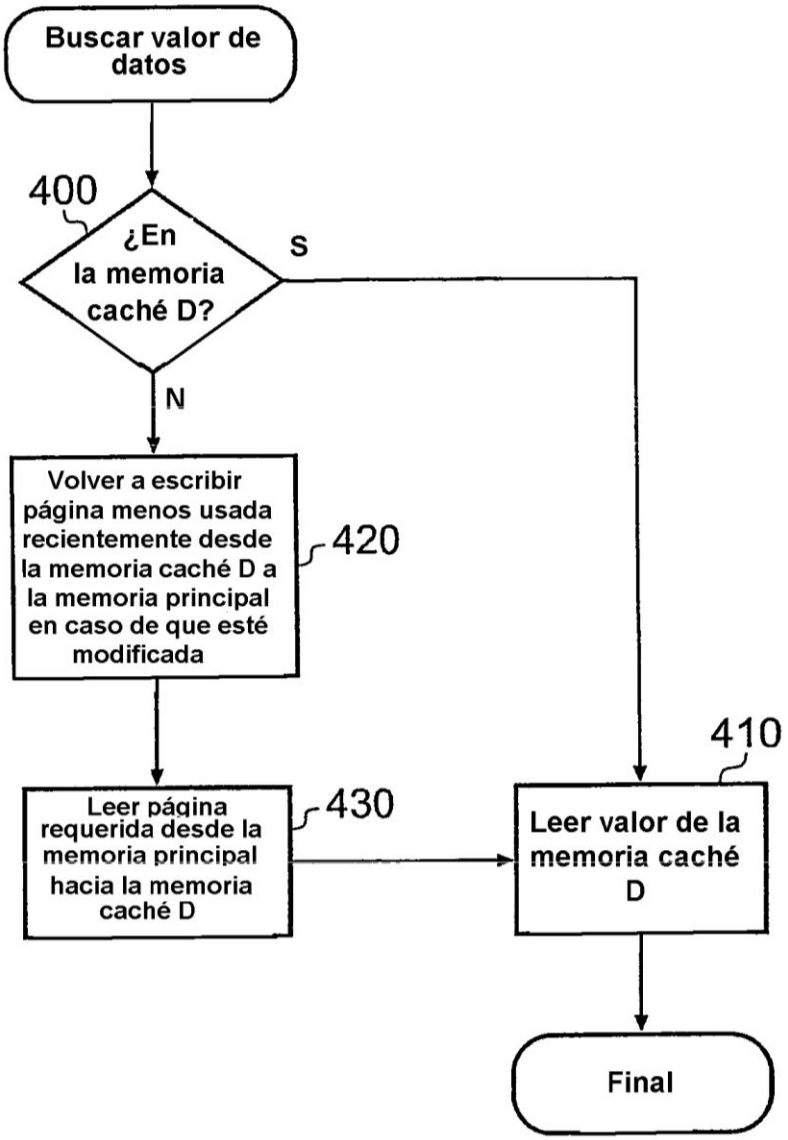
Buscar valor de datos
400
¿En la memoria caché D?
S
N
Volver a escribir página menos usada recientemente desde la memoria caché D a la memoria principal en caso de que esté modificada
420
410
Leer página requerida desde la memoria principal hacia la memoria caché D
430
Leer valor de la memoria caché D
Final

Fig. 5