(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 394 154**

(51) Int. CI.:

G06F 9/445 (2006.01)
G06F 12/10 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **23.07.2009 E 09780961 (0)**

(97) Fecha y número de publicación de la solicitud europea: **11.05.2011 EP 2318933**

(54) Título: **Método para actualización de datos en memorias utilizando una unidad de gestión de memoria**

(30) Prioridad:

24.07.2008 EP 08161092

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
22.01.2013

(73) Titular/es:

NAGRAVISION S.A. (100.0%)
Route de Genève 22-24
1033 Cheseaux-sur-Lausanne, CH

(72) Inventor/es:

GREMAUD, FABIEN y
KUDELSKI, HENRI

(74) Agente/Representante:

TOMAS GIL, Tesifonte Enrique

DESCRIPCIÓN

Método para actualización de datos en memorias utilizando una unidad de gestión de memoria

Campo de la invención

[0001] La presente invención se refiere a dispositivos que comprenden uno o más microprocesadores que realizan varias tareas gracias a, al menos, un código de la máquina incluido un programa de software y datos almacenados en memorias. Más particularmente, el método de la invención se dirige a actualizaciones completas o parciales de realización del programa de software utilizando una unidad de gestión de memoria asociada al microprocesador.

Antecedentes técnicos

[0002] Un microprocesador controla la memoria con un sistema de gestión de memoria que es comúnmente implementado en forma de hardware y software. La parte de hardware del sistema forma una unidad de gestión de memoria (MMU) que se puede incluir en el microprocesador o estar en la forma de un componente separado, bien integrado en el mismo chip como el microprocesador, o formando un circuito integrado diferente conectado al microprocesador y la memoria.

[0003] La unidad de gestión de memoria (MMU) ejecuta transacciones de memoria que incluyen funciones para traducir direcciones virtuales en direcciones físicas, funciones para la protección de la memoria y para el control de las memorias caché.

[0004] Los microprocesadores actualmente usados principalmente sostienen la noción de memoria virtual. En un sistema de memoria virtual, instrucciones de programa ejecutadas por el microprocesador se refieren a direcciones virtuales de uso de datos en un espacio de memoria virtual del microprocesador. Este espacio puede ser mucho más grande que el espacio de memoria física disponible en el sistema.

[0005] Las direcciones virtuales del microprocesador se traducen por una unidad de gestión de memoria (MMU) en direcciones físicas usadas para el acceso a la memoria de sistema o para otros dispositivos de almacenamiento. La memoria virtual usa un mecanismo de paginación para traducir o vincular las direcciones virtuales con direcciones físicas. El espacio de memoria física se divide en páginas físicas con un tamaño fijo, por ejemplo 4 kilobytes. Las direcciones virtuales comprenden una parte de dirección de página y una parte de desvío de página o impresión por transferencia de página. La dirección de página virtual se traduce por el mecanismo de paginación en la dirección de página física. La impresión por transferencia de página especifica un desvío físico en la página física, es decir un desvío de la dirección de la página física.

[0006] La paginación de la memoria permite la ejecución del programa con un espacio de memoria virtual más grande que el espacio de memoria física existente. Por otra parte, la paginación facilita el movimiento de programa en diferentes lugares de la memoria física durante ejecuciones diferentes o múltiples del programa. La paginación también permite la ejecución simultánea de procesos múltiples por el microprocesador, cada proceso pudiendo acceder a sus propias páginas de memoria física asignadas sin estar recogido del disco duro por ejemplo y sin necesidad de asignar toda la memoria física a sólo un proceso. La protección de la memoria de otros procesos puede también ser facilitada basándose en la paginación.

[0007] La traducción de la dirección de página virtual en la dirección de página física se realiza consecutivamente leyendo una tabla de página llamada operación de mapa de tablas de páginas o mapa de tablas de páginas. El software operativo del microprocesador mantiene y almacena en la memoria de sistema, tablas de página con información para traducir una dirección de página virtual en una dirección de página física. Por lo tanto es relativamente oneroso llevar a cabo un mapa de tablas de páginas cuando múltiples accesos de memoria deben ser realizados para la traducción. El mapa de tablas de páginas se realiza por el hardware, el software o una combinación de ambos.

[0008] Para mejorar los rendimientos del sistema reduciendo el número de mapas de tabla de página, muchos microprocesadores se benefician de un mecanismo de almacenamiento de información de tablas de páginas en una memoria caché. Esta información comprende direcciones de página física traducidas de direcciones de páginas virtuales recientemente usadas. Esta memoria caché de información de página es comúnmente conocida como una "memoria tampón de traducción de direcciones de página" o "Translation Lookaside Buffer" o "TLB cache". La dirección de página virtual está provista al TLB caché que busca esta dirección. Cuando esta dirección es encontrada, el TLB caché proporciona directamente la dirección de la página física correspondiente sin realizar el mapa de tabla de página para la traducción. De esta manera, los rendimientos del sistema al igual que la velocidad de ejecución de los procesos son mejorados gracias a esta conversión de direcciones directas.

[0009] Cuando una actualización completa o parcial de una pieza almacenada de software debe ser realizada de una manera discontinua en diferentes tipos de memoria tal como ROM (*Read Only Memory*, memoria de sólo lectura), NVM (*Non-Volatile Memory*, memoria no volátil), RAM (*Random Access Memory*, memoria de acceso aleatorio), etc., la

unidad de gestión de memoria y el mecanismo de paginación facilitan en gran medida las operaciones.

[0010] La actualización de un sistema puede llevarse a cabo según una primera solución que consiste en almacenar el nuevo software en una memoria física no volátil libre y modificando la tabla de página. Así el procesador usará las direcciones virtuales que apuntan hacia las direcciones físicas de la memoria donde el nuevo software es almacenado. Cuando la aplicación de instalación de actualización no tiene que ser actualizada, una aplicación específica fija ejecuta la actualización del software relacionado según un mecanismo relativamente simple. El sistema trabajará con el software viejo mientras la actualización no esté finalizada y se conectará al nuevo software sólo cuando éste esté completamente instalado y controlado por una secuencia de control tal como una suma de control o un hash.

[0011] Cuando una única aplicación también incluyendo la aplicación de instalación de actualización debe ser renovada en un sistema, la aplicación pasa a través de un estado inestable durante el proceso de modificación. La seguridad de tal aplicación bajo modificación depende de la fiabilidad de las instrucciones u órdenes usadas para la actualización. Esta situación de vulnerabilidad de la aplicación no puede ser aceptable en el campo de las tarjetas chip o módulos de seguridad que requieren un alto nivel de seguridad. Por otra parte, si la operación de actualización falla o se interrumpe de forma inesperada, un mecanismo bastante complejo para devolver el sistema al estado precedente es requerido. Esta restauración además implica un paso del sistema a través de un estado inestable que compromete la seguridad.

[0012] El documento US2006/129794A1 describe un método para actualizar un programa en una memoria flash sin proceder con el reinicio completo del procesador después de la instalación de la actualización. Un procesador se une con una unidad de gestión de memoria, una memoria de acceso aleatorio y una memoria flash. El contenido de la memoria flash es dinámicamente renovado por una actualización de una tabla de página en la que un indicador designa un nuevo programa almacenado en la memoria flash. El programa obsoleto es bien eliminado, o mantenido en la memoria flash después de la actualización de la tabla de página.

Resumen de la invención

[0013] El objetivo de la invención es de llevar a cabo una actualización completa o parcial de la pieza de software en un sistema de forma fiable evitando los estados inestables o incoherentes transitorios que generan fallos de seguridad en el sistema. Otro objetivo es ejecutar una actualización fijada en un entorno con un espacio de memoria limitada como en el campo de portadores de datos portátiles incluido software operativo integrado.

[0014] Este objetivo se alcanza por un método ejecutado por una aplicación para actualizar datos almacenados en una pluralidad de memorias físicas conectada a una unidad de gestión de memoria controlada por un microprocesador, dicha unidad de gestión de memoria realizando una transacción de actualización traduciendo direcciones de página de memoria virtual en direcciones de páginas de memoria física por lectura de una tabla de página siguiendo una pluralidad de indicadores de dirección que forman una estructura de ramificación, el método se caracteriza porque éste comprende las siguientes etapas:
- definir, por un primer atributo de versión, indicadores de una primera estructura de sub-ramificación, dicha primera estructura de sub-ramificación siendo usada para la unidad de gestión de memoria para el actual acceso de la aplicación a las páginas de memoria física,
- definir, por un segundo atributo de versión, indicadores para ser usados por la unidad de gestión de memoria, para la transacción de actualización de datos en dichas páginas de memoria física,
- seleccionar, por la aplicación, un conjunto de páginas de memoria libre comprendiendo páginas con direcciones físicas designadas por indicadores definidos por el primer atributo de versión y páginas con direcciones físicas designadas por indicadores definidos por el segundo atributo de versión, dichos indicadores forman una segunda estructura de sub-ramificación,
- actualizar datos de las páginas de memoria física seleccionadas designadas por los indicadores de la segunda estructura de sub-ramificación,
- configurar, por la unidad de gestión de memoria, la tabla de página de manera que la aplicación usa los indicadores de la segunda estructura de sub-ramificación para el corriente acceso a las páginas de memoria física en vez de los indicadores de la primera estructura de sub-ramificación usados antes de la actualización.

[0015] El método de la invención consiste en una actualización de antecedentes de datos almacenados en memorias físicas sin afectación de las corrientes operaciones realizadas por el microprocesador. Sólo cuando la actualización está totalmente finalizada, la aplicación cambia de la versión antigua a la nueva versión de datos. Esta conmutación ocurre por una reconfiguración de la tabla de página durante la cual la primera estructura de sub-ramificación de indicadores que acceden a la versión antigua de los datos de memoria se sustituye por la segunda estructura de sub-ramificación de indicadores así permitiendo el acceso a la nueva versión de datos. Este método de actualización previene los estados incoherentes transitorios del sistema mientras éste funciona con la versión precedente de datos hasta que la instalación de la versión nueva se vuelve utilizable. En caso de interrupción del proceso de actualización, la aplicación siempre será capaz de reinicializar debido a que la versión antigua de datos se puede reactivar por retorno a la configuración precedente de la tabla de página.

[0016] Otra ventaja es que este método se puede aplicar a sistemas de capacidad de memoria limitada tal como tarjetas de chip o módulos de seguridad. De hecho, la definición por un primer atributo de versión de los indicadores de la

estructura de ramificación que representa las direcciones de memoria de los datos de versión actual y la definición por un segundo atributo de versión de los datos representantes de indicadores para ser actualizados permite la construcción de dos estructuras de ramificación incluyendo indicadores comunes. La presencia de estos indicadores comunes permite la prevención de la multiplicación de la capacidad de memoria necesaria para almacenamiento de dos versiones de datos.

Breve descripción de las figuras

[0017] La invención será mejor entendida mediante la siguiente descripción detallada, que se refiere a las figuras anexas dadas como ejemplos no limitativos.
- La Figura 1 muestra un diagrama de bloques de una actualización de una página de memoria realizada modificando el atributo de versión de un indicador
- La Figura 2 muestra una extensión del diagrama de bloques de la figura 1 en el que la actualización causa la creación de tres páginas de memoria nuevas y dos nuevos indicadores.

Descripción detallada de la invención

[0018] La Figura 1 muestra el mecanismo para actualizar una página de memoria P1 con una dirección virtual (VA) por medio de una unidad de gestión de memoria MMU traduciendo las direcciones de página virtual en las direcciones de página correspondientes a memorias físicas M. Esta traducción se realiza por medio de una o más tablas de página (PT) comprendiendo un conjunto de indicadores a través de los cuales la unidad de gestión de memoria parece en orden para averiguar la dirección física de la página de memoria designada por la dirección de la página virtual de inicio VA1. Estos indicadores forman nodos según la terminología usada en el campo relacionado con unidades de gestión de memorias MMU.

[0019] Según la invención un nodo tiene un atributo O (impar) o un atributo E (par) correspondiente respectivamente a un indicador Impar y a un indicador Par o por otra parte a una versión par o impar del nodo. Los indicadores indican bien otro indicador impar o par de otro nodo o la dirección física de la página de memoria que puede también contener un indicador hacia datos localizados en una dirección predeterminada. En otras palabras, un nodo reúne al menos un par de indicadores impares y pares y cambia a una versión impar o par dependiendo de la versión indicadora usada. Según un ejemplo de implementación, los indicadores impares indican páginas de memoria física con una dirección impar mientras los indicadores pares designan páginas con una dirección par.

[0020] Los nodos se distribuyen en la tabla según niveles diferentes L1, L2, L3 y forman una estructura de ramificación A con una entrada E1 resultando de la dirección de página virtual de inicio VA1 y salidas (S1, S2; S3) conduciendo a direcciones de páginas de memoria física M. En la mayoría de los casos, estas direcciones de página se distribuyen en varios tipos de memorias, a saber ROM (memoria de sólo lectura), NVM (memoria no volátil), RAM (memoria de acceso aleatorio), etc., y esto en una forma discontinua, es decir en una forma no consecutiva donde la página N+1 no sigue necesariamente la página N en el mismo tipo de memoria.

[0021] En el ejemplo de la Figura 1, la unidad de gestión de memoria MMU se configura en orden para usar la versión O (Impar) de los nodos antes de iniciar la actualización. La página de memoria virtual P1 con la dirección VA1 corresponde a la página P1' de la memoria física M. La unidad de gestión MMU usando la tabla de página PT determina la dirección de la página P1' por mapa a través de una estructura de sub-ramificación SA1 de la estructura de ramificación global A usando los indicadores O (Impar) de los nodos L1a, L2b y L3c entre la entrada E1 y la salida S1 conduciendo a la página P1'. Las salidas S2 y S3 conducen a otras páginas de la memoria física o a otros nodos (no representados) de la estructura de sub-ramificación SA1 terminando en direcciones de página física.

[0022] Durante una actualización, la aplicación selecciona una página libre de memoria física usando los indicadores pares (E) de los nodos. Así en el ejemplo de la Figura 1, datos son escritos en la nueva página P1 n usando nodos L1 a, L2b y L3c en su versión Par (E) formando una estructura de sub-ramificación SA2 de la estructura de ramificación A conduciendo a la página de memoria P1 n.

[0023] Una vez la actualización ha sido terminada la unidad de gestión de memoria modifica su configuración para usar el nodo L3c en su versión Par en vez de versión Impar para finalizar en la nueva página P1 n. La versión de los otros nodos L1 L2a, L2b, L3a y L3b permanece invariada.

[0024] Esta operación después de la modificación de datos en una página de memoria física es similar a la conmutación de una primera estructura de sub-ramificación SA1 a una segunda estructura de sub-ramificación SA2 con respecto a la página P1' que será sustituida por la página P1 n. Las otras páginas enlazadas a las salidas de estructura de ramificación S2 y S3 no se modifican y son dirigidas mediante versiones impares de los nodos correspondientes.

[0025] Según una forma de realización, la página P1' señalada por la versión impar del nodo L3c se puede reciclar escribiendo en ésta el contenido de la página P1 n designada por el indicador par del nodo L3c. De esta manera el nodo L3c puede quedar en su versión impar mientras designando el contenido actualizado de la página P1'.

[0026] En la terminología de las unidades de gestión de memoria MMU, la operación para cambiar la versión de un nodo para designar una nueva página de memoria es llamada "confirmación de hardware". La operación de la forma de realización anteriormente mencionada que consiste en la sustitución del contenido de página señalado por el nodo en la versión impar con el contenido de página señalado por el nodo en la versión par es llamada "confirmación de software".

[0027] Por supuesto, una actualización puede referirse a diferentes páginas de diferentes tipos de memoria disponibles para la aplicación del sistema. Todas estas páginas tienen una dirección virtual usada por el microprocesador que será convertida en direcciones de páginas físicas mediante la tabla de página de la unidad de gestión de memoria MMU.

[0028] La Figura 2 muestra un ejemplo donde tres páginas adicionales se actualizan con respecto al ejemplo de la Figura 1. Como en el ejemplo precedente, la unidad de gestión de memoria MMU se configura para usar la versión O (Impar) de los nodos antes de la actualización.

[0029] Durante una etapa para preparar la actualización, la aplicación explora la memoria disponible para determinar de una parte qué páginas deben ser modificadas y por otro lado el número de páginas nuevas que deben ser asignadas según el tamaño de la actualización y el de la memoria.

[0030] En una primera etapa, la aplicación almacena datos de la actualización en una página existente P1 n por uso del Indicador par del nodo L3c como antes. Por otra parte, la aplicación asigna las páginas P2n, P3n y P4n para almacenar los nuevos datos de actualización, que necesita la creación de dos nuevos nodos L2c y L3d en el segundo y tercer nivel (L2, L3) de la estructura de ramificación. Estos nuevos nodos L2c y L3d serán designados por el indicador Par del nodo L2a localizado en el segundo nivel L2. La tabla de página así se vuelve más grande proporcionalmente al aumento en el número de páginas de memoria usada por la actualización. El número de nodos con indicadores impares y pares aumenta de la misma manera para proporcionar las siguientes actualizaciones usando una u otra de las versiones de los nodos para apuntar a las páginas apropiadas.

[0031] En una segunda etapa, cuando los datos de actualización se almacenan en las páginas de memoria física apropiada, estos son verificados antes de su implementación mediante la tabla de páginas modificadas. Esta verificación consiste en controlar la integridad de los datos almacenados para detectar cualquier modificación posible o manipulación que pueda ser realizada durante el proceso de actualización.

[0032] La verificación es preferiblemente realizada con datos de control que forman una secuencia de control (suma de control) o una firma que resulta de un resumen calculado con una función unidireccional matemática (Hash) y encriptada con una clave conocida por el sistema. Estos datos de control calculados en todos o parte de los datos de actualización son facilitados por éste y almacenados en una o más páginas de memoria predeterminada/s. La aplicación de instalación de actualización luego recalcula estos datos de control con un programa o algoritmo correspondiente usando los datos de actualización eficazmente almacenados en las memorias.

[0033] Una comparación exitosa de los datos de control almacenados con los datos calculados permite a la unidad de gestión de memoria MMU que convalide la instalación de actualización y la modificación de versión de los nodos relacionada con dicha actualización en la tabla de página.

[0034] Si la comparación falla, la unidad de gestión de memoria MMU bloquea la conmutación dela antigua a la nueva versión. Por consiguiente, la aplicación trabajará en las condiciones que preceden a la actualización, es decir con una versión de los nodos de la tabla de página quedando invariada.

[0035] En el ejemplo de la Figura 2, cuando la actualización es instalada y convalidada, la operación de inicio de la nueva versión o "confirmación" realizada por la unidad de gestión de memoria MMU consiste en modificar la versión de nodos L2a y L3c de la versión impar en la Versión par. El indicador de Versión par del segundo nodo de nivel L2a indicará así el nuevo segundo nodo de nivel L2c cuyo indicador impar o par designa el nuevo tercer nodo de nivel L3d. Éste es un ejemplo de nodo con tres pares de indicadores impares y pares destinados a dirigir las tres páginas nuevas (P2n, P3n, P4n) de memoria física. Estas páginas son respectivamente designadas por uno de los indicadores impares o pares del nodo L3d. El otro indicador de cada par se puede reservar para otra actualización de una de estas tres páginas de memoria (P2n, P3n, P4n)).

[0036] El indicador de versión par del nodo L3c del tercer nivel designa la nueva primera página P1 n que reemplaza la página P1' que fue designada por la versión impar del nodo L3c en la versión antigua de la tabla.

[0037] Además de la adición de tres páginas de memoria nueva, la actualización ha modificado así la estructura de ramificación de la tabla de página a través de la creación de dos nuevos nodos y el cambio de versión de otros dos nodos.

[0038] Debe ser notado que los nodos de una estructura de ramificación en una tabla de página de memoria no son todos modificables por la unidad de gestión de memoria, es decir capaz de cambiar de una versión impar a una versión par e inversamente. Algunos nodos así corresponden a un único indicador que permanece invariable durante las actualizaciones.

[0039] Ya que cada nodo tiene un espacio en la memoria del sistema proporcional al número de indicadores asociados al nodo y para minimizar el espacio de memoria ocupado por las estructuras de ramificación, los nodos se pueden definir por un parámetro de configuración. Éste informa a la unidad de gestión de memoria MMU sobre el número de indicadores asociado a dicho nodo que define una característica invariable o variable del nodo. Durante una actualización, un nodo variable corresponde a uno o más pares de indicadores impar/par pudiendo cambiar la versión mientras un nodo invariable generalmente corresponde a un único indicador que permanece constante con un atributo impar o par o sin atributo. Cuando la unidad de gestión de memoria MMU construye una estructura de ramificación para una nueva actualización, dicha unidad considera el parámetro de configuración de los nodos para racionalizar el uso de los nodos variables y de los nodos invariables para salvar el espacio de memoria usado por la estructura de ramificación.

[0040] El ejemplo de la figura 2 no es exhaustivo con respecto a la configuración de la tabla de página usada para esta actualización. De hecho, una modificación y la creación de otros nodos a niveles diferentes pueden llevar a un resultado similar. En otras palabras, varias estructuras de ramificación son posibles en una tabla de página para traducir un conjunto dado de direcciones de página de memoria virtual en un conjunto de direcciones de páginas de memoria física correspondiente.

[0041] Para acelerar el mapa de la tabla de página durante las actualizaciones, algunas de las direcciones de página física obtenidas de las direcciones virtuales correspondientes durante los mapas precedentes de la tabla de página se puede memorizar en una memoria caché asociada a la unidad de gestión de memoria MMU. Esta memoria caché llamada "memoria tampón de traducción de direcciones de página" o "Translation Lookaside Buffer" o "TLB cache" permite realizar una conversión inmediata de direcciones sin mapa a través de la tabla de página. El tamaño limitado de esta memoria caché no permite memorizar todos los mapas de la tabla de página, sólo el último.

[0042] Por ejemplo, la memoria caché puede contener un indicador dirigido hacia una página de memoria física conteniendo datos de control que la unidad de gestión de memoria MMU usa para verificar la integridad de la actualización antes de su validación.

REIVINDICACIONES

1. Método ejecutado por una aplicación para actualizar datos almacenados en una pluralidad de memorias físicas conectadas a una unidad de gestión de memoria (MMU) controlada por un microprocesador, dicha unidad de gestión de memoria (MMU) realizando una transacción de actualización traduciendo direcciones de página de memoria virtual en direcciones de páginas de memoria física por lectura de una tabla de página (PT) siguiendo una pluralidad de indicadores de dirección que forman una estructura de ramificación, el método se caracteriza porque éste comprende las siguientes etapas:
- definir, por un primer atributo de versión (O), indicadores de una primera estructura de sub-ramificación (SA1), dicha primera estructura de sub-ramificación (SA1) siendo usada por la unidad de gestión de memoria (MMU) para el actual acceso de la aplicación a las páginas de memoria física (P1),
- definir, por un segundo atributo de versión (E), indicadores para ser usados por la unidad de gestión de memoria (MMU), para la transacción de actualización de datos en dichas páginas de memoria física (P1),
- seleccionar, por la aplicación, de un conjunto de páginas de memoria libre incluyendo páginas con direcciones físicas designadas por indicadores definidos por el primer atributo de versión (O) y páginas con direcciones físicas designadas por indicadores definidos por el segundo atributo de versión(E), dichos indicadores formando una segunda estructura de sub-ramificación (SA2),
- actualizar datos de las páginas de memoria física seleccionadas designadas por los indicadores de la segunda estructura de sub-ramificación (SA2),
- configurar, por la unidad de gestión de memoria (MMU), la tabla de página (PT) de manera que la aplicación usa los indicadores de la segunda estructura de sub-ramificación (SA2) para el actual acceso a las páginas de memoria física en vez de los indicadores de la primera estructura de sub-ramificación (SA1) usados antes de la actualización.

2. Método según la reivindicación 1 caracterizado por el hecho de que la tabla de página (PT) incluye nodos reuniendo al menos un par de indicadores definidos por el primer atributo de versión (O) y por el segundo atributo de versión (E) respectivamente, dichos nodos formando una estructura de ramificación leída por la unidad de gestión de memoria (MMU).

3. Método según la reivindicación 1 o 2 caracterizado por el hecho de que la primera estructura de sub-ramificación (SA) comprende nodos definidos por el primer atributo de versión (O), dichos nodos con indicadores definidos por el primer atributo de versión (O) usados por la aplicación para acceder a las páginas de memoria física.

4. Método según cualquiera de las reivindicaciones 1 a 3 caracterizado por el hecho de que la segunda estructura de sub-ramificación (SA2) incluye al menos un nodo definido por el segundo atributo de versión (E), dicho nodo con un indicador definido por el segundo atributo de versión (E) usado por la aplicación para designar una nueva página de memoria destinada a almacenar datos de actualización.

5. Método según cualquiera de las reivindicaciones 1 a 4 caracterizado por el hecho de que la tabla de página (PT) se configura por la unidad de gestión de memoria (MMU) de manera que la aplicación accede a las páginas de memoria actualizada usando los indicadores de la segunda estructura de sub-ramificación (SA2) incluyendo los nodos definidos por el primer atributo de versión (O) y al menos un nodo definido por el segundo atributo de versión (E).

6. Método según cualquiera de las reivindicaciones 1 a 5 caracterizado por el hecho de que, durante la actualización de datos de las páginas de memoria física, comprende las siguientes etapas adicionales:
- explorar la memoria física disponible y asignación de páginas de memoria adicional destinada a almacenar datos de actualización,
- crear nodos adicionales que designan las páginas adicionales previamente asignadas,
- modificar el atributo de versión de al menos un nodo de la estructura de ramificación durante la configuración de la tabla de página (PT) por la unidad de gestión de memoria (MMU) de manera que la aplicación accede a los nodos adicionales que designan las páginas adicionales previamente asignadas.

7. Método según cualquiera de las reivindicaciones 1 a 4 caracterizado por el hecho de que, cuando los datos de actualización se almacenan en las páginas de memoria física apropiada, incluye una etapa de verificación que consiste en comparar datos de control previamente almacenados en una o más páginas de memoria predeterminadas con datos de control calculados de los datos de actualización almacenados en las memorias físicas.

8. Método según la reivindicación 7 caracterizado por el hecho de que la verificación se realiza con datos de control que forman una firma que resulta de un resumen calculado con una función unidireccional matemática del tipo hash y encriptado con una clave conocida por el sistema, dichos datos de control calculados en todos o parte de los datos de actualización se proveen con la actualización y son almacenados en una o más páginas de memoria predeterminadas, la aplicación recalcula los datos de control usando los datos de actualización eficazmente almacenados y los compara con los datos de control previamente almacenados.

9. Método según la reivindicación 7 o 8 caracterizado por el hecho de que, si la comparación es exitosa, la unidad de gestión de memoria (MMU) convalida la instalación de la actualización y la modificación de los atributos de versión de los nodos concernidos por dicha actualización en la tabla de página (PT).

10. Método según la reivindicación 7 o 8 caracterizado por el hecho de que, si la comparación falla, la unidad de gestión de memoria (MMU) bloquea la conmutación de la antigua a la nueva versión de la aplicación, dicha aplicación trabaja en las condiciones que preceden a la actualización con atributos de versión invariados de los nodos de la tabla de página (PT).

11. Método según la reivindicación 1 caracterizado por el hecho de que las direcciones de página de memoria física son leídas desde una memoria caché asociada a la unidad de gestión de memoria (MMU), dichas direcciones de página de memoria física almacenadas en dicha memoria caché siendo obtenida de las direcciones correspondientes virtuales durante la lectura precedente de la tabla de página (PT).

12. Método según cualquiera de las reivindicaciones 1 a 6 caracterizado por el hecho de que, durante la modificación del atributo de versión de un nodo de la tabla de página (PT) por la unidad de gestión de memoria (MMU), el contenido de una página de memoria designado por un indicador de dicho nodo definido por el atributo de la versión que precede a la actualización se sustituye por el contenido de la página de memoria designado por el indicador definido por el atributo de la nueva versión.

13. Método según la reivindicación 2 caracterizado por el hecho de que los nodos se definen por un parámetro de configuración que informa a la unidad de gestión de memoria (MMU) del número de indicadores asociados a dicho nodo definiendo una característica invariable o variable del nodo durante una actualización.

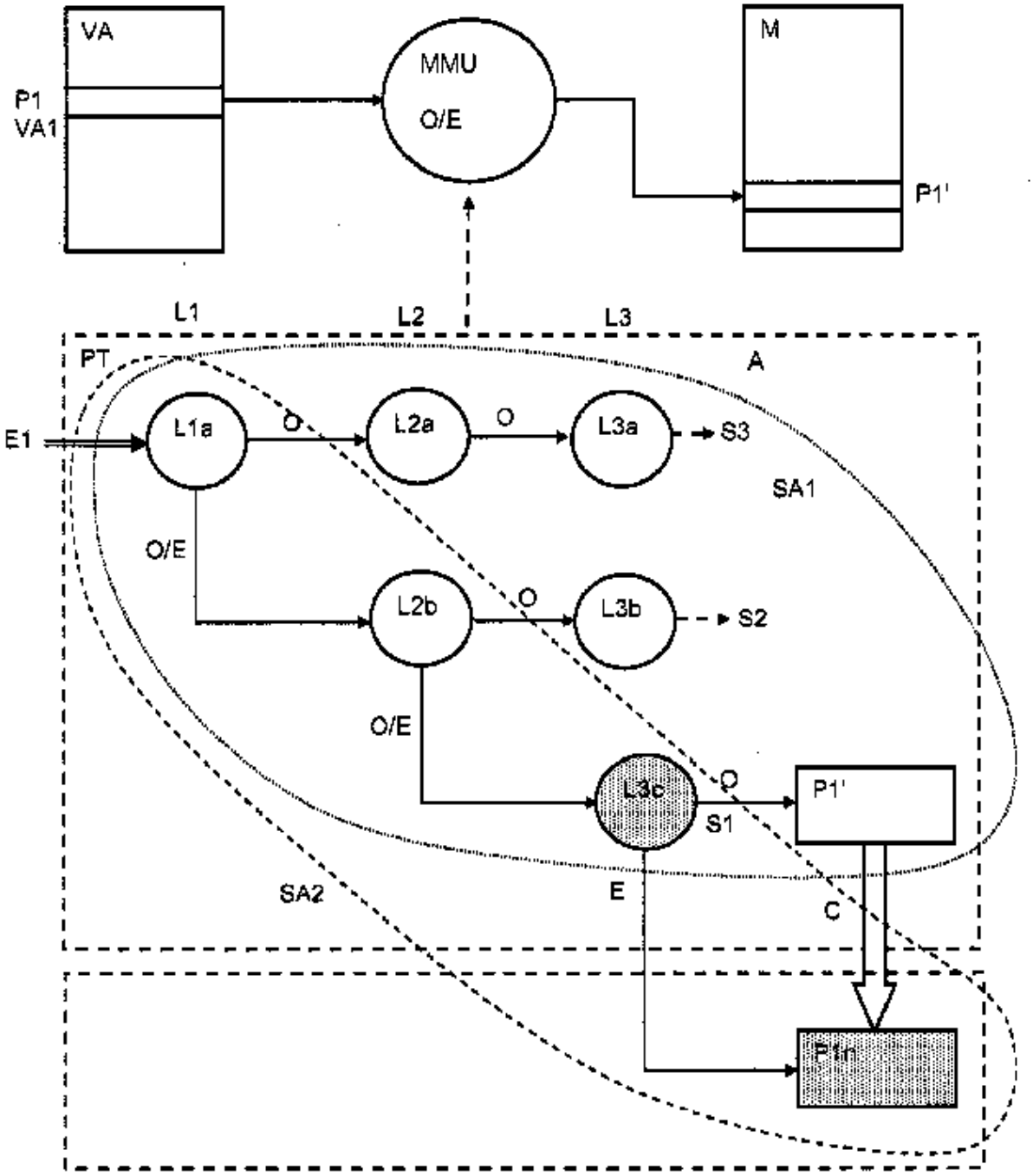
VA
P1
VA1
MMU
O/E
M
P1'
L1
L2
L3
PT
A
E1
L1a
O
L2a
O
L3a
S3
SA1
O/E
L2b
O
L3b
S2
O/E
L3c
O
P1'
S1
SA2
E
C
P1n

Fig. 1

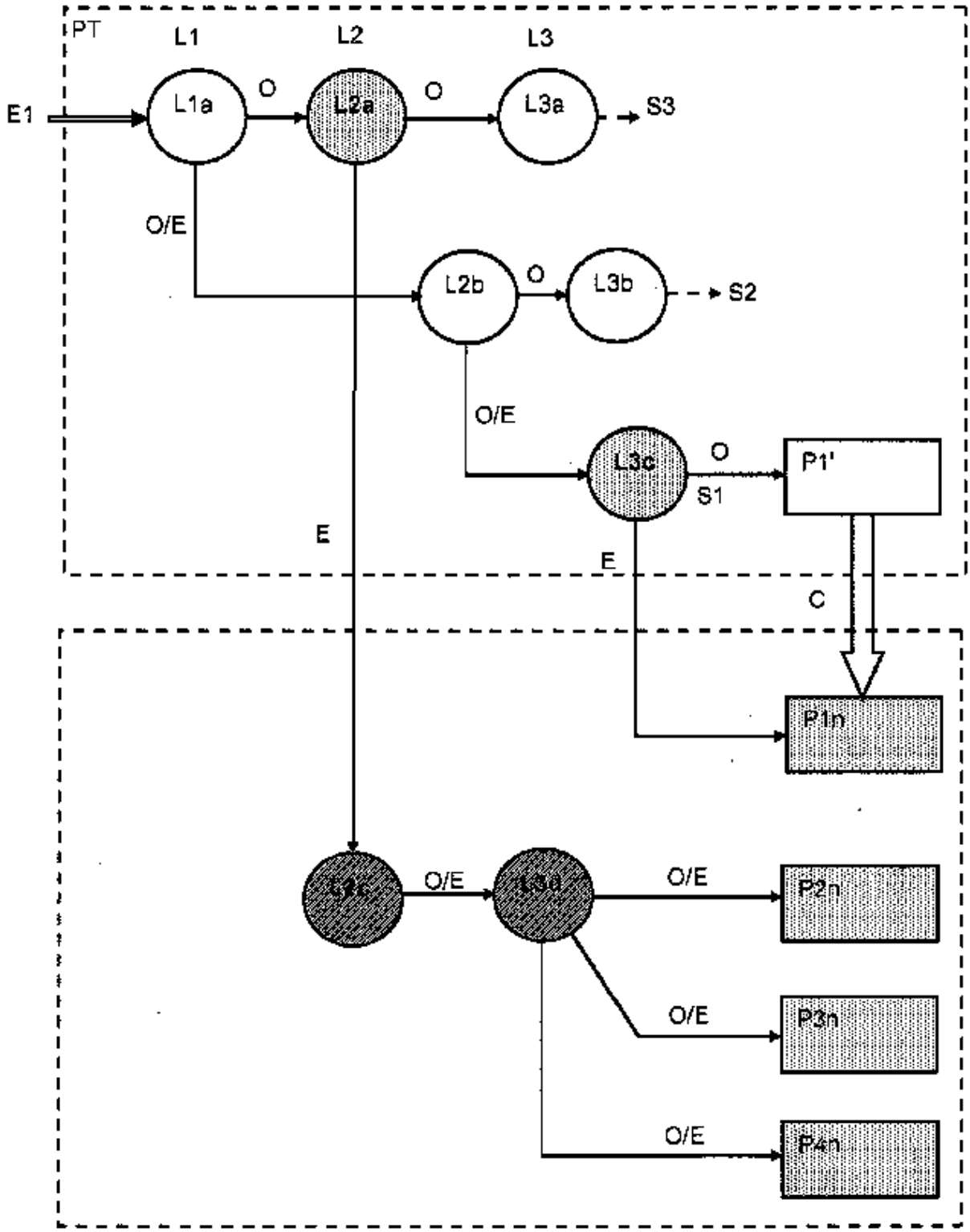
PT
L1
L2
L3
E1
L1a
O
L2a
O
L3a
S3
O/E
L2b
O
L3b
S2
O/E
L3c
O
P1'
S1
E
E
C
P1n
L3c
O/E
L3d
O/E
P2n
O/E
P3n
O/E
P4n

Fig. 2