(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 386 087**

(51) Int. CI.:
G06F 9/312 (2006.01)
G06F 9/38 (2006.01)
G06F 9/46 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **08254178 .0**
(96) Fecha de presentación: **30.12.2008**
(97) Número de publicación de la solicitud: **2075690**
(97) Fecha de publicación de la solicitud: **01.07.2009**

(54) Título: **Mecanismo para una atomicidad fuerte en un sistema de memoria transaccional**

(30) Prioridad:
30.12.2007 US 967231

(45) Fecha de publicación de la mención BOPI:
08.08.2012

(45) Fecha de la publicación del folleto de la patente:
08.08.2012

(73) Titular/es:
INTEL CORPORATION
2200 MISSION COLLEGE BOULEVARD
SANTA CLARA, CA 95054, US

(72) Inventor/es:
Saha, Bratin;
Adl-Tabatabai, Ali-Reza;
Wang, Cheng y
Shpeisman, Tatiana

(74) Agente/Representante:
Carpintero López, Mario

DESCRIPCIÓN

Mecanismo para una atomicidad fuerte en un sistema de memoria transaccional

Campo

La presente invención se refiere al campo de la ejecución en procesador y, en particular, acerca de la ejecución de grupos de instrucciones.

Antecedentes

Los avances en el procesamiento de semiconductores y en el diseño de lógica han permitido un aumento en la cantidad de lógico que puede estar presente en dispositivos de circuitos integrados. En consecuencia, han evolucionado configuraciones de sistemas de ordenadores desde circuitos integrados únicos o múltiples en un sistema a múltiples núcleos y múltiples procesadores lógicos presentes en circuitos integrados individuales. Típicamente, un procesador o un circuito integrado comprende un único dado procesador, pudiendo incluir el cubo procesador cualquier número de núcleos o de procesadores lógicos.

El número siempre creciente de núcleos y procesadores lógicos en los circuitos integrados permite que se ejecuten de forma concurrente más hilos de soporte lógico. Sin embargo, el aumento en el número de hilos de soporte lógico que pueden ser ejecutados simultáneamente ha creado problemas de sincronización de datos compartidos entre los hilos de soporte lógico. Una solución común para acceder a datos compartidos en sistemas de múltiples núcleos o de múltiples procesadores lógicos comprende el uso de bloqueos para garantizar la exclusión mutua en accesos múltiples a los datos compartidos. Sin embargo, la capacidad siempre creciente de ejecutar múltiples hilos de soporte lógico potencialmente da como resultado una falsa disputa y una serialización de la ejecución.

Por ejemplo, consideremos una tabla de claves calculadas que contiene datos compartidos. Con un sistema de bloqueos, un programador puede bloquear toda la tabla de claves calculadas, permitiendo que un hilo acceda a toda la tabla de claves calculadas. Sin embargo, el desempeño y el rendimiento de otros hilos se ve potencialmente afectado de manera adversa, ya que son incapaces de acceder a ninguna entrada de la tabla de claves calculadas hasta que se quite el bloqueo. Alternativamente, puede bloquearse cada entrada de la tabla de claves calculadas. Sin embargo, esto aumenta la complejidad de programación, ya que los programas tienen que responder de más bloqueos dentro de una tabla de claves calculadas.

Otra técnica de sincronización de datos incluye el uso de memoria transaccional (TM). A menudo, la ejecución transaccional incluye la ejecución especulativa de una agrupación de una pluralidad de microoperaciones, operaciones o instrucciones. En el ejemplo que antecede, ambos hilos se ejecutan dentro de la tabla de claves calculadas, y sus accesos son monitorizados/seguidos. Si ambos hilos acceden/alteran la misma entrada, una de las transacciones puede ser abortada para resolver el conflicto. Un tipo de ejecución transaccional incluye una memoria transaccional de soporte lógico (STM) en la que los accesos son objeto de seguimiento, la resolución de conflictos, el aborto de tareas y otras tareas transaccionales son llevados a cabo en soporte lógico.

Previamente, para garantizar que no ocurren conflictos en tiempo de ejecución entre operaciones de memoria transaccional y operaciones de memoria no transaccional, los compiladores tratan cada operación de memoria no transaccional como una transacción de una sola operación. En otras palabras, se insertan operaciones transaccionales en operaciones de memoria no transaccional para garantizar que no ocurren conflictos de tiempo de ejecución. Sin embargo, la ejecución de barreras exhaustivas en las operaciones de memoria no transaccional desperdicia potencialmente ciclos de ejecución.

El documento “Transactional Lock-Free Execution of Lock-Based Programs”, de Ravi Rajwar y James Goodman (publicado en PROCEEDINGS OF THE 10TH SYMPOSIUM ON ARCHITECTURAL SUPPORT FOR PROGRAMMING LANGUAGES AND OPERATING SYSTEMS), propone una técnica denominada “eliminación transaccional de bloqueos” (TLR) que permite que un programa que usa la sincronización basada en bloqueos sea ejecutado por el soporte físico de una manera libre de bloqueos. El bloqueo define el alcance de la transacción. El código que usaba el bloqueo es ejecutado entonces especulativamente como una transacción sin solicitar ni adquirir el bloqueo. Se usa un esquema de resolución de conflictos para ejecutar transacciones contrapuestas en el debido orden.

El documento “Virtualizing Transactional Memory”, de Rajwar et. al. (publicado en COMPUTER ARCHITECTURE. 2005. ISCA ’05. PROCEEDINGS. 32ND INTERNATIONAL SYMPOSIUM ON MADISON WI, USA 04-08 JUNIO DE 2005. PISCATAWAY, NJ, EE. UU. IEEE 4 JUNIO DE 2005), propone un sistema que aborda las deficiencias en los sistemas de memoria transaccional de soporte físico. Su sistema, denominado “memoria transaccional virtual” (VTM), es un sistema transparente al usuario que protege al programador contra diversas limitaciones de recursos específicas a la plataforma. La VTM mantiene la ventaja de rendimiento de las transacciones en soporte físico, incurre en una baja sobrecarga en el tiempo y tiene costes modestos en soporte del soporte físico.

Breve descripción de los dibujos

La presente invención, tal como se expone en las reivindicaciones independientes 1, 10 y 19, se ilustra a título de ejemplo y no se pretende que esté limitada por las figuras de los dibujos adjuntos.

La Figura 1 ilustra una realización de un sistema capaz de proporcionar una atomicidad fuerte eficiente.

La Figura 2 ilustra una realización de un sistema de memoria transaccional de soporte lógico (STM) para proporcionar una atomicidad fuerte eficiente.

La Figura 3a ilustra una realización de lógica para proporcionar una atomicidad fuerte eficiente.

La Figura 3b ilustra otra realización de lógica para proporcionar una atomicidad fuerte eficiente.

La Figura 4a ilustra una realización de un diagrama de flujo de un procedimiento para proporcionar una atomicidad fuerte eficiente.

La Figura 4b ilustra una realización del diagrama de flujo representado en la Figura 4a.

La Figura 4c ilustra otra realización del diagrama de flujo representado en la Figura 4a.

Descripción detallada

En la siguiente descripción, se exponen numerosos detalles específicos, tales como ejemplos de soporte específico de soporte físico para la ejecución transaccional, procedimientos específicos de seguimiento/metadatos, tipos específicos de ubicaciones/memoria en procesadores y tipos específicos de accesos a memoria y ubicaciones, etc., para proporcionar una comprensión cabal de la presente invención. Sin embargo, será evidente para un experto en la técnica que no es preciso emplear estos detalles específicos para poner en práctica la presente invención. En otros casos, componentes o procedimientos bien conocidos, tales como la codificación de transacciones en soporte lógico, la demarcación de transacciones, arquitecturas específicas de procesadores de núcleos múltiples e hilos múltiples, soporte físico de transacciones, organizaciones de memoria tampón y detalles operacional específicos de microprocesadores, no han sido descritos con detalle para evitar hacer innecesariamente confusa la presente invención.

Un valor, según se usa en el presente documento, incluye cualquier representación conocida de un número, un estado, un estado lógico o un estado lógico binario. A menudo, el uso de niveles de lógica, valores de lógica o valores lógicos también se denomina unos y ceros, lo cual simplemente representa estados lógicos binarios. Por ejemplo, un 1 se refiere a un nivel lógico alto y 0 se refiere a un nivel lógico bajo. Sin embargo, en los sistemas de ordenadores se han usado otras representaciones de valores. Por ejemplo, el número decimal 10 también puede ser un valor binario 1010 y una letra hexadecimal A.

Además, los estados pueden ser representados por valores o porciones de valores. Por ejemplo, un estado de bloqueo puede ser representado por un primer valor en una ubicación, tal como un número impar, mientras que un número de versión, tal como un valor par, en la ubicación representa un estado sin bloqueo. Aquí puede usarse una porción de los valores primero y segundo para representar los estados, tales como los dos bits más bajos de los valores, un bit de signo asociado con los valores u otra porción de los valores. Además, las expresiones puesto a uno y puesto a cero, en una realización, se refieren a un valor o un estado por defecto y actualizado, respectivamente. Por ejemplo, un valor por defecto incluye, potencialmente, un valor lógico alto, es decir, puesto a uno, mientras que un valor actualizado incluye, potencialmente, un valor lógico bajo, es decir, puesto a cero.

El procedimiento y el aparato descritos en el presente documento son para proporcionar una atomicidad fuerte en un sistema de memoria transaccional. Específicamente, proporcionar una atomicidad fuerte se expone fundamentalmente con referencia a un sistema (STM) que utiliza una detección de conflictos basada en una línea de memoria tampón y coherencia basada en sellos de tiempo. Sin embargo, los procedimientos y los aparatos para proporcionar una atomicidad fuerte no están limitados de esa manera, ya que pueden ser implementados en cualquier sistema de memoria transaccional o en asociación con el mismo.

Con referencia a la **Figura 1**, se ilustra una realización de un procesador de múltiples elementos de proceso capaz de proporcionar una atomicidad fuerte optimizada en un sistema de memoria transaccional. Un elemento de proceso se refiere a un hilo, un proceso, un contexto, un procesador lógico, un hilo de soporte físico, un núcleo y/o cualquier elemento de proceso que comparta el acceso a recursos del procesador, tales como unidades de reserva, unidades de ejecución, canales y/o memorias tampón/memoria de mayor nivel. Típicamente, un procesador físico se refiere a un circuito integrado, que, potencialmente, incluye cualquier número de otros elementos de proceso, tales como núcleos o hilos de soporte físico.

Un núcleo se refiere a menudo a lógica situada en un circuito integrado capaz de mantener un estado arquitectónico independiente en el que cada estado arquitectónico independientemente mantenido está asociado con al menos algunos recursos dedicados de ejecución. A diferencia de los núcleos, un hilo de soporte físico se refiere,

típicamente, a cualquier lógica situada en un circuito integrado capaz de mantener un estado arquitectónico independiente en el que los estados arquitectónicos independientemente mantenidos comparten el acceso a los recursos de ejecución. El procesador físico ***100,*** tal como se ilustra en la **Figura 1**, incluye dos núcleos, el núcleo ***101*** y el ***102***, que comparten el acceso a la memoria tampón ***110*** de nivel superior. Además, el núcleo ***101*** incluye dos hilos ***101a*** y ***101b*** de soporte físico***,*** mientras que el núcleo ***102*** incluye dos hilos ***102a*** y ***102b*** de soporte físico. Por lo tanto, las entidades de soporte lógico, tales como un sistema operativo, ven al procesador ***100***, potencialmente, como cuatro procesadores separados, mientras que el procesador ***100*** es capaz de ejecutar cuatro hilos de soporte lógico.

Como puede verse, cuando ciertos recursos están compartidos y otros están dedicados a un estado arquitectónico, la línea entre la nomenclatura de un hilo de soporte físico y un núcleo se solapa. No obstante, a menudo, un sistema operativo ve un núcleo y un hilo de soporte físico como procesadores lógicos individuales, siendo capaz el sistema operativo de programar operaciones individualmente en cada procesador lógico. En otras palabras, el soporte lógico ve dos núcleos o hilos en un procesador físico como dos procesadores independientes. además, cada núcleo incluye potencialmente múltiples hilos de soporte físico para ejecutar múltiples hilos de soporte lógico. Por lo tanto, un elemento de proceso incluye cualquiera de los elementos mencionados anteriormente capaz de mantener un contexto, tales como núcleos, hilos, hilos de soporte físico, máquinas virtuales u otros recursos.

En una realización, el procesador ***100*** es un procesador de múltiples núcleos capaz de ejecutar múltiples hilos en paralelo. Aquí, un primer hilo está asociado con los registros ***101a*** de estado arquitectónico, un segundo hilo está asociado con los registros ***101b*** de estado arquitectónico, un tercer hilo está asociado con los registros ***102a*** de estado arquitectónico y un cuarto hilo está asociado con los registros ***102b*** de estado arquitectónico. La referencia a los elementos de proceso en el procesador ***100,*** en una realización, incluye la referencia a los núcleo ***101*** y ***102,*** así como a los hilos ***101a, 101b, 102a*** y ***102b.*** En otra realización, un elemento de proceso se refiere a elementos al mismo nivel en una jerarquía de dominio de proceso. Por ejemplo, el núcleo ***101*** y el ***102*** están en el mismo nivel de dominio, y los hilos ***101a, 101b, 102a*** y ***102b*** están en el mismo nivel de dominio, ya que están todos incluidos dentro del dominio de un núcleo.

Aunque el procesador ***100*** puede incluir núcleos asimétricos, es decir, núcleos con configuraciones, unidades funcionales y/o lógica diferentes, se ilustran núcleos simétricos. En consecuencia, el núcleo ***102,*** que se ilustra idéntico al núcleo ***101,*** no será presentado en detalle para evitar hacer confusa la exposición.

Tal como se ilustra, los registros ***101a*** de estado arquitectónico están replicados en los registros ***101b*** de estado arquitectónico, de modo que se pueden guardar estados/contextos arquitectónicos individuales para el procesador lógico ***101a*** y el procesador lógico ***101b.*** Otros recursos menores, tales como punteros a instrucciones y lógica de cambio de nombre en la lógica ***130*** de asignación de cambio de nombre, también pueden ser replicados para los hilos ***101a*** y ***101b.*** Algunos recursos, tales como memorias tampón de reordenación en la unidad ***135*** de reordenación/retirada***,*** ILTB ***120,*** memorias tampón de carga/almacenamiento y colas, pueden ser compartidos mediante partición. Otros recursos, tales como registros internos de uso general, registro base de tablas de páginas, memoria tampón de datos de bajo nivel y TLB ***115*** de datos, unidad(es) ***140*** de ejecución y porciones de la unidad ***135*** fuera de orden son compartidos plenamente de forma potencial.

El módulo ***105*** de interfaz de bus es para comunicarse con dispositivos externos al procesador ***100,*** tales como una memoria ***175*** del sistema, un conjunto de chips, un puente norte u otro circuito integrado. La memoria ***175*** puede estar dedicada al procesador ***100*** o estar compartida con otros dispositivos en un sistema. Ejemplos de memoria ***175*** incluyen la memoria dinámica de acceso aleatorio (DRAM), la RAM estática (SRAM), la memoria no volátil (memoria NV) y el almacenamiento a largo plazo.

Típicamente, la unidad ***105*** de interfaz de bus incluye memorias tampón de entrada/salida (E/S) para transmitir y recibir señales de bus en la interconexión ***170***. Ejemplos de interconexión ***170*** incluyen un bus con lógica de transceptor de gran velocidad (GTL), un bus GTL+, un bus de doble velocidad de transferencia de datos (DDR), un bus de bombeo, un bus diferencial, un bus coherente de memoria intermedia, un bus punto a punto, un bus multipunto u otra interconexión conocida que implemente cualquier protocolo conocido de bus. La unidad ***105*** de interfaz de bus, tal como se muestra, es también para comunicarse con la memoria intermedia ***110*** de nivel superior.

La memoria intermedia ***110*** de mayor nivel o más alejada es para poner en memoria intermedia elementos buscados y/o con los que se ha operado recientemente. Obsérvese que la memoria intermedia de mayor nivel o más alejada se refiere a niveles de memoria intermedia que aumentan o se alejan de la o las unidades de ejecución. En una realización, la memoria intermedia ***110*** de mayor nivel es una memoria intermedia de datos de segundo nivel. Sin embargo, la memoria intermedia ***110*** de mayor nivel no está limitada así, ya que puede ser o incluir una memoria intermedia de instrucciones, que puede también denominarse memoria intermedia de seguimiento de ejecución. En vez de ello, una memoria intermedia de seguimiento de ejecución puede estar acoplada después del decodificador ***125*** para almacenar trazas recientemente decodificadas. El módulo ***120*** también incluye potencialmente una memoria tampón de derivación para predecir derivaciones que han de ser ejecutadas/tomadas y una memoria tampón de traducción de instrucciones (ITLB) para almacenar entradas de traducción de direcciones para

instrucciones. Aquí, un procesador capaz de una ejecución especulativa busca potencialmente de antemano y ejecuta especulativamente derivaciones predichas.

El módulo ***125*** de decodificación está acoplado a la unidad ***120*** de búsqueda para decodificar elementos buscados. En una realización, el procesador ***100*** está asociado con una arquitectura de conjuntos de instrucciones (ISA), que define/especifica instrucciones ejecutables en el procesador ***100***. Aquí, a menudo, las instrucciones de código máquina reconocidas por la ISA incluyen una porción de la instrucción denominada código de operación, que referencia/especifica una instrucción o una operación que deben llevarse a cabo.

En un ejemplo, el bloque ***130*** de asignación y cambio de nombres incluye un asignado para reservar recursos, tales como ficheros de registro para almacenar resultados del procesamiento de instrucciones. Sin embargo, los hilos ***101a*** y ***101b*** son potencialmente capaces de una ejecución fuera de orden, en la que el bloque ***130*** de asignación y cambio de nombres también reserva otros recursos, tales como memorias tampón de reordenación para seguir los resultados de las instrucciones. La unidad ***130*** también puede incluir un cambiador de nombres de registros para cambiar el nombre de registros de referencia a programas/instrucciones a otros registros internos al procesador ***100***. La unidad ***135*** de reordenación/retirada incluye componentes tales como las memorias tampón de reordenación mencionadas en lo que antecede, memorias tampón de carga y memorias tampón de almacenamiento para soportar la ejecución fuera de orden y, posteriormente, la retirada en orden de instrucciones ejecutadas fuera de orden.

El bloque ***140*** de la o las unidades de programación y ejecución, en una realización, incluye una unidad de programación para programar instrucciones/operaciones en unidades de ejecución. De hecho, las instrucciones/operaciones son programadas potencialmente en unidades de ejecución según la disponibilidad de sus tipos. Por ejemplo, una instrucción de coma flotante es programada en un puerto de una unidad de ejecución que tiene una unidad de ejecución disponible de coma flotante. También se incluyen los ficheros de registro asociados con las unidades de ejecución para almacenar resultados de procesamiento de instrucciones de información. Unidades ejemplares de ejecución incluyen una unidad de ejecución de coma flotante, una unidad de ejecución de enteros, una unidad de ejecución de saltos, una unidad de ejecución de cargas, una unidad de ejecución de almacenamiento y otras unidades de ejecución conocidas.

La memoria intermedia de datos de menor nivel y la memoria tapón de traducción de datos (D-TLB) ***150*** están acopladas a la o las unidades ***140*** de ejecución. La memoria intermedia de datos es para almacenar elementos usados o con los que se ha operado recientemente, tales como operandos de datos, que son mantenidos potencialmente en estados de coherencia de memoria, tales como los estados modificado, exclusivo, compartido e inválido (MESI). La D-TLB ha de almacenar instrucciones virtuales/lineales recientes en traducciones de direcciones físicas. Como ejemplo específico, un procesador puede incluir una estructura de tablas de páginas para dividir la memoria física en una pluralidad de páginas virtuales. Puede usarse la memoria intermedia ***150*** de datos como una memoria transaccional u otra memoria para seguir los accesos provisionales durante la ejecución de una transacción, tal como se expone con más detalle en lo que sigue. Además, cuando se siguen los accesos provisionales utilizando un sistema STM, pueden mantenerse datos/datos de soporte lógico en la memoria ***175*** del sistema y ser metidas en la memoria intermedia ***150*** de nivel inferior.

Una transacción, que también puede denominarse sección crítica de código, incluye una agrupación de instrucciones, operaciones o microoperaciones que pueden estar a agrupadas por soporte físico, soporte lógico, soporte lógico inalterable o una combinación de los mismos. Por ejemplo, pueden usarse instrucciones u operaciones para demarcar una transacción o una sección crítica. Típicamente, durante la ejecución de una transacción, las actualizaciones de la memoria no son hechas globalmente visibles hasta que la transacción es confirmada. Mientras la transacción sigue pendiente, las ubicaciones de las que se carga y a las que se escribe dentro de la memoria son objeto de seguimiento. Tras la validación con éxito de esas ubicaciones de memoria, la transacción es confirmada y las actualizaciones realizadas durante la transacción son hechas globalmente visibles.

Sin embargo, si la transacción es invalidada durante su estado pendiente, la transacción es reiniciada sin hacer las actualizaciones globalmente visibles. En consecuencia, el estado pendiente de una transacción, tal como se usa en el presente documento, se refiere a una transacción que han comenzado su ejecución y no ha sido confirmada ni abortada; es decir, está pendiente. Implementaciones ejemplares para la ejecución transaccional incluyen un sistema de memoria transaccional de soporte físico (HTM), un sistema de memoria transacciones de soporte lógico (STM) y una combinación de los mismos.

Un sistema de memoria transaccional de soporte físico (HTM) se refiere a menudo a seguir el acceso, durante la ejecución, de una transacción con el procesador ***100*** en el soporte físico del procesador ***100***. Por ejemplo, la memoria intermedia ***150*** es para meter en memoria intermedia un elemento/objeto de datos desde la memoria ***175*** del sistema. Durante la ejecución de una transacción, se utiliza un campo de anotaciones/atributos, que está asociado con la línea de memoria intermedia en la memoria intermedia ***150*** que contiene el objeto de datos para seguir los accesos a la línea de memoria intermedia o desde la misma. En una realización, el campo de anotaciones incluye una celda de almacenamiento de lectura y una celda de almacenamiento de escritura. Cada una de las celdas de almacenamiento es puesta a 1 tras la correspondiente lectura o escritura para indicar si ha ocurrido una lectura o una escritura durante un estado pendiente de una transacción.

Un sistema de memoria transaccional de soporte lógico (STM) se refiere a menudo a seguir el acceso, la resolución de conflictos u otras tareas transaccionales de la memoria en o al menos en parte en soporte lógico. Como ejemplo general, un compilador, cuando es ejecutado, compila código de programa para insertar barreras de lectura y escritura para operaciones de carga y almacenamiento en consecuencia, que son parte de transacciones dentro del código de programa. Un compilador también puede insertar otras operaciones relacionadas o no con transacciones, tal como operaciones de confirmación, operaciones de aborto, operaciones de contabilidad, operaciones de detección de conflictos y operaciones de atomicidad fuerte.

Previamente, tal como se afirma en lo que antecede, las operaciones de memoria no transaccional se venían tratando como transacciones únicas en un sistema fuertemente atómico que proporcionaba aislamiento entre código transaccional y no transaccional. Normalmente, un compilador inserta operaciones en el acceso a memoria no transaccional para garantizar una atomicidad fuerte, es decir, garantizar la validez entre las operaciones de memoria transaccional y no transaccional. Un ejemplo de una operación previa insertada en una operación de acceso a memoria no transaccional para garantizar una atomicidad fuerte incluye una operación de bloqueo, tal como una operación de prueba, para determinar si un bloqueo indica que una ubicación de memoria está disponible o no tiene propietario. Sin embargo, en una realización, se inserta un número reducido de operaciones de atomicidad fuerte para garantizar la validez sin llevar a cabo algunas operaciones de vía lenta, tales como la operación de bloqueo, para determinar si un bloqueo está disponible. En lo que sigue se expone con más detalle el proporcionar una atomicidad fuerte eficiente.

Con referencia a la **Figura 2**, se representa una realización ilustrativa simplificada de un sistema STM. El objeto ***201*** de datos incluye cualquier granularidad de datos, tal como una palabra, un elemento/operando de datos, una instrucción, una línea de memoria, una línea de memoria intermedia, un objeto definido de lenguaje de programación, un campo de un objeto definido de lengua de programación, una tabla, una tabla de claves calculadas o cualquier otra estructura o cualquier otro objeto conocidos de datos. En una realización, se crea una clave calculada de una dirección que se refiere al objeto ***201*** de datos o está asociada con él, tal como una dirección física o virtual, a un índice a una matriz/tabla de ubicaciones de bloqueo/metadatos, tal como una matriz ***240*** de metadatos. Como ejemplo específico, varios bits inferiores de una dirección son desenmascarados y luego se crea con ellos una clave calculada a un índice en una matriz ***240*** de bloqueos. Aquí, se dice que el objeto ***201*** de datos está asociado con la línea ***215*** de memoria intermedia, ya que la línea ***215*** de memoria intermedia ha de contener/meter en memoria intermedia el objeto ***201*** de datos. Además, también se dice que el objeto ***201*** de datos está asociado con la ubicación ***250*** de metadatos, ya que se utiliza una dirección que referencia al objeto ***201*** de datos o a la ubicación ***250*** para indexar en la tabla ***240*** en la ubicación ***250***.

Normalmente, un valor mantenido en la ubicación ***250*** de los metadatos indica si el objeto ***201*** de datos está bloqueado o disponible. En una realización, cuando el objeto ***201*** de datos está bloqueado, la ubicación ***250*** de los metadatos incluye un primer valor para representar un estado bloqueado, tal como un estado ***252*** con propietario de lectura/escritura. No obstante, puede utilizarse y representarse cualquier bloqueo o estado de bloqueo en la ubicación ***250*** de los metadatos. Cuando está desbloqueada o disponible, la ubicación ***250*** de los metadatos incluye un segundo valor para indicar no estado desbloqueado. En una realización, el segundo valor ha de representar el número ***251*** de versión. Aquí, el número ***251*** de versión se actualiza, por ejemplo se incrementa, tras una escritura al objeto ***201*** de datos, para seguir una versión actual del objeto ***201*** de datos.

Las operaciones de lectura/carga son registradas en el registro ***265*** de lectura, mientras que las operaciones de escritura/almacenamiento son metidas en memoria tampón o registradas en el espacio ***270*** de escritura. Esta introducción en el registro/la memoria tampón se denomina a menudo operaciones barrera, ya que, a menudo, proporciona obstáculos que deben superarse para validar una operación de lectura o escritura transaccional. En una realización, el registro de una lectura incluye actualizar o crear una entrada en el registro ***265*** de lectura con representación de una dirección ***266*** asociada con el número ***251*** de versión. Aquí, el registro ***265*** de lectura puede asemejarse a una tabla de consulta basada en direcciones con la dirección ***266***, que puede ser una porción lineal, virtual, física u otra de un objeto ***201*** de datos que haga referencia a direcciones que esté asociado con el correspondiente número ***251*** de versión. Obsérvese que el registro ***265*** de lectura puede ser un objeto de datos, tal como el objeto ***201*** de datos, que ha de estar mantenido en una memoria del sistema y metido en memoria intermedia en la memoria intermedia ***205***.

En una realización, una escritura en el objeto ***201*** de datos actualiza la línea ***215*** de memoria intermedia con un nuevo valor, y el antiguo valor ***272*** es mantenido en el espacio ***270*** de escritura. Tras la confirmación de la transacción, los antiguo valores del registro ***270*** de escritura son descartados y los valores provisionales son hecho globalmente visibles y, por el contrario, tras abortar la transacción, los antiguas valores son restaurados a las ubicaciones originales escribiendo encima de los valores mantenidos provisionalmente. A menudo, este tipo de sistema de memoria transaccional de soporte lógico (STM) se denomina STM de registro de escritura o STM de actualización in situ, ya que el espacio ***270*** de escritura se asemeja a un registro de escritura para mantener los valores antiguas, mientras que los valores provisionales de la transacción son “actualizados in situ”.

En otra realización, una escritura en el objeto ***201*** de datos es introducida mediante una memoria tampón en el espacio ***270*** de escritura, que se asemeja a una memoria tampón de escritura, mientras que los valores antiguos

permanecen en sus ubicaciones originales. Aquí, la memoria tampón ***270*** de escritura mantiene un valor provisional de la transacción que ha de ser escrito a la ubicación ***215***. Al abortar la transacción, los valores provisionales mantenidos en la memoria tampón ***270*** de escritura son descartados y, por el contrario, tras confirmar la transacción, los valores provisionales son copiados a las correspondientes ubicaciones de memoria escribiendo encima de los valores antiguos. A menudo, este tipo de sistema de memoria transaccional de soporte lógico (STM) se denomina STM de memoria tampón de escritura, ya que el espacio ***270*** de escritura se asemeja a una memoria tampón de escritura, metiéndose en la memoria tampón/manteniéndose los valores provisionales de la transacción en el espacio ***270*** de escritura.

Obsérvese que el espacio ***270*** de escritura puede incluir cualquier área de almacenamiento. En una realización, el espacio ***270*** de escritura es una memoria de nivel más elevado, tal como una memoria intermedia de segundo nivel o la memoria del sistema. En otra realización, el espacio ***270*** de escritura puede ser un espacio de escritura separado mantenido en registros u otras ubicaciones de memoria. El espacio ***270*** de escritura puede asemejarse a una tabla de consulta con una dirección asociada con un valor antiguo registrado o un valor provisional puesto en memoria tampón. En otra realización adicional, el espacio ***270*** de escritura puede incluir una pila de programa o una memoria de pila separada mantenida en cualquiera de las áreas de almacenamiento mencionadas anteriormente o en un área de almacenamiento separada.

Sin embargo, con independencia de si el espacio ***270*** de escritura se utiliza como una memoria tampón de escritura para poner en memoria tampón valores provisionales o como un registro de escritura para registrar valores antiguos, la escritura, cuando es confirmada, quita el bloqueo ***250***. En una realización, la liberación del bloqueo ***250*** incluye devolver la ubicación ***250*** de metadatos a un valor que represente un estado desbloqueado o sin propietario. Este valor se obtiene incrementando un sello de tiempo global. Este sello de tiempo cuenta el número de transacciones que han acabado de ejecutarse, es decir, de confirmarse o de abortarse. Este control de versiones permite que otras transacciones validen sus lecturas que cargaron el objeto ***201*** de datos comparando los valores de versión registrados por otras transacciones en sus registros de lectura con el valor ***251*** de versión actual.

El ejemplo anterior incluye una realización de implementación de una STM; sin embargo, puede usarse cualquier implementación conocida de una STM. De hecho, también puede usarse cualquier sistema conocido para realizar una memoria transaccional, tal como una HTM, una STM, un sistema de memoria transaccional ilimitada (UTM), un sistema de memoria transaccional híbrida, tal como una STM acelerada por soporte físico (HASTM), o cualquier otro sistema de memoria transaccional. Por ejemplo, pueden utilizarse características HTM, tales como los bits de anotaciones, para acelerar una STM, tal como que la puesta a cero o a uno se base en accesos a la línea de memoria intermedia, que el soporte lógico puede interpretar y utilizar para acelerar el seguimiento transaccional o la detección de conflictos a nivel de línea de memoria intermedia.

Tal como se ha afirmado más arriba, un compilador, cuando es ejecutado para compilar código de programa o aplicación, puede insertar, en los accesos a la memoria transaccional, operaciones transaccionales, cuando es ejecutado, para llevar a cabo funciones barrera de lectura y escritura. Ejemplos de operaciones barrera de lectura incluyen: calcular un índice a la tabla de bloqueos ***240*** para determinar una ubicación ***250***, realizar una comprobación para ver si la ubicación ***250*** de metadatos contiene un valor de versión sin propietario y registrar el valor de versión en el registro ***265*** de lectura.

En una realización, se insertan operaciones de atomicidad fuerte eficiente en los accesos a la memoria no transaccional para realizar funciones barrera de lectura optimizadas para proporcionar una atomicidad fuerte, es decir, garantizar accesos válidos a la memoria no transaccional. Por ejemplo, las operaciones de atomicidad fuerte, cuando se ejecutan, han de determinar si una ubicación de memoria que ha de ser objeto de acceso por el acceso de lectura no transaccional ha sido actualizada desde el comienzo de una función. Como ilustración, se inserta una operación de atomicidad fuerte al comienzo de cada función. La operación de atomicidad fuerte, cuando se ejecuta, ha de realizar una copia local del valor de transacción global (GTV).

El GTV es incrementado en respuesta a la finalización de una transacción. En una realización, después de que el GTV es incrementado, se usa el nuevo valor GTV como un valor de versión sin propietario para quitar bloqueos. En otras palabras, un GTV sigue/indica la última transacción o la más reciente que abortó o se confirmó, lo que, a menudo, da como resultado que se haga referencia al GTV como valor de la transacción más reciente. En consecuencia, una copia local del GTV en un momento específico, como al comienzo de una función, puede ser denominada sello de tiempo, ya que proporciona una instantánea del GTV al comienzo de la función. Por lo tanto, en la realización expuesta en lo que antecede, al comienzo de una función, se realiza una copia del GTV. Esta copia local, que es denominada aquí valor de transacción local, indica la última transacción que actualizó las ubicaciones de memoria en el momento de la copia o el desplazamiento, es decir, el comienzo de la función.

En una realización, una operación de atomicidad fuerte insertada en un acceso a memoria no transaccional dentro de la función, cuando se ejecuta, ha de comparar el LTV con un valor de versión asociado con una ubicación de memoria que ha de ser objeto de acceso como consecuencia de ejecutar la operación de lectura de la memoria no transaccional. Aquí, si el valor de versión es mayor que el LTV, la ubicación de memoria ha sido actualizada desde el comienzo de la función. En otras palabras, desde que el GTV fue copiado como un LTV, ha terminado una

transacción, se ha incrementado el GTV y se ha escrito el nuevo GTV como una nueva versión a una ubicación de bloqueo asociada con la ubicación de memoria. Por lo tanto, cuando la versión es mayor que el LTV, se determina que ha ocurrido, al menos, el anterior proceso de actualización. En cambio, si el valor de versión es igual o menor que el LTV, la ubicación de memoria no ha sido actualizada desde el comienzo de la función. En una realización, la ubicación de bloqueo almacena un valor con el bit más alto puesto a uno cuando está bloqueada, mientras que todos los números de versión (y el valor en el GTV) siempre tienen el bit más alto puesto a cero. Para evitar el desbordamiento del GTV como consecuencia de muchas transacciones de confirmación, puede utilizarse un valor mayor, tal como un valor de 64 bits.

Como ilustración, supongamos que una operación de memoria no transaccional en una función incluye una operación de carga para cargar de la línea ***215*** de memoria intermedia. Al comienzo de la función, se copia un GTV, que en este ejemplo comienza con un valor decimal de 10, como un LTV en respuesta a la ejecución de una primera operación de atomicidad fuerte. Al continuar la ejecución, supongamos que una transacción actualiza la línea ***215*** de memoria intermedia durante la confirmación, incrementa el GTV a un valor decimal de 12 y quita el bloqueo ***250*** asociado con la línea ***215*** utilizando el valor incrementado de GTV de 12. Cuando se encuentra la carga no transaccional, se ejecuta una segunda operación de atomicidad fuerte para comparar el LTV con una versión actual mantenida en la ubicación ***250*** de metadatos. Dado que la ubicación ***250*** de metadatos ha sido actualizada previamente a 12 y el LTV contiene un valor de 10, se determina que la línea ***215*** ha sido actualizada desde el comienzo de la función. En consecuencia, puede emprenderse cualquier número de acciones en respuesta a la determinación de que se ha actualizado una ubicación que ha de ser objeto de acceso, tales como ejecutar una vía lenta, es decir, una barrera de atomicidad superfuerte, operaciones para garantizar la validez de la operación de carga no transaccional.

En otra realización, la operación de atomicidad fuerte insertada en un acceso a memoria no transaccional, cuando se ejecuta, ha de comparar el valor LTV con un valor de contador que indica un número de transacciones que se han iniciado. En otras palabras, aquí no se determina si la ubicación específica de memoria ha sido actualizada desde el comienzo de una función, sino, más bien, si cualquier transacción ha iniciado la actualización de alguna ubicación de memoria desde el comienzo de la función. El contador de transacciones se incrementa en respuesta a que se inicie una transacción y/o a que una transacción empiece a actualizar ubicaciones de memoria. Por ejemplo, en un STM de memoria tampón de escritura, el contador es incrementado cuando una transacción alcanza un punto de confirmación y empieza a actualizar ubicaciones de memoria. Como ejemplo adicional, en una STM de registro de escritura o de actualización in situ, el contador se incrementa cuando se inicia la transacción.

Utilizando el ejemplo de lo que antecede, al comienzo de una función que incluye la operación de carga no transaccional, se copia un GTV de 10 como un LTV, y un contador de inicio de transacciones también contiene inicialmente un valor de 10. Aquí, cuando otra transacción comienza o empieza a actualizar ubicaciones de memoria, dependiendo de la implementación, se incrementa el contador de inicio de transacciones, es decir, de un valor de 10 a 11. Sin embargo, la segunda operación de atomicidad fuerte en esta realización, cuando se ejecuta, ha de comparar el LTV con el contador de inicio de transacciones. Cuando el contador es igual o menor que el LTV, se determina que la carga no transaccional es válida, es decir, una transacción no ha actualizado ubicaciones de memoria desde el comienzo de la función.

Sin embargo, como en este ejemplo, cuando el valor del contador de inicio de transacciones, es decir, 11, es mayor que el LTV, es decir, 10, se termina que las ubicaciones de memoria han sido actualizadas desde el comienzo de la función, es decir, se ha iniciado una transacción que actualiza ubicaciones de memoria desde el comienzo de la función. Como en lo que antecede, puede emprenderse cualquier número de acciones en respuesta a la determinación de la que se han actualizado ubicaciones de memoria desde el comienzo de la función, tal como la ejecución de operaciones de vía lenta, la ejecución de un gestor y/o la utilización de otro procedimiento para garantizar la validez de los datos.

Pasando a la **Figura 3a**, se ilustra una realización de lógica para proporcionar una atomicidad fuerte eficiente. En una realización, la lógica ilustrada en la **Figura 3a** está incluida en un circuito integrado, tal como en un microprocesador. Los elementos ***301*** y ***302*** de procesamiento incluyen cualquier elemento de procesamiento mencionado en lo que antecede, tal como un hilo de soporte físico, un núcleo u otro procesador lógico/elemento de procesamiento. El elemento ***305*** de almacenamiento ha de contener un valor de transacción global (GTV). Obsérvese que el elemento de almacenamiento incluye cualquier área de almacenamiento para contener valores, tal como un registro, una ubicación de memoria, una ubicación de memoria intermedia, una ubicación de pila u otra área de almacenamiento. Por ejemplo, el elemento ***305*** de almacenamiento incluye un registro para contener un GTV. En otra realización, una ubicación de la pila de programa ha de contener el GTV.

Al comienzo de una función que ha de ser ejecutada en el PE ***301***, ha de mantenerse una copia del GTV en el elemento ***301b*** de almacenamiento como valor de transacción local (LTV) en respuesta a la ejecución de la operación ***351***. Como en lo que antecede, el elemento ***301b*** de almacenamiento puede incluir cualquier elemento/área de almacenamiento, tal como un registro, una ubicación de memoria, una ubicación de memoria intermedia, una ubicación de pila u otra área de almacenamiento. Obsérvese que se ilustra el elemento ***301b*** de almacenamiento dentro del PE ***301***; sin embargo, el elemento ***301b*** de almacenamiento puede estar asociado con el

PE ***301*** de cualquier manera. Por ejemplo, el elemento ***301b*** de almacenamiento puede ser un registro en un grupo de registros asociados con el PE ***301*** o ser una ubicación de pila de programa asociada con el elemento ***301*** de procesamiento. Aunque no se expone específicamente, el elemento ***302b*** de almacenamiento opera de una manera similar al elemento ***301b*** de almacenamiento. Según se ilustra, los elementos ***301b*** y ***302b*** están separados; sin embargo, pueden estar situados físicamente dentro del mismo dispositivo o de la misma área de almacenamiento y estar asociados en consecuencia con los PE ***301*** y ***302***. El GTV también puede ser denominado valor de transacción más reciente, es decir, indicando la última transacción que se abortó/confirmó, o sello de tiempo global, es decir, indicando una transacción o un número de transacciones confirmadas/abortadas en un punto temporal específico. Según se ilustra, esencialmente la operación ***351*** toma una instantánea del GTV contenido en el elemento ***305*** de almacenamiento al comienzo de la ejecución de una función.

La operación ***352*** incluye una operación de carga no transaccional, es decir, una operación de carga no incluida en una sección crítica de una transacción, cuando se ejecuta, para carga de la línea ***310*** de memoria intermedia. El elemento ***302*** de procesamiento ejecuta la operación transaccional ***353***, tal como una operación de confirmación, una operación de aborto y/o una operación de almacenamiento, lo que da como resultado la actualización de la línea ***310*** de memoria intermedia. En una realización, al confirmarse o abortar la transacción, el GTV contenido en el elemento ***305*** de almacenamiento es actualizado/incrementado. El GTV recién incrementado puede ser utilizado como número de versión para actualizar la ubicación ***315*** de metadatos, que está asociada con la línea ***310*** de memoria intermedia.

Obsérvese, a partir de lo anterior, por ejemplo, que la ubicación ***315*** de metadatos puede estar asociada con la línea ***310*** de memoria intermedia a través de una función de clave calculada de al menos una porción de una dirección asociada con la línea ***310*** para indexar en la ubicación ***315*** de metadatos dentro de una tabla de ubicaciones de metadatos. Además, en una realización, un número de versión en la ubicación ***315*** indica que la línea ***310*** de memoria intermedia está desbloqueada/sin propietario. Alternativamente, la línea ***310*** de memoria intermedia es bloqueada en respuesta a que en la ubicación ***315*** se contenga un valor de bloqueo/con propietario. Como ejemplo, se indica un valor de bloqueo/con propietario en la ubicación ***315*** porque se contenga un uno lógico en la posición del bit más significativo (MSB) de la ubicación ***315***, mientras que se indica un número de versión porque se contenga un cero lógico en la posición MSB de la ubicación ***315***.

La lógica ***320*** de comparación, en la operación ***354***, compara el valor de bloqueo contenido en la ubicación ***315*** de metadatos con la copia del GTV ***305*** desde el comienzo de la función contenida en el área ***310b*** de almacenamiento del LTV. Si el valor contenido en la ubicación ***315*** es menor o igual que el LTV en ***301b***, indicando que la versión contenida en la ubicación ***315*** y, por asociación, la ubicación ***310*** de memoria no han sido actualizadas desde que el GTV ***305*** fue copiado al elemento ***301b*** de almacenamiento del LTV, entonces la ejecución continúa normalmente. Cuando se determina que el acceso es válido, el flujo de ejecución continúa sin la ejecución de operaciones barrera extrínsecas adicionales.

En cambio, si el valor contenido en la ubicación ***315*** de metadatos es mayor que el LTV contenido en el elemento ***301b*** de almacenamiento, entonces se determina que la ubicación ***310*** de memoria ha sido actualizada desde que el GTV fue puesto en memoria intermedia en el elemento ***301b*** de almacenamiento del LTV. Obsérvese que, en la realización en la que un valor de bloqueo incluye un uno lógico contenido en la posición MSB de la ubicación ***315***, el valor de bloqueo será mayor que cualquier GTV utilizado como número de versión para actualizar la ubicación ***315***, ya que un número de versión incluye un cero en la posición MSB. Por lo tanto, en una realización, se determina que la ubicación ***310*** ha sido actualizada desde el momento que la ubicación ***315*** contiene un valor de bloqueo hasta el momento en que la ubicación ***315*** de metadatos es actualizada a un valor de versión mayor.

En respuesta a que el valor de bloqueo contenido en la ubicación ***315*** sea mayor que un LTV contenido en el elemento ***301b*** de almacenamiento, puede emprenderse cualquier número de acciones de resolución de atomicidad fuerte. Por ejemplo, puede ejecutarse un grupo de operaciones de vía lenta para garantizar un acceso válido a la memoria no transaccional. Las operaciones de vía lenta pueden incluir otras operaciones de mayor cautela, tales como la comprobación/verificación de la disponibilidad de bloqueo antes de llevar a cabo el acceso. Como ejemplo adicional, pueden utilizarse otros algoritmos de resolución de disputas, tales como la ejecución de un gestor o abortar una transacción que disputa la ubicación de memoria.

Como ejemplo ilustrativo sumamente simplificado para demostrar una realización de la operación, el GTV ***305*** contiene inicialmente un GTV de 0001 (valor decimal de uno) y la ubicación ***315*** de metadatos también incluye una versión de 0001. Al comienzo de una función, el valor GTV de 0001 es copiado al elemento ***301b*** de almacenamiento del LTV. Se lleva a cabo una carga no transaccional desde la línea ***310***. Además, se confirma una transacción que está siendo ejecutada por el PE ***302***, lo que da como resultado que el GTV ***305*** se incremente a 0010 (un valor decimal de dos). Durante la ejecución de la transacción, la ubicación ***315*** pasa a un valor de bloque, tal como 1000 (valor decimal de ocho) y se actualiza la línea ***310***. Además, después de la actualización, cuando se quita el bloqueo, se almacena el numero valor GTV de 0010 como una nueva versión en la ubicación ***315*** de metadatos. Cuando la lógica ***320*** de comparación compara el valor de bloqueo contenido en la ubicación ***315*** de metadatos con el LTV ***301b***, el valor de bloqueo es mayor que el LTV ***301b***, tanto cuando se mantiene el bloqueo, es

decir, 1000 > 0001, como después de que se actualiza la versión, es decir, 0010 > 0001. En consecuencia puede llevarse a cabo una resolución de atomicidad fuerte, tal como la ejecución de operaciones de vía lenta.

Pasando a la **Figura 3b**, se ilustra otra realización de un módulo para proporcionar una atomicidad fuerte eficiente. Aquí, las operaciones ***351-353*** operan de manera similar a lo expuesto con referencia a la **Figura 3a**. Sin embargo, en la operación ***354***, la lógica ***320*** de comparación compara el LTV ***301b*** con el valor de recuento de transacciones contenido en el contador ***330*** de inicio de transacciones. En una realización, el contador ***330*** de transacciones se incrementa en respuesta al inicio de una transacción. En otra realización, el contador ***330*** de transacciones se incrementa en respuesta a que una transacción empiece a actualizar ubicaciones de memoria. En una STM de actualización in situ, las actualizaciones ocurren/comienzan cuando se llevan a cabo los almacenamientos transaccionales. Por lo tanto, el contador ***330*** se incrementaría en ese punto. Sin embargo, en una STM de memoria tampón de escritura, el contador ***330*** se incrementa cuando se confirma la transacción, es decir, cuando comienzan las actualizaciones de la memoria.

Si el contador ***330*** es menor o igual al LTV ***301b***, ninguna transacción ha empezado a actualizar ubicaciones de memoria y la ejecución puede continuar normalmente. Sin embargo, si el contador ***330*** es mayor que LTV ***301b***, una transacción ha empezado a actualizar ubicaciones de memoria. En consecuencia, pueden ejecutarse otras operaciones de vía más lenta. Por ejemplo, al comparar LTV y un valor de contador, otras operaciones pueden ser optimizadas o evitadas, tal como calcular una ubicación de memoria objeto de acceso, un valor de clave calculada a una ubicación de metadatos, y un valor de versión contenido en la ubicación de metadatos. Por lo tanto, si el valor del contador es mayor que el LTV, que indica que las ubicaciones de memoria han sido actualizadas, entonces una versión puede ser objeto de comparación para determinar si la ubicación de memoria objeto de acceso ha sido actualizada. Además, pueden ejecutarse cualesquiera otras instrucciones u operaciones de vía lenta.

Como ejemplo ilustrativo sumamente simplificado para demostrar una realización de la operación, el GTV ***305*** contiene inicialmente un GTV de 0001 (valor decimal de uno) y el contador ***330*** incluye un valor de transacción de 0001. Al comienzo de una función, el valor GTV de 0001 es copiado al elemento ***301b*** de almacenamiento del LTV. Se lleva a cabo una carga no transaccional desde la línea ***310***. Además, se confirma una transacción que está siendo ejecutada por el PE ***302*** en una STM de memoria tampón de escritura, lo que da como resultado que el contador ***330*** se incremente a 0010. Cuando se comparan el valor de la transacción de 0010 y un valor del LTV de 0001, el valor de la transacción es mayor, lo que indica que una transacción ha actualizado ubicaciones de memoria. Obsérvese que, aquí, la transacción puede no actualizar la ubicación ***310*** de memoria. Por lo tanto, la lógica ***320*** de comparación puede comparar condicionalmente la versión con el LTV, tal como se expuso en lo que antecede, para determinar si la transacción ha actualizado la ubicación ***310*** de memoria específicamente.

Con referencia a continuación a la **Figura 4a**, se ilustra una realización de un diagrama de flujo de un procedimiento para proporcionar una atomicidad fuerte eficiente. Además, tal como se ha expuesto en lo que antecede, un compilador, cuando se ejecuta, ha de compilar código de aplicaciones/programas. Durante la compilación, pueden insertarse operaciones transaccionales, tales como barreras de lectura, barreras de escritura, operaciones de confirmación y operaciones de aborto. Además, pueden insertarse operaciones de atomicidad fuerte no transaccionales que, cuando se ejecuten, han de llevar a cabo las operaciones/funciones descritas en lo que antecede y/o en lo que sigue con referencia a las **Figuras 4a-4c**.

En el flujo ***405***, se copia un valor de transacción global (GTV) a un valor de transacción local (LTV) al inicio de una función. En una realización, el GTV es un sello de tiempo global incrementado/actualizado en respuesta a la confirmación o el aborto de una transacción. Por ejemplo, se inserta al comienzo de una función no transaccional una operación de atomicidad fuerte para que, cuando se ejecute, lleve a cabo la copia o el desplazamiento.

A continuación, en el flujo ***410***, se lleva a cabo una operación de carga no transaccional, que está incluida dentro de la función, para cargar desde una ubicación de memoria. En el flujo ***415***, se determina si una transacción ha actualizado ubicaciones de memoria desde el comienzo de la función. Si una transacción no ha actualizado ubicaciones de memoria desde el comienzo de la función, se determina que la carga es válida y la ejecución continúa normalmente en el flujo ***425***. En cambio, si la transacción ha actualizado ubicaciones de memoria desde el comienzo de la función, entonces en la flujo ***420*** se llevan a cabo operaciones de vía lenta. Obsérvese que las operaciones de vía lenta pueden incluir cualesquiera operaciones de resolución para garantizar la validez de la carga no transaccional, incluyendo la comprobación, la verificación y/o la entrada en bucle/espera en un bloqueo, el cálculo de una versión, la comprobación de que no ha cambiado una versión y/o llevar a cabo nuevamente la carga no transaccional.

La **Figura 4b** ilustra una realización de un diagrama de flujo para determinar si una transacción ha actualizado ubicaciones de memoria desde el comienzo de la función de la **Figura 4a**. Los flujos ***405, 410, 420*** y ***425*** son similares a los mismos flujos de la **Figura 4a**. Sin embargo, en el flujo ***416***, se determina un valor de registro de la transacción, que está asociado con la ubicación de memoria desde la que se carga. En una realización, se crea una clave calculada de al menos una porción de una dirección que hace referencia a la ubicación de memoria y se la introduce en una tabla de registros de transacción. Se determina el valor del registro de la transacción a partir del registro de la transacción asociado con la ubicación de memoria. En el flujo ***417***, se compara el valor del registro de

la transacción con el valor de transacción local copiado del GTV en el flujo ***405***. En una realización se incrementa el valor de transacción global (GTV) en respuesta a que se confirme/aborte una transacción. Además, se actualiza el valor del registro de la transacción con el GTV recién incrementado para quitar un bloqueo en la ubicación de memoria propiedad de la transacción que se está confirmando.

Por lo tanto, si un valor de registro de la transacción es menor o igual al LTV, se determina que la ubicación de memoria no ha sido actualizada desde el comienzo de la función y la ejecución continúa normalmente en el flujo ***425***. Sin embargo, si el valor del registro de la transacción es mayor que el LTV, se determina que la ubicación de memoria ha sido actualizada desde que el GTV fue puesto en memoria intermedia como el LTV, es decir, desde el comienzo de la función. Aquí se ejecutan instrucciones/operaciones de vía lenta para garantizar la validez de la carga no transaccional, tales como comprobar el registro de la transacción, entrar en bucle/espera hasta que el registro de la transacción se desbloquee o quede sin propietario, adquirir la propiedad del registro de la transacción y/o realizar de nuevo el acceso no transaccional.

Pasando a la **Figura 4c**, se ilustra otra realización de un diagrama de flujo de un procedimiento para determinar las ubicaciones de memoria han sido actualizadas desde el comienzo de una función. Aquí, se incrementa un contador de inicio global cuando una transacción empieza a actualizar ubicaciones de memoria en el flujo ***412***. En una STM de memoria tampón de escritura, las ubicaciones son actualizadas en la confirmación, de modo que el contador de inicio de transacciones se incrementaría al confirmarse una transacción. Sin embargo, en una STM de actualización in situ, las ubicaciones de memoria son actualizadas provisionalmente durante la ejecución de la transacción, de modo que el contador puede incrementarse en respuesta al inicio de la transacción o cuando se encuentra el primer almacenamiento transaccional. Obsérvese que el contador puede incrementarse en un entero o en múltiplos del mismo. Por ejemplo, si un número de versión incluye números pares y un valor con propietario incluye números impares, el contador puede incrementarse en múltiplos pares para que coincida con versiones o un GTV.

En el flujo ***418***, se compara el valor del contador de inicio con el valor de transacción local (LTV). Por lo tanto, se determina si alguna transacción ha empezado a actualizar ubicaciones de memoria desde que el LTV fue actualizado por última vez, es decir, el comienzo de la función. Obsérvese que la ubicación específica de memoria puede no haberse actualizado; sin embargo, una transacción ha empezado a actualizar ubicaciones de memoria. Aquí, en la ejecución de una operación de vía lenta en el flujo ***420***, puede compararse una versión, de manera similar al procedimiento ilustrado en la **Figura 4b**, para determinar si la ubicación específica de memoria ha sido actualizada. Sin embargo, pueden ejecutarse de manera alternativa o adicional otras operaciones para garantizar la validez de la carga no transaccional.

Tal como se ilustra en lo que antecede, pueden insertarse operaciones optimizadas de atomicidad fuerte, cuando se ejecutan, para proporcionar atomicidad fuerte eficiente para accesos de lectura no transaccional. Potencialmente, pueden evitarse las operaciones barrera exhaustivas para cada carga no transaccional para acelerar la ejecución de las cargas no transaccionales en un sistema de memoria transaccional. Además, pueden utilizarse combinaciones de niveles de ejecución intensa. Por ejemplo, pueden llevarse a cabo accesos no transaccionales con solo determinar si se ha iniciado otra transacción, sin tener que calcular un valor de versión. Si ninguna transacción ha empezado a actualizar ubicaciones de memoria, la ejecución puede continuar sin tener que llevar a cabo una contabilidad exhaustiva. Sin embargo, si se han actualizado ubicaciones de memoria, una versión puede ser calculada y comparada con un sello de tiempo local para determinar si se ha actualizado una ubicación específica de memoria. En consecuencia, la ejecución puede ser acelerada un tanto evitando comprobar/entrar en un bucle en un bloqueo. Además, si la ubicación de memoria ha sido actualizada, pueden ejecutarse operaciones previas completas de atomicidad fuerte, es decir, una vía más lenta de instrucciones, para garantizar la validez de los datos entre accesos a memoria transaccional y no transaccional.

Las realizaciones de los procedimientos, el soporte lógico, el soporte físico o el código expuestas en lo que antecede pueden ser implementadas mediante instrucciones o código almacenados en un medio accesible por máquina o legible por máquina que sean ejecutable por un elemento de proceso. Un medio accesible/legible por máquina incluye cualquier mecanismo que proporciones (por ejemplo, almacene y/o transmita) información en una forma legible por una máquina, tal como un ordenador o un sistema electrónico. Por ejemplo, un medio accesible por máquina incluye memoria de acceso aleatorio (RAM), tal como RAM estática (SRAM) o RAM dinámica (DRAM); ROM; un medio de almacenamiento magnético u óptico; dispositivos de memoria flash; señales eléctricas, ópticas, acústicas u otra forma de señales propagadas (por ejemplo, ondas portadoras, señales infrarrojas, señales digitales); etc.

La referencia en toda la presente memoria a “una realización” significa que se incluye un rasgo, una estructura o una característica particulares descritos en conexión con la realización en al menos una realización de la presente invención. Así, las apariciones de la expresión “en una realización” en diversos lugares de toda la presente memoria no se refieren todas necesariamente a la misma realización. Además, los rasgos, las estructuras o las características particulares pueden combinarse de cualquier manera adecuada en una o más realizaciones.

Otros aspectos y otras características de esta divulgación se presentan en las cláusulas siguientes:

1. Un aparato que comprende:

un elemento de proceso para ejecutar una pluralidad de operaciones transaccionales y una pluralidad de operaciones no transaccionales, habiendo de ejecutar el elemento de proceso una operación de carga no transaccional de la pluralidad de operaciones no transaccionales dentro de una función, siendo la operación de carga no transaccional, cuando se ejecuta, para cargar desde una ubicación de memoria; y

lógica asociada con el elemento de proceso para determinar si la ubicación de memoria ha sido actualizada desde el comienzo de la función;

en el que el elemento de proceso no ha de ejecutar una operación de bloqueo en respuesta a la determinación de que la ubicación de memoria no ha sido actualizada desde el comienzo de la función, habiendo de determinar la operación de bloqueo, cuando se ejecute, si una ubicación de metadatos asociada con la ubicación de memoria contiene un valor desbloqueado.

2. El aparato de la cláusula 1 que, además, comprende un área de almacenamiento asociada con el elemento de proceso para ser actualizada al comienzo de la función para contener una copia local de un valor de la transacción más reciente (MRTV), en el que el MRTV, que ha de mantenerse en una segunda área de almacenamiento, ha de ser actualizado en respuesta a la retirada de una operación de confirmación o una de aborto de la pluralidad de operaciones transaccionales.

3. El aparato de la cláusula 2 en el que la lógica asociada con el elemento de proceso para determinar si la ubicación de memoria ha sido actualizada desde el comienzo de la función comprende: lógica de comparación para comparar un valor de versión que ha de estar contenido en la ubicación de metadatos con la copia local del MRTV, habiendo de determinar la lógica de comparación que la ubicación de memoria ha sido actualizada desde el comienzo de la función en respuesta a que el valor de versión sea mayor que la copia local del MRTV.

4. El aparato de la cláusula 3 en el que el área de almacenamiento y la segunda área de almacenamiento se seleccionan cada una independientemente de un grupo que consiste en un registro, una ubicación en una memoria intermedia, una ubicación en memoria y una ubicación en una pila de programa.

5. El aparato de la cláusula 3 en el que la ubicación de metadatos asociada con la ubicación de memoria ha de contener el valor de versión, habiendo de actualizarse la ubicación de metadatos al valor de versión a partir de un valor de bloqueo en respuesta a que el elemento de proceso ejecute una operación de actualización para actualizar un valor contenido en la ubicación de memoria.

6. El aparato de la cláusula 5 en el que una celda de almacenamiento más significativa (MSB) de la ubicación de metadatos ha de contener un uno lógico para representar el valor de bloqueo y en el que el MSB de la ubicación de metadatos ha de contener un cero lógico para representar el valor de versión.

7. El aparato de la cláusula 2 que, además, comprende un contador acoplado con la lógica que ha de ser implementada en respuesta al inicio de una transacción, en el que la lógica asociada con el elemento de proceso para determinar si la ubicación de memoria ha sido actualizada desde el comienzo de la función comprende lógica de comparación para comparar la copia local del MRTV con el valor actual contenido por el contador, en el que se determina que la ubicación de memoria no ha sido actualizada desde el comienzo de la función en respuesta a que la copia local del MRTV sea igual al valor actual contenido por el contador.

8. El aparato de la cláusula 1 en el que el elemento de proceso ha de ejecutar una pluralidad de operaciones de vía lenta de la pluralidad de operaciones no transaccionales, incluyendo la pluralidad de operaciones de vía lenta la operación de bloqueo en respuesta a que el valor de versión sea mayor que la copia local del MRTV, y en el que el elemento de proceso no ha de ejecutar la pluralidad de operaciones de vía lenta en respuesta a que el valor de versión sea menor o igual a la copia local del MRTV.

REIVINDICACIONES

1. Un procedimiento para proporcionar mecanismos para una atomicidad fuerte en un sistema de memoria transaccional
caracterizado por:

detectar una operación de carga no transaccional en una función, siendo la operación de carga no transaccional, cuando se ejecuta, para cargar desde una ubicación de memoria;
insertar en la función una pluralidad de operaciones de atomicidad fuerte en respuesta a la detección de la operación de carga no transaccional en la función, siendo las operaciones de atomicidad fuerte para garantizar la validez entre accesos a memoria transaccionales y no transaccionales;

en el que las operaciones de atomicidad fuerte, cuando son ejecutadas, garantizan la validez determinando (415) si la ubicación (215) de memoria o las ubicaciones (205) de memoria han sido actualizadas por una transacción, habiendo tenido lugar dicha actualización desde el inicio de la función, en el que dicha inserción comprende:

insertar una primera operación de atomicidad fuerte en la función que, cuando es ejecutada, actualiza un registro con una copia (301b, 302b) de valor de transacción local LTV de un valor de transacción global (305), siendo dicho valor de transacción global un valor incrementado en respuesta a la finalización de una transacción y ocurriendo dicha actualización del registro al inicio de dicha función;
insertar una segunda operación de atomicidad fuerte que, cuando es ejecutada, obtiene un valor de versión asociado con la ubicación (215) de memoria; e
insertar una tercera operación de atomicidad fuerte que, cuando es ejecutada, compara el valor de versión con la copia (301b, 302b) de LTV para determinar si la ubicación (215) de memoria ha sido actualizada desde el inicio de la función;

y en el que el procedimiento comprende, además, insertar (420) una pluralidad de operaciones de vía lenta que han de ser ejecutadas en respuesta a la determinación (415) de que la ubicación (215) de memoria o las ubicaciones (205) de memoria han sido actualizadas desde el inicio de la función, para resolver el conflicto entre la transacción y la operación de carga no transaccional, en el que dichas operaciones de vía lenta incluyen:

entrar en bucle o esperar en un bloqueo asociado con el valor de versión,
adquirir la propiedad de los bloqueos y/o
volver a llevar a cabo la operación de carga no transaccional.

2. El procedimiento de la reivindicación 1 en el que la pluralidad de operaciones de atomicidad fuerte no incluye una operación de bloqueo que, cuando se ejecute, determine si un bloqueo (250) asociado con la ubicación (215) de memoria no tiene propietario.

3. El procedimiento de la reivindicación 1 en el que la determinación de si se han iniciado transacciones desde el inicio de la función para actualizar ubicaciones (205) de memoria que son objeto de acceso por la operación de carga no transaccional comprende:

insertar una primera operación alternativa de atomicidad fuerte en la función que, cuando es ejecutada, ha de copiar un valor (305) de transacción global GTV a un valor (301b, 302b) de transacción local LTV; e
insertar una segunda operación alternativa de atomicidad fuerte que, cuando es ejecutada, ha de comparar el LTV (301b, 302b) con un valor (330) de transacción de inicio STV para determinar si se han iniciado transacciones que actualizan ubicaciones (205) de memoria desde el inicio de la función.

4. El procedimiento de las reivindicaciones 1 o 3 en el que la primera operación de atomicidad fuerte o la primera operación alternativa de atomicidad fuerte, respectivamente, cuando son ejecutadas, han de actualizar también un registro para contener el LTV (301b, 302b).

5. El procedimiento de la reivindicación 1 en el que la segunda operación de atomicidad fuerte, que, cuando es ejecutada, obtiene el valor (251) de versión asociado con la ubicación (215) de memoria, comprende:

una primera operación de versión que, cuando es ejecutada, obtiene una dirección asociada con la ubicación (215) de memoria, y
una segunda operación de versión que, cuando es ejecutada, calcula un índice a un registro (250) de transacciones en una tabla (240) de registros de transacciones para obtener el valor (251) de versión.

6. El procedimiento de la reivindicación 1 en el que el registro (250) de transacciones ha de contener un valor (252) de bloqueo para indicar que la ubicación (215) de memoria tiene propietario y de contener el valor (251) de versión para indicar que la ubicación (215) de memoria no tiene propietario, y en el que el valor (252) de bloqueo incluye un uno lógico en un bit más significativo MSB del registro (250) de transacciones y el valor (251) de versión incluye un cero lógico en el MSB del registro (250) de transacciones.

7. El procedimiento de la reivindicación 1 en el que la tercera operación de atomicidad fuerte, que, cuando es ejecutada, compara el valor de versión con la copia (301b, 302b) del LTV para determinar si la ubicación (215) de memoria ha sido actualizada desde el inicio de la función, comprende:

que la tercera operación de atomicidad fuerte, cuando sea ejecutada, determine (415) que la ubicación (215) de memoria ha sido actualizada desde el inicio de la función en respuesta a que el valor (251) de versión sea mayor que la copia (301b, 302b) del LTV, y
determinar (415) que la ubicación (215) de memoria no ha sido actualizada desde el inicio de la función en respuesta a que el valor (251) de versión sea menor o igual a la copia (301b, 302b) del LTV.

8. El procedimiento de la reivindicación 7 en el que el valor (251) de versión es mantenido en una ubicación (240) de los metadatos que es objeto de referencia por un valor de clave calculado de al menos una porción de una dirección asociada con la ubicación (215) de memoria.

9. El procedimiento de la reivindicación 3 en el que la determinación de si se han iniciado transacciones desde el inicio de la función para actualizar ubicaciones (205) de memoria objeto de acceso por la operación de carga no transaccional comprende, además:

que el STV contenga un valor de recuento;
actualizar el valor de recuento en respuesta al inicio de cualquier transacción;
comparar el LTV (301b, 302b) con el valor de recuento;
determinar (415) que las transacciones no han actualizado la pluralidad de ubicaciones (205) de memoria desde el inicio de la función en respuesta a que el valor de recuento sea igual al LTV (301b, 302b); y
determinar (415) que las transacciones han actualizado la pluralidad de ubicaciones (205) de memoria desde el inicio de la función en respuesta a que el valor de recuento sea mayor que el LTV (301b, 302b).

10. Un sistema que comprende una memoria transaccional con mecanismos para una atomicidad fuerte **caracterizado porque** el sistema incluye un código de programa que proporciona:

un medio para detectar una operación de carga no transaccional en una función, siendo la operación de carga no transaccional, cuando se ejecuta, para cargar desde una ubicación de memoria;
un medio para insertar una pluralidad de operaciones de atomicidad fuerte en respuesta a la detección de la operación de carga no transaccional en la función;
un medio para permitir las operaciones de atomicidad fuerte para garantizar la validez entre los accesos a memoria transaccionales y no transaccionales;
en el que el medio que permite las operaciones de atomicidad fuerte para garantizar la validez comprende un medio para determinar (415) si la ubicación (215) de memoria o las ubicaciones (205) de memoria han sido actualizadas por una transacción, habiendo tenido lugar dicha actualización desde el inicio de la función, en el que el medio para dicha determinación comprende: un medio para insertar una primera operación de atomicidad fuerte en la función que, cuando es ejecutada, actualiza un registro con una copia (301b, 302b) de valor de transacción local LTV de un valor de transacción global (305), siendo dicho valor de transacción global un valor incrementado en respuesta a la finalización de una transacción y ocurriendo dicha actualización del registro al inicio de dicha función;
un medio para insertar una segunda operación de atomicidad fuerte que, cuando es ejecutada, obtiene un valor de versión asociado con la ubicación (215) de memoria; y
un medio para insertar una tercera operación de atomicidad fuerte que, cuando es ejecutada, compara el valor de versión con la copia (301b, 302b) de LTV para determinar si la ubicación (215) de memoria ha sido actualizada desde el inicio de la función;
y en el que el código de programa, además, proporciona: un medio para insertar (420) una pluralidad de operaciones de vía lenta que han de ser ejecutadas en respuesta a la determinación (415) de que la ubicación (215) de memoria o las ubicaciones (205) de memoria han sido actualizadas desde el inicio de la función, para resolver el conflicto entre la transacción y la operación de carga no transaccional, en el que dichas operaciones de vía lenta incluyen:
un medio para permitir entrar en bucle o esperar en un bloqueo asociado con el valor de versión,
un medio para adquirir la propiedad de los bloqueos y/o
un medio para volver a llevar a cabo la operación de carga no transaccional.

11. El sistema de la reivindicación 10 en el que la pluralidad de operaciones de atomicidad fuerte no incluye una operación de bloqueo que, cuando se ejecute, determine si un bloqueo (250) asociado con la ubicación (215) de memoria no tiene propietario.

12. El sistema de la reivindicación 10 en el que el medio para determinar si se han iniciado transacciones desde el inicio de la función para actualizar ubicaciones (205) de memoria que son objeto de acceso por la operación de carga no transaccional comprende:

un medio para insertar una primera operación alternativa de atomicidad fuerte en la función que, cuando es ejecutada, efectúa una copia de un valor (305) de transacción global GTV a un valor (301b, 302b) de transacción local LTV; y
un medio para insertar una segunda operación alternativa de atomicidad fuerte que, cuando es ejecutada, ha de comparar el LTV (301b, 302b) con un valor (330) de transacción de inicio STV para determinar si se han iniciado transacciones que actualizan ubicaciones (205) de memoria desde el inicio de la función.

13. El sistema de las reivindicaciones 10 o 12 en el que la primera operación de atomicidad fuerte o la primera operación alternativa de atomicidad fuerte, cuando son ejecutadas, han de actualizar también un registro para contener el LTV (301b, 302b).

14. El sistema de la reivindicación 10 en el que la segunda operación de atomicidad fuerte, que, cuando es ejecutada, obtiene el valor (251) de versión asociado con la ubicación (215) de memoria, comprende:

una primera operación de versión que, cuando es ejecutada, obtiene una dirección asociada con la ubicación (215) de memoria, y
una segunda operación de versión que, cuando es ejecutada, calcula un índice a un registro (250) de transacciones en una tabla (240) de registros de transacciones para obtener el valor (251) de versión.

15. El sistema de la reivindicación 10 en el que el registro (250) de transacciones está configurado para contener un valor (252) de bloqueo para indicar que la ubicación (215) de memoria tiene propietario y está configurado, además, para contener el valor (251) de versión para indicar que la ubicación (215) de memoria no tiene propietario, y en el que el valor (252) de bloqueo comprende un uno lógico en un bit más significativo MSB del registro (250) de transacciones y el valor (251) de versión comprende un cero lógico en el MSB del registro (250) de transacciones.

16. El sistema de la reivindicación 10 en el que la tercera operación de atomicidad fuerte, que, cuando es ejecutada, compara el valor de versión con la copia (301b, 302b) del LTV para determinar si la ubicación (215) de memoria ha sido actualizada desde el inicio de la función:

está configurada para determinar (415) que la ubicación (215) de memoria ha sido actualizada desde el inicio de la función en respuesta a que el valor (251) de versión sea mayor que la copia (301b, 302b) del LTV, y
está configurada para determinar (415) que la ubicación (215) de memoria no ha sido actualizada desde el inicio de la función en respuesta a que el valor (251) de versión sea menor o igual a la copia (301b, 302b) del LTV.

17. El sistema de la reivindicación 16 que comprende una ubicación (240) en los metadatos en la que se mantiene el valor (251) de versión, estando configurada la ubicación (240) en los metadatos para ser objeto de referencia por un valor de clave calculado de al menos una porción de una dirección asociada con la ubicación (215) de memoria.

18. El sistema de la reivindicación 10 en el que el medio para determinar si ubicaciones (205) de memoria han sido actualizadas por una transacción comprende:

un medio para permitir que el STV contenga un valor de recuento;
un medio para actualizar el valor de recuento en respuesta al inicio de una transacción;
un medio para comparar el LTV (301b, 302b) con el valor de recuento;
un medio para determinar (415) que la transacción no ha actualizado las ubicaciones (205) de memoria desde el inicio de la función en respuesta a que el valor de recuento sea igual al LTV (301b, 302b); y
un medio para determinar (415) que la transacción ha actualizado las ubicaciones (205) de memoria desde el inicio de la función en respuesta a que el valor de recuento sea mayor que el LTV (301b, 302b).

19. Un soporte que porta un programa de ordenador que comprende una pluralidad de instrucciones implementables por procesador para hacer que un procesador lleve a cabo un procedimiento según cualquiera de las reivindicaciones 1 a 9.

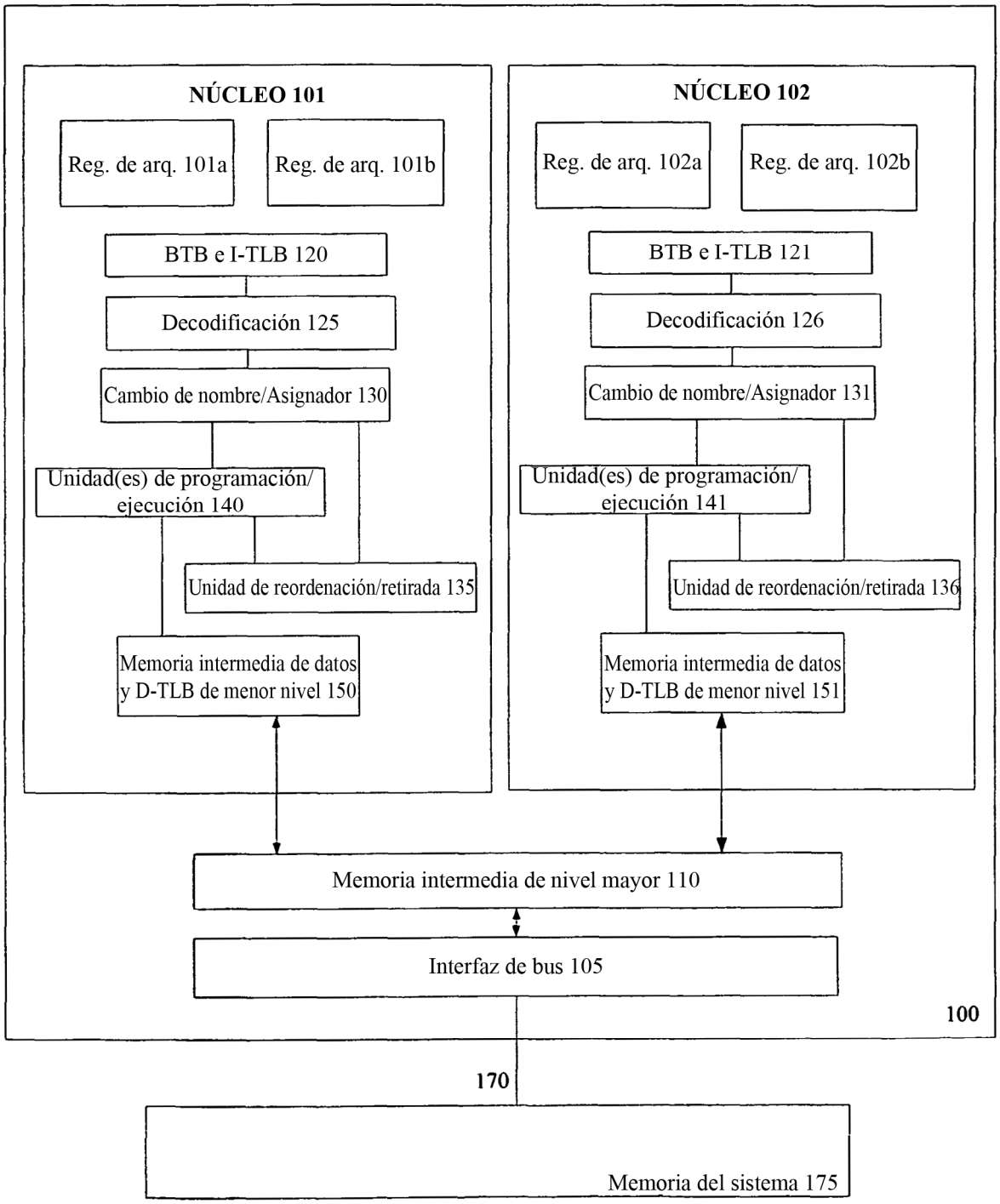
NÚCLEO 101
Reg. de arq. 101a
Reg. de arq. 101b
BTB e I-TLB 120
Decodificación 125
Cambio de nombre/Asignador 130
Unidad(es) de programación/ ejecución 140
Unidad de reordenación/retirada 135
Memoria intermedia de datos y D-TLB de menor nivel 150
NÚCLEO 102
Reg. de arq. 102a
Reg. de arq. 102b
BTB e I-TLB 121
Decodificación 126
Cambio de nombre/Asignador 131
Unidad(es) de programación/ ejecución 141
Unidad de reordenación/retirada 136
Memoria intermedia de datos y D-TLB de menor nivel 151
Memoria intermedia de nivel mayor 110
Interfaz de bus 105
100
170
Memoria del sistema 175

FIG. 1

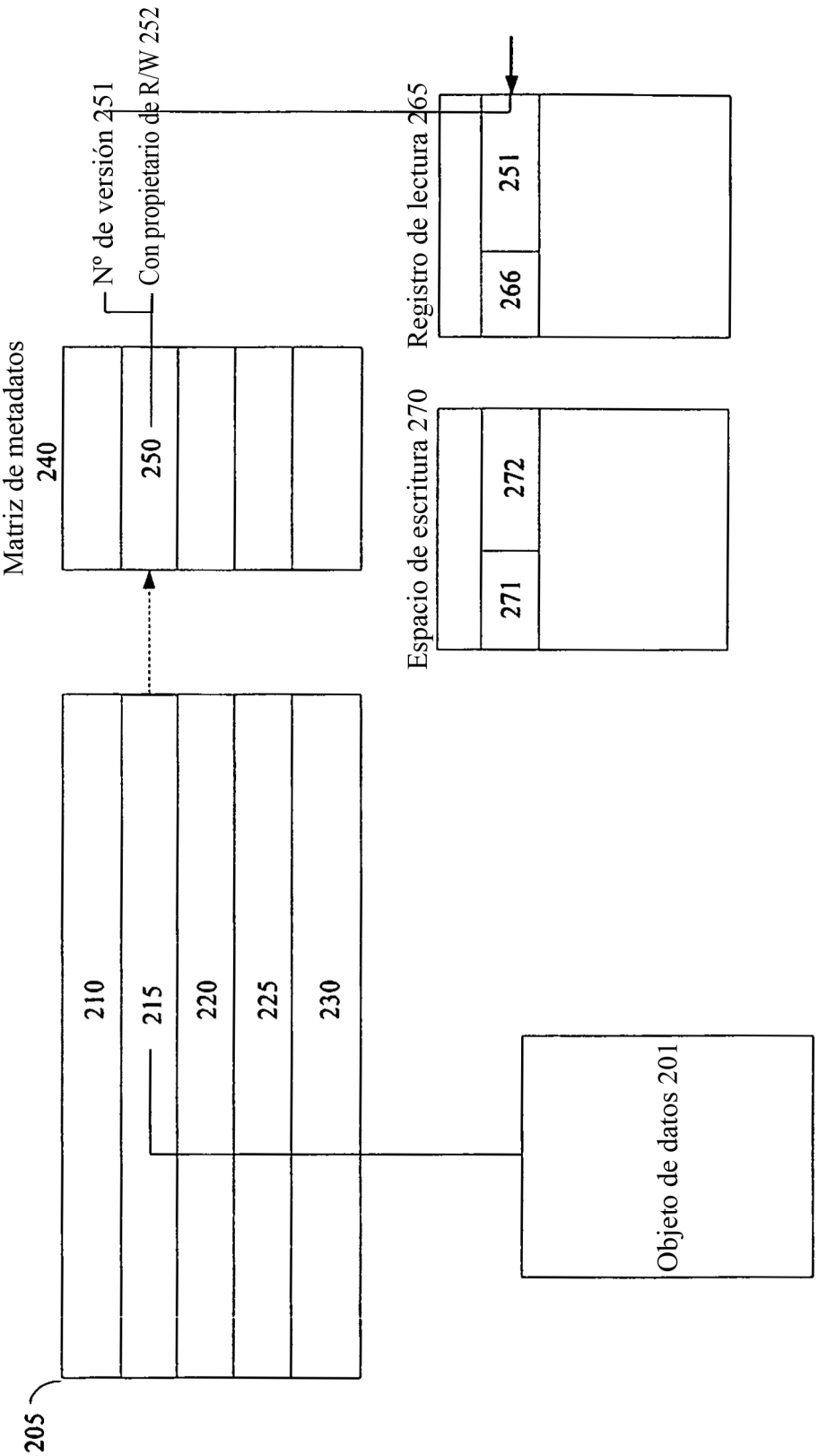
205
210
215
220
225
230
Objeto de datos 201
Matriz de metadatos
240
250
Nº de versión 251
Con propietario de R/W 252
Registro de lectura 265
251
266
Espacio de escritura 270
271
272

FIG. 2

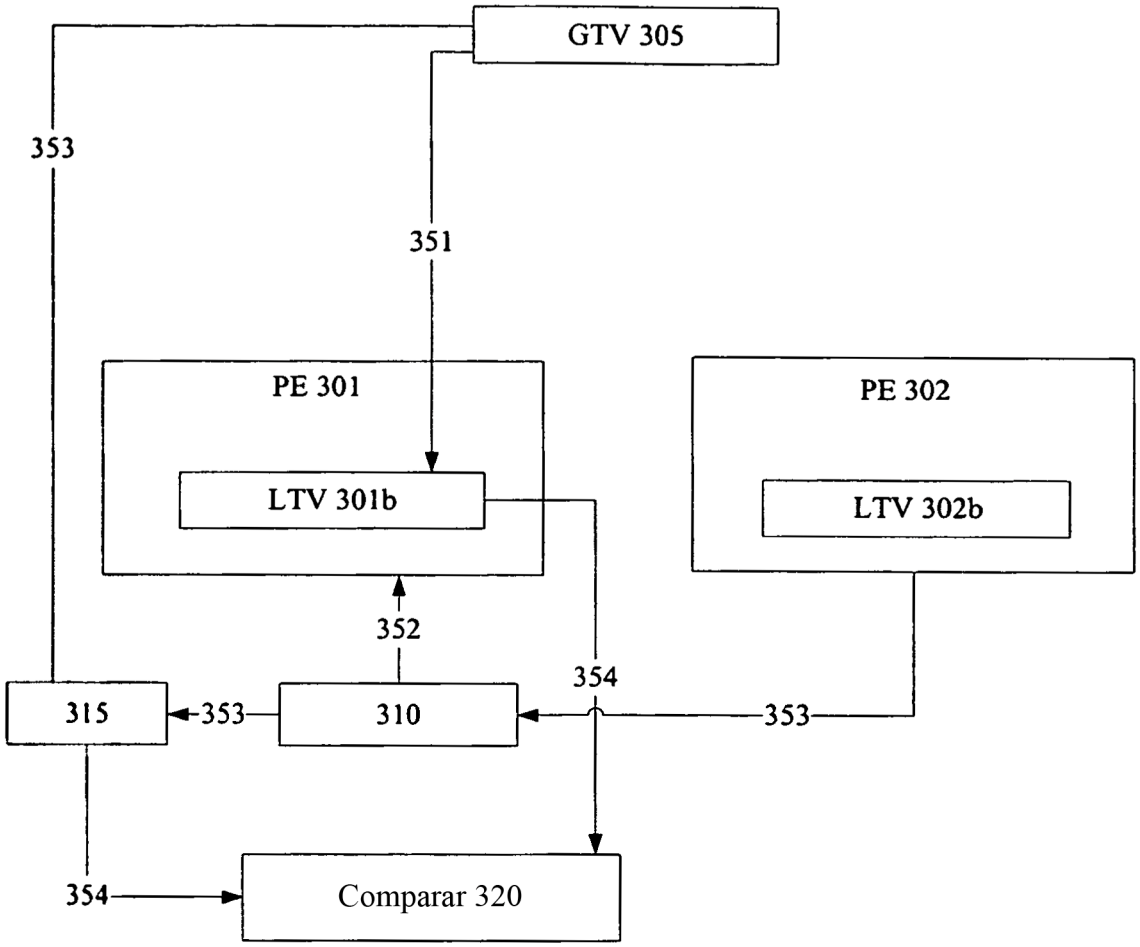
GTV 305
353
351
PE 301
LTV 301b
PE 302
LTV 302b
352
354
315
353
310
353
354
Comparar 320

FIG. 3a

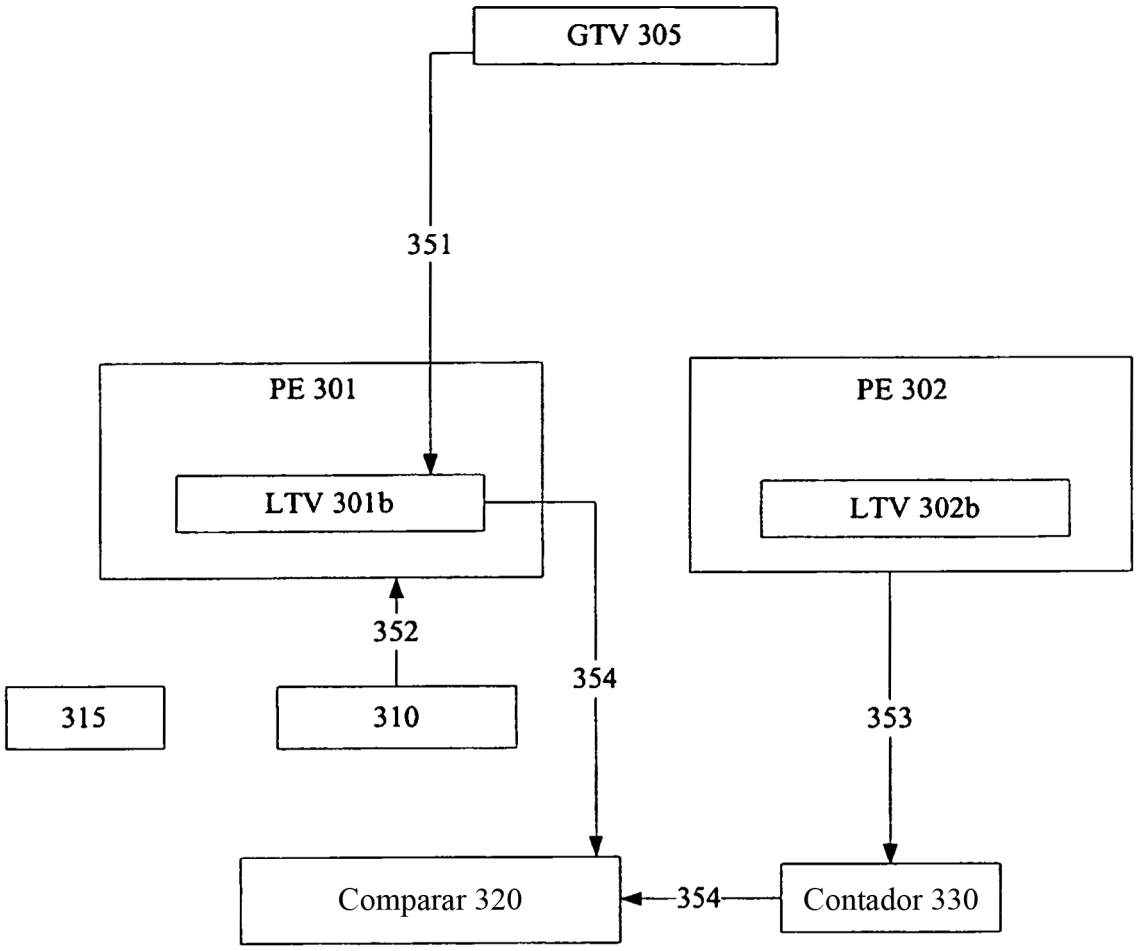
GTV 305
351
PE 301
LTV 301b
PE 302
LTV 302b
352
354
315
310
353
Comparar 320
354
Contador 330

FIG. 3b

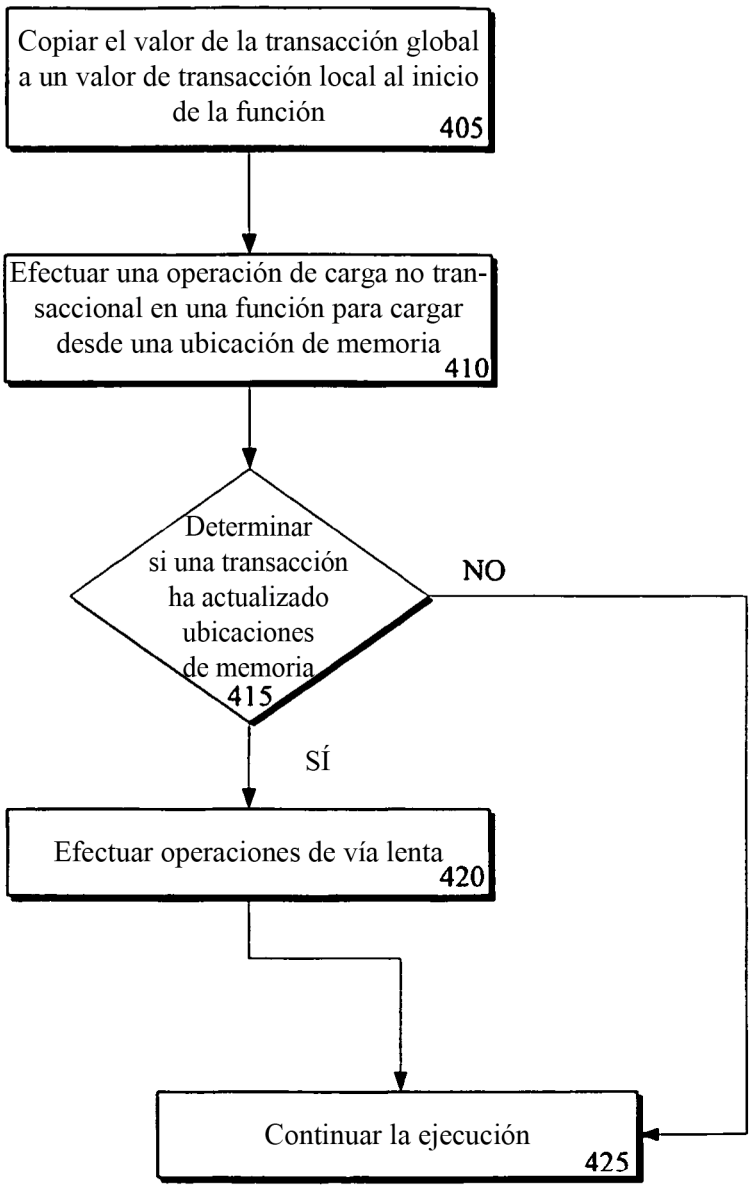
Copiar el valor de la transacción global a un valor de transacción local al inicio de la función
405
Efectuar una operación de carga no transaccional en una función para cargar desde una ubicación de memoria
410
Determinar si una transacción ha actualizado ubicaciones de memoria
415
NO
SÍ
Efectuar operaciones de vía lenta
420
Continuar la ejecución
425

FIG. 4a

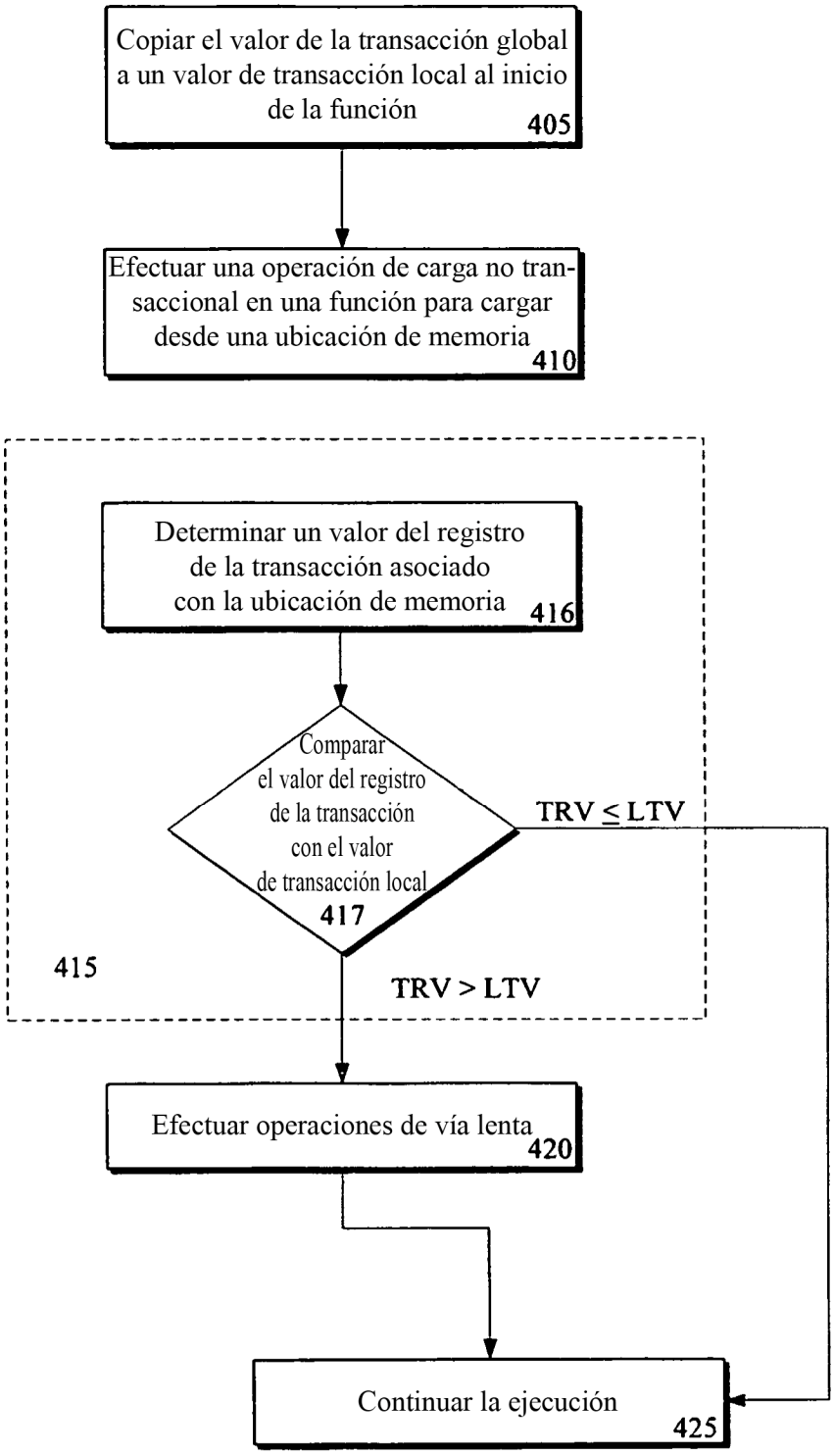
Copiar el valor de la transacción global a un valor de transacción local al inicio de la función
405
Efectuar una operación de carga no transaccional en una función para cargar desde una ubicación de memoria
410
Determinar un valor del registro de la transacción asociado con la ubicación de memoria
416
Comparar el valor del registro de la transacción con el valor de transacción local
417
TRV ≤ LTV
TRV > LTV
415
Efectuar operaciones de vía lenta
420
Continuar la ejecución
425

FIG. 4b

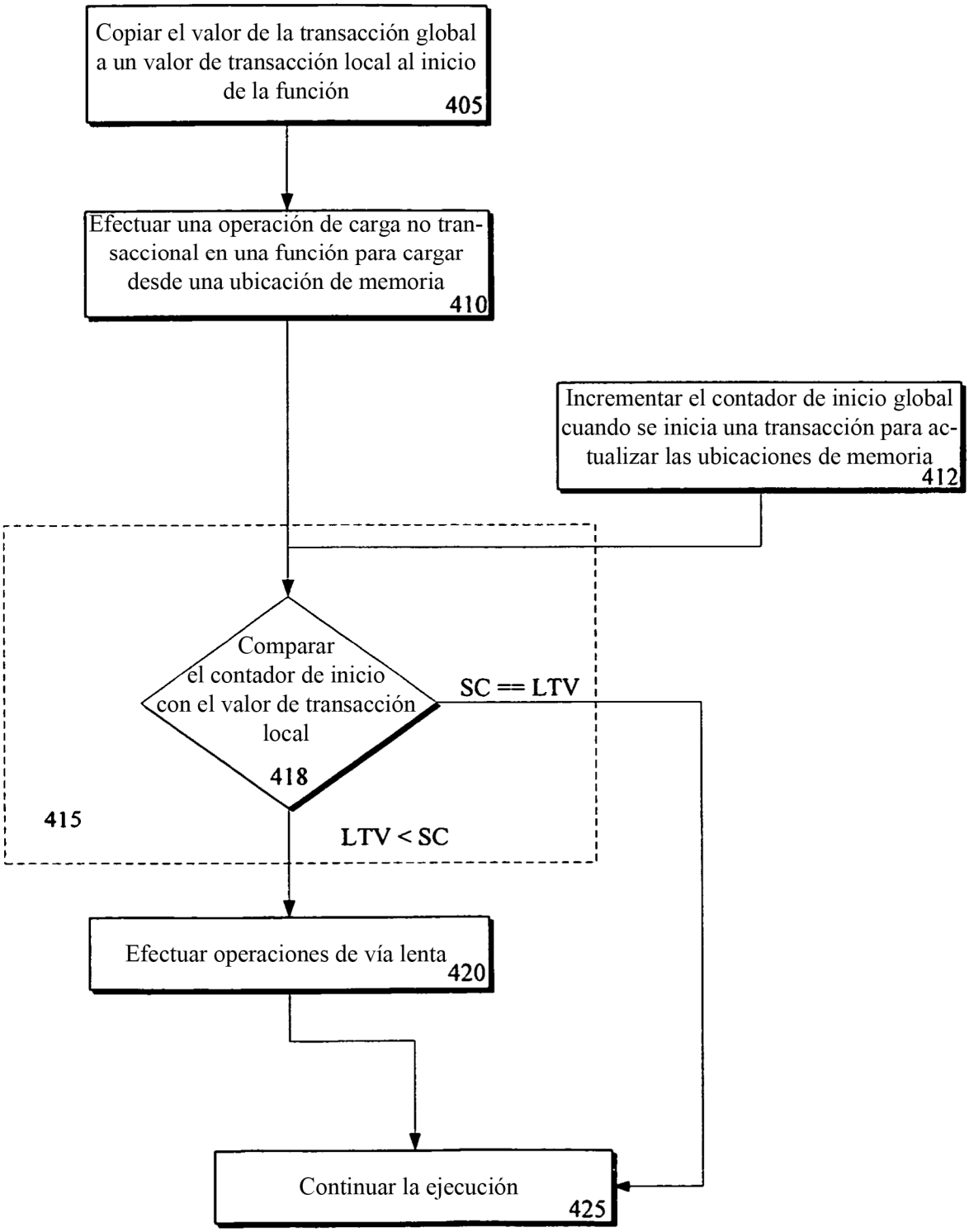
Copiar el valor de la transacción global a un valor de transacción local al inicio de la función
405
Efectuar una operación de carga no transaccional en una función para cargar desde una ubicación de memoria
410
Incrementar el contador de inicio global cuando se inicia una transacción para actualizar las ubicaciones de memoria
412
Comparar el contador de inicio con el valor de transacción local
418
SC == LTV
LTV < SC
415
Efectuar operaciones de vía lenta
420
Continuar la ejecución
425

FIG. 4c