(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 620 667**

(51) Int. CI.:

G06F 9/54 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **10.05.2012 PCT/EP2012/058687**

(87) Fecha y número de publicación internacional: **03.01.2013 WO2013000616**

(96) Fecha de presentación y número de la solicitud europea: **10.05.2012 E 12722111 (7)**

(97) Fecha y número de publicación de la concesión europea: **22.02.2017 EP 2591417**

(54) Título: **Facilitación de la comunicación entre espacios de memoria aislados de un entorno de comunicaciones**

(30) Prioridad:

30.06.2011 US 201113172978

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
29.06.2017

(73) Titular/es:

INTERNATIONAL BUSINESS MACHINES CORPORATION (100.0%)
New Orchard Road
Armonk, NY 10504, US

(72) Inventor/es:

MACCHIANO, ANGELO;
TARCZA, RICHARD;
WINTER, ALEXANDRA;
SITTMANN III, GUSTAV y
STEVENS, JERRY

(74) Agente/Representante:

DE ELZABURU MÁRQUEZ, Alberto

DESCRIPCIÓN

Facilitación de la comunicación entre espacios de memoria aislados de un entorno de comunicaciones

Antecedentes

Un aspecto de la presente invención se refiere, en general, a comunicar dentro de un entorno de comunicaciones, y en particular, a facilitar la transferencia de datos entre espacios de memoria aislados del entorno de comunicaciones.

Para transferir datos entre espacios de memoria aislados, habitualmente se utiliza tecnología y protocolos de trabajo en red. Por ejemplo, se pueden enviar datos desde un espacio de memoria aislado a otro espacio de memoria aislado utilizando un protocolo TCP/IP sobre una conexión Ethernet. La actual tecnología de trabajo en red permite enviar los datos de manera síncrona o bien asíncrona. Esto es una elección exclusiva de la parte del emisor.

Cuando los datos se envían de manera síncrona, el emisor es interrumpido hasta que se completa la transferencia de datos. Por otra parte, si los datos se envían de manera asíncrona, entonces el emisor puede continuar las operaciones.

La publicación de solicitud de patente de Estados Unidos número US 2009/0024714 A1 presenta un procedimiento para proporcionar acceso remoto directo a memoria (RDMA, remote direct memory access) entre dos ordenadores.

La publicación de solicitud de patente de Estados Unidos número US 2003/065709 A1 utiliza heurística para determinar dinámicamente si una solicitud determinada se debe conducir de manera síncrona o asíncrona.

Breve compendio

En la presente memoria también se describen y reivindican procedimientos y sistemas relacionados con uno o varios aspectos de la presente invención. Además, en la presente memoria también se describen y se pueden reivindicar servicios relacionados con uno o varios aspectos de la presente invención.

Se analizan características y ventajas adicionales por medio de las técnicas de uno o varios aspectos de la presente invención. En la presente memoria se describen en detalle otras realizaciones y aspectos de la invención, y se consideran parte de la invención reivindicada.

Considerada desde un primer aspecto, la presente invención da a conocer un procedimiento para facilitar las comunicaciones en un entorno de comunicaciones según la reivindicación 1.

Considerada desde otro aspecto, la presente invención da a conocer un producto de programa informático para facilitar las comunicaciones en un entorno de comunicaciones, comprendiendo el producto de programa informático un medio de almacenamiento legible por ordenador, que puede ser leído por un circuito de procesamiento y que almacena instrucciones para su ejecución por el circuito de procesamiento con el fin de llevar a cabo un procedimiento para implementar las etapas de la invención.

Considerada desde otro aspecto más, la presente invención da a conocer un sistema para facilitar las comunicaciones en un entorno de comunicaciones de acuerdo con la reivindicación 10.

Breve descripción de las diversas vistas de los dibujos

Uno o varios aspectos de la presente invención se señalan particularmente y se reivindican expresamente como ejemplos en las reivindicaciones a la finalización de la memoria descriptiva. Los anteriores y otros objetivos, características y ventajas de uno o varios aspectos de la invención son evidentes a partir de la siguiente descripción detallada, tomada junto con los dibujos adjuntos, en los cuales:

la figura 1 representa un ejemplo de un entorno de comunicaciones para incorporar y/o utilizar uno o varios aspectos de la presente invención;

la figura 2 representa ejemplos de espacios de memoria asociados con particiones lógicas de la figura 1, de acuerdo con un aspecto de la presente invención;

la figura 3 representa un ejemplo de transferencia de datos de salida síncrona utilizada de acuerdo con un aspecto de la presente invención.

la figura 4 representa ejemplos de estructuras de control utilizadas para transformar automáticamente una transferencia de datos síncrona en una transferencia de datos asíncrona, de acuerdo con un aspecto de la presente invención;

la figura 5A representa una realización de la lógica para transformar automáticamente una transferencia de datos síncrona en una transferencia de datos asíncrona, de acuerdo con un aspecto de la presente invención;

la figura 5B representa gráficamente un ejemplo de finalización satisfactoria de una transferencia de datos asíncrona, de acuerdo con un aspecto de la presente invención;

la figura 6 representa una realización de un producto de programa informático que incorpora uno o varios aspectos de la presente invención;

la figura 7 representa una realización de un sistema de ordenador principal para incorporar y utilizar uno o varios aspectos de la presente invención;

la figura 8 representa otro ejemplo de un sistema informático para incorporar y utilizar uno o varios aspectos de la presente invención.

la figura 9 representa otro ejemplo de un sistema informático que comprende una red informática para incorporar y utilizar uno o varios aspectos de la presente invención;

la figura 10 representa una realización de varios elementos de un sistema informático para incorporar y utilizar uno o varios aspectos de la presente invención;

la figura 11A representa una realización de la unidad de ejecución del sistema informático de la figura 10 para incorporar y utilizar uno o varios aspectos de la presente invención;

la figura 11B representa una realización de la unidad de derivación del sistema informático de la figura 10 para incorporar y utilizar uno o varios aspectos de la presente invención;

la figura 11C representa una realización de la unidad de carga/almacenamiento del sistema informático de la figura 10 para incorporar y utilizar uno o varios aspectos de la presente invención; y

la figura 12 representa una realización de un sistema de ordenador principal para incorporar y utilizar en uno o varios aspectos de la presente invención.

Descripción detallada

De acuerdo con un aspecto de la presente invención, se da a conocer la capacidad de transformar automáticamente una transferencia de datos síncrona en una transferencia de datos asíncrona. Por ejemplo, una transferencia de datos síncrona se transforma automáticamente en una transferencia de datos asíncrona, en respuesta a la determinación de que hay un retardo en la compleción de la transferencia de datos, tal como, por ejemplo, que el receptor de los datos no puede recibir los datos en el momento de la transferencia. La transformación de una transferencia de datos síncrona a una transferencia de datos asíncrona se lleva a cabo automáticamente, por cuanto que no es a petición del emisor (ni del receptor), y el emisor (o el receptor) no está en conocimiento de la transformación en el momento en el que se inicia la transformación. Además, en el momento de la transformación, el emisor (o el receptor) no tiene que adoptar ninguna acción o intervenir en la transformación.

Se describe una realización de un entorno de comunicaciones para incorporar y/o utilizar uno o varios aspectos de la presente invención haciendo referencia a la figura 1. En el ejemplo, un entorno de comunicaciones 100 incluye un complejo de ordenadores centrales (CPC, central processor complex) 102, que está basado en z/Architecture®, ofrecida por International Business Machines Corporation (IBM®). Se describen aspectos de z/Architecture® en la publicación de IBM® titulada "z/Architecture Principles of Operation," publicación IBM número SA22-7832-08, agosto de 2010.

Un sistema que puede incluir el complejo de procesadores centrales 102 es el sistema zEnterprise 196 (z196), ofrecido por Business Machines Corporation, Armonk, Nueva York. IBM® y z/Architecture® son marcas comerciales registradas, y zEnterprise 196 y z196 son marcas comerciales de International Business Machines Corporation, Armonk, Nueva York, USA. Otros nombres utilizados en la presente memoria pueden ser marcas comerciales registradas, marcas comerciales o nombres de producto de International Business Machines Corporation u otras compañías.

El complejo de procesadores centrales 102 incluye, por ejemplo, una o varias particiones 104, un monitor de máquina virtual 106, uno o varios procesadores centrales 108 y uno o varios componentes de un subsistema de entrada/salida 110. En este ejemplo, una o varias particiones 104 son particiones lógicas (alias, LPARs), que incluyen un conjunto de recursos de hardware del sistema virtualizados como un sistema independiente.

Cada partición lógica 104 puede funcionar como un sistema independiente. Es decir, cada partición lógica se puede reiniciar independientemente, cargar inicialmente con un sistema operativo 120, si se desea, y funcionar con diferentes programas. Un sistema operativo o un programa de aplicación que se ejecuta en una partición lógica parece tener acceso a un sistema entero y completo, pero en realidad está disponible solamente una parte del mismo. Una combinación de hardware y código interno con licencia (LIC, licensed internal code), denominada software inalterable, evita que un programa en una partición lógica interfiera con un programa en una partición lógica diferente. Esto permite que diversas particiones lógicas diferentes funcionen en uno solo, o en múltiples procesadores físicos en un modo de segmentación temporal. En este ejemplo, una serie de particiones lógicas

tienen un sistema operativo residente 120, que puede diferir para una o varias particiones lógicas. En una realización, el sistema operativo 120 es el sistema operativo z/OS®, ofrecido por International Business Machines Corporation, Armonk, Nueva York.

Tal como se utiliza en la presente memoria, software inalterable incluye, por ejemplo, microcódigo, milicódigo y/o macrocódigo del procesador. Esto incluye, por ejemplo, las instrucciones a nivel de hardware y/o las estructuras de datos utilizadas en la implementación del código máquina de nivel superior. En una realización, esto incluye, por ejemplo, código propietario que se suministra habitualmente como microcódigo que incluye soporte de confianza o microcódigo específico del hardware subyacente, y controla el acceso del sistema operativo al hardware del sistema.

Las particiones lógicas 104 son administradas por un monitor de máquina virtual 106, que está implementado mediante software inalterable que se ejecuta en procesadores centrales 108. Un ejemplo de monitor de máquina virtual 106 es el Processor Resource/Systems Manager (PR/SM™), ofrecido por International Business Machines Corporation, Nueva York.

Los procesadores centrales 108 son recursos de procesadores físicos que están asignados a las particiones lógicas. Por ejemplo, una partición lógica 104 incluye uno o varios procesadores lógicos, cada uno de los cuales representa la totalidad o una parte del recurso de procesadores físicos 108 asignado a la partición. Los procesadores lógicos de una partición particular 104 pueden estar dedicados a la partición, de tal modo que el recurso de procesador subyacente está reservado para dicha partición; o bien estar compartidos con otra partición, de tal modo que el recurso de procesador subyacente está disponible potencialmente para otra partición.

Las particiones lógicas 104 y los monitores de máquina virtual 106 comprenden, cada uno, uno o varios programas que residen en partes respectivas de memoria principal 150 asociadas con los procesadores centrales. En un ejemplo, cada partición lógica está asignada a una parte de memoria principal, denominada un espacio de memoria, tal como se describe en mayor detalle haciendo referencia a la figura 2.

Haciendo referencia a la figura 2, en una realización, la memoria principal 150 incluye una serie de espacios de memoria, que comprenden cada uno un intervalo de direcciones de memoria principal. Un espacio de memoria se puede asignar a una entidad, tal como una partición lógica u otra entidad. En el ejemplo mostrado en la figura 2, hay dos espacios de memoria asignados a dos particiones lógicas, respectivamente. Un espacio de memoria se denomina en el presente documento el espacio de memoria 202 del emisor, y el otro espacio de memoria se denomina el espacio de memoria 204 del receptor, dado que se describen a continuación en mayor detalle comunicaciones entre un emisor y un receptor. El espacio de memoria 202 del emisor incluye, por ejemplo, una o varias colas de entrada 210, una o varias colas de salida 212 y una o varias memorias tampón 214. De manera similar, el espacio de memoria 204 del receptor incluye una o varias colas de entrada 220, una o varias colas de salida 222 y una o varias memorias tampón 224. La utilización de las colas y las memorias tampón se describe en mayor detalle a continuación.

Los espacios de memoria individuales están aislados entre sí en el sentido de que los datos no se pueden escribir directamente desde un espacio de memoria al otro espacio de memoria sin un control por parte del software inalterable. En un ejemplo, se utilizan transmisiones de trabajo en red que utilizan, por ejemplo, TCP/IP sobre conexiones de internet, para transferir datos de un espacio de memoria a otro espacio de memoria. En un ejemplo particular, para llevar a cabo la transferencia se utiliza una tecnología ofrecida por Business Machines Corporation, denominada HiperSockets™.

HiperSockets™ proporciona conectividad TCP/IP de alta velocidad en el interior de un complejo de procesadores centrales. Esto elimina la necesidad de cualquier cableado físico o conexiones externas de trabajo en red entre servidores funcionando en particiones lógicas diferentes. Por el contrario, la comunicación es a través de la memoria de sistema del procesador. La implementación de HiperSockets™ está basada en el protocolo de E/S de cola directa (QDIO, Queued Direct I/O) OSA-Express. El software inalterable emula la capa de control de enlace de una interfaz QDIO OSA-Express.

La transferencia de datos de un espacio de memoria a otro espacio de memoria utilizando tecnología de trabajo en red es, por ejemplo síncrona, en la que después de la iniciación de la transferencia de datos, el emisor es interrumpido hasta que se completa la transferencia. Se describe un ejemplo de una transferencia de datos síncrona de un espacio de memoria a otro haciendo referencia a la figura 3.

Haciendo referencia a la figura 3, un emisor 300, tal como una pila TCP/IP o un programa en ejecución en el espacio de memoria del emisor, inicia una solicitud para enviar datos a un receptor 310, tal como otro programa u otra pila TCP/IP, como ejemplos, en el espacio de memoria del receptor. Dado que el espacio de memoria del receptor está aislado del espacio de memoria del emisor, en este ejemplo, se utiliza un protocolo de comunicaciones de red para llevar a cabo una transmisión de datos síncrona de los datos que se envían del emisor al receptor. Utilizar una transferencia de datos síncrona proporciona un camino de comunicaciones directo muy rápido, de baja latencia, entre el emisor y el receptor mediante la realización de una transferencia de memoria desde una ubicación a otra, bajo el control del software inalterable. En un ejemplo, la transferencia de datos de una memoria a otra se lleva a cabo mediante HiperSockets™. Este mecanismo de transferencia es muy eficiente siempre que el receptor pueda recibir los datos a la misma velocidad a la que el emisor está enviando los datos, o más rápidamente.

Para transferir los datos, el emisor toma los datos contenidos dentro de la memoria tampón de datos 320 seleccionada y los pone en una cola de salida 330 del emisor. Por ejemplo, se sitúa un puntero 332 de la memoria tampón de datos seleccionada, en la cola de salida. A continuación, el emisor 300 señaliza 340 el procesador para llevar a cabo la transferencia de datos al receptor 310. En un ejemplo, es software inalterable 350 del procesador lo que es señalizado y va a llevar a cabo la transferencia, si bien en otros ejemplos no es software inalterable, sin otro código y/o hardware del procesador.

En respuesta a la recepción de la señal que solicita una transferencia de datos del emisor al receptor, el software inalterable copia los datos de la cola de salida del emisor y los coloca en la cola de entrada 360 del receptor. Por ejemplo, los datos se copian a una memoria tampón vacía 370 y se coloca un puntero 372 a los datos en la cola de entrada 360. Después de la compleción de la transferencia de datos, el software inalterable proporciona una señal de vuelta al emisor indicando que la transferencia sea completado. Hasta que el emisor recibe esta señal de compleción, el emisor está interrumpido y no puede llevar a cabo ninguna otra operación.

Como un ejemplo particular, para señalizar el procesador, el emisor emite una instrucción de adaptador de señal (SIGA, Signal Adapter), que indica una función de escritura (SIGA-w) que señaliza al procesador que una o varias colas de entrada tienen datos para transmitir al receptor. La función de escritura se especifica como un código de función proporcionado en un primer registro general utilizado por la instrucción, y la dirección de la conexión de red (por ejemplo, una palabra de identificación de subsistema) de la función de escritura se indica en un segundo registro general utilizado por la instrucción. Además, las colas de salida se especifican en un tercer registro general utilizado por la instrucción.

En este ejemplo particular, las colas se implementan como colas de E/S de cola directa (QDIO), y cada cola tiene una serie de memorias tampón asociadas con la misma, así como diversa información de control. En una realización, una cola QDIO indica estructuras de datos que describen la cola, así como bloques de almacenamiento de memoria tampón que se utilizan para transferencia de datos. Como un ejemplo, las múltiples estructuras de datos de almacenamiento, denominadas componentes de cola, que describen colectivamente las características de la cola y proporcionan los controles para permitir el intercambio de datos, incluyen, por ejemplo:

Un bloque de información de cola (QIB, queue information block) que incluye información sobre la recopilación de colas de entrada y salida QDIO. El QIB incluye una dirección de bloque de información de lista de almacenamiento (SLIB, storage list information block) para colas de entrada y una dirección SLIB para colas de salida.

Hay un SLIB para cada cola, y cada SLIB proporciona información sobre la cola y sobre cada memoria tampón de cola de la cola. Cada SLIB tiene una cabecera y una o varias entradas, denominadas entradas de bloque de información de lista de almacenamiento (SLIBEs, storage list information block entries) que contienen información sobre cada una de las memorias tampón para cada cola. En un ejemplo, cada bloque de información de lista de almacenamiento incluye una dirección de un siguiente bloque de información de lista de almacenamiento, una dirección de una lista de almacenamiento (SL, storage list) y una dirección de un bloque de estado de lista de almacenamiento (SLSB, storage list state block).

Hay una lista de almacenamiento definida para cada cola, y una SL incluye, por ejemplo, 128 entradas, una entrada para cada una de las memorias tampón de la cola. La lista de almacenamiento proporciona información acerca de las ubicaciones de memoria tampón de E/S en memoria principal. Cada entrada incluye la dirección absoluta de una lista de direcciones de bloque de almacenamiento (SBAL, storage block address list). Cada lista de dirección de bloques de almacenamiento incluye una lista de direcciones absolutas de los bloques de almacenamiento que constituyen colectivamente una de las memorias tampón de datos asociadas con cada cola.

Se proporciona una entrada de lista de bloque de almacenamiento (SBALE, storage block list entry) como parte de cada SBAL. Cada SBALE incluye la dirección de almacenamiento absoluta de un bloque de almacenamiento. Colectivamente, los bloques de almacenamiento a los que se dirigen todas las entradas de una SBAL constituyen una de las muchas posibles memorias tampón QDIO de una cola QDIO. En un ejemplo, una cola QDIO puede tener 128 memorias tampón QDIO asociadas con la misma.

El SLSB incluye indicadores de estado que proporcionan información de estado acerca de las memorias tampón que componen la cola.

Se describen detalles adicionales en relación con SIGA, colas QDIO y estructuras de control asociadas en la patente U.S.A. número de serie 6.332.171 B1, titulada "Self-Contained Queues With Associated Control Information For Receipt And Transfer Of Incoming And Outgoing Data Using A Queued Direct Input-Output Device", expedida el 18 de diciembre de 2001; la patente U.S.A. número de serie 6.345.241 B1, titulada "Method And Apparatus For Simulation Of Data In a Virtual Environment Using A Queued Direct Input-Output Device", expedida el 5 de febrero de 2002; la patente U.S.A. número de serie 6.519.645 B2, titulada "Method And Apparatus For Providing Configuration Information Using A Queued Direct Input-Output Device," expedida el 11 de febrero de 2003; y la patente U.S.A. de número de serie 7.941.799 B2, titulada "Interpreting I/O Operation Requests From Pageable Guests Without Host Intervention", expedida el 10 de mayo de 2011.

En el proceso anterior, cuando un receptor no puede proporcionar memorias tampón vacías a la misma velocidad a la que el emisor está enviando datos, se introduce una mayor latencia y una sobrecarga de la CPU en el lado del emisor, debido a la naturaleza síncrona del protocolo. Cuando no está disponible una memoria tampón vacía en el lado del receptor, el emisor tiene dos opciones. Puede someterse a la sobrecarga de poner en cola la operación fallida y la subsiguiente retransmisión de datos al mismo receptor, o puede desechar los datos permitiendo que un protocolo de comunicaciones de nivel superior, tal como TCP/IP, reconduzca la operación. Los problemas de poner en cola y retransmitir los datos son que ello no sólo requiere recuperar ciclos de CPU adicionales, sino que potencialmente puede bloquear o retardar mar las transmisiones del emisor a otros destinos, que pueden estar en disposición de aceptar datos entrantes.

Los entornos virtualizados que permiten que múltiples servidores (por ejemplo, emisores, receptores) compartan recursos de CPU, es más probable que creen situaciones en las que un receptor puede no conseguir mantener el ritmo de varios emisores enviándole datos. Éste es habitualmente el caso cuando un monitor de máquina virtual, tal como PR/SM, controla la expedición de servidores en varios procesadores compartidos disponibles. El problema se complica cuando hay múltiples niveles de monitores de máquina virtual entre el emisor y el recurso CPU compartido. Éste es el caso, por ejemplo, cuando el emisor se está ejecutando en una máquina virtual bajo z/VM®, que está ejecutando asimismo una partición lógica. En este caso, es necesario que dos monitores de máquina virtual expidan al servidor para permitir que recargue las memorias tampón a tiempo.

Expedir al receptor a tiempo no es un problema cuando hay suficientes recursos disponibles de CPU. La transferencia de datos síncrona se interrumpe solamente cuando los recursos de CPU se limitan a corto o bien a largo plazo. Por lo tanto, de acuerdo con un aspecto de la presente invención, el emisor puede explotar una transmisión de datos síncrona de baja latencia en un entorno de CPU sin restricciones, eliminando al mismo tiempo los problemas y sobrecarga asociados cuando los recursos de CPU pasan a ser limitados. Esto se consigue transformando el protocolo de comunicaciones de síncrono a asíncrono cuando el receptor no consigue mantener el ritmo del emisor. En una realización, la transformación de protocolo síncrono a asíncrono se lleva a cabo automáticamente sin ninguna remisión al emisor (es decir, el emisor no tiene que llevar a cabo ninguna acción en el momento de la transformación). Esto elimina que el emisor tenga que realizar ningún tipo de proceso de recuperación, o bloquear o ralentizar transmisiones de datos a otros receptores cuando los recursos de CPU pasan a estar limitados para un receptor particular. Además, permite que las comunicaciones síncronas se reanuden automáticamente a un receptor que estaba limitado anteriormente.

De acuerdo con un aspecto de la presente invención, se proporciona la capacidad de que el emisor ponga en cola datos dentro de su memoria hasta que el receptor proporcione memorias tampón vacías que puedan recibir los datos. Para facilitar este proceso, se utilizan una o varias estructuras de control, tal como se describe haciendo referencia a la figura 4. Por ejemplo, el emisor asigna un bloque de memoria vacío 400 en la memoria del emisor para cada transferencia asíncrona pendiente. En un ejemplo, puede haber un número X de estos bloques cada vez, donde X depende del modelo y es configurable. X representa el número de solicitudes asíncronas simultáneas permitidas por el emisor. Este bloque de memoria, denominado bloque de funcionamiento asíncrono QDIO (entrada/salida de cola directa) (QAOB, QDIO asynchronous operation block), se utiliza para mantener un seguimiento de la transferencia de datos asíncrona hasta que el software inalterable completa la operación. El QAOB contiene solamente información de control y no los propios datos, en este ejemplo. El emisor proporciona este bloque cuando inicia una transferencia de datos para solicitudes que permitirá opcionalmente ejecutar de manera asíncrona; de lo contrario, no tiene que proporcionar el bloque de control. Esto proporciona al emisor la capacidad de controlar el máximo número de solicitudes asíncronas pendientes. El propio QAOB es solamente utilizado e inicializado mediante software inalterable, cuando éste determina que la transferencia de datos se tiene que llevar a cabo de manera asíncrona. El emisor no tiene que hacer nada para la transferencia de datos salvo proporcionar memoria para el QAOB, en caso de que la transferencia de datos se lleve a cabo de manera asíncrona. En un ejemplo particular, el QAOB está incluido en la instrucción de SIGA que proporciona la solicitud. El emisor emite una instrucción de escritura SIGA con QAOB (SIGA-wq) que especifica un código de función seleccionado en el primer registro general que indica la función de escritura con QAOB. La dirección QAOB se especifica en un cuarto registro general utilizado por la instrucción. Este registro general tiene un 0 cuando no se especifica ningún QAOB o una dirección absoluta de un QAOB (por ejemplo, un QAOB de 256 bytes (octetos)). El software inalterable, en respuesta al envío del código de función de escritura con QAOB, determina si en el cuarto registro general se especifica un QAOB que puede ser utilizado en una transferencia de datos asíncrona.

Cuando el software inalterable cambia la transferencia de datos a un protocolo asíncrono, utiliza el QAOB para mantener un seguimiento de los datos residentes en la memoria del emisor asociada con la transferencia de datos de salida. En el ejemplo en el que se utiliza HiperSockets™ para la transmisión, el QAOB mantiene un seguimiento de las direcciones y controles especificados por el emisor en una lista de direcciones de bloque de almacenamiento (SBAL) asociada con la transferencia de datos. Ejemplos de campos extraídos de la SBAL y situados en el QAOB incluyen, por ejemplo:

- Todas las SBALE con significado procedentes de la SBAL (por ejemplo, la primera SBALE hasta la SBALE con el último conjunto de bits de entrada). Esto incluye la dirección de memoria tampón absoluta de los datos y un cómputo de bytes;

- El número de cola de salida para la SBAL;
- El número de memoria tampón (por ejemplo, 1 a 27) de la SBAL que inicia la solicitud;
- El número de entradas de SBALE con significado; y
- La clave de almacenamiento utilizada para acceder a los bloques de almacenamiento indicados mediante dicha SBALE con significado.

Además del QAOB, se utiliza otra estructura de control denominada cola de compleción (CQ, completion queue) 410. Es decir, en un ejemplo, además de proporcionar un QAOB, el emisor asigna asimismo un nuevo tipo de cola, una cola de compleción, en su memoria cuando establece colas de comunicación. En el caso de HiperSockets™, éste es un nuevo tipo de cola de entrada QDIO con SBAL, pero sin memorias tampón asociadas con las SBALE. Esta nueva cola de entrada no se utiliza para transferir datos; en lugar de esto, es utilizada por el software inalterable para publicar eventos de compleción para señalizar a un emisor que una transferencia de datos asíncrona se ha completado. Cuando una entrada de cola pasa a 'preparada para entrada', la información sobre el evento de compleción se sitúa en los propios SBALE, que están incluidos en la cola de compleción. El software inalterable publica la dirección del QAOB asociado con la transferencia de datos asíncrona completada en la CQ y genera una interrupción, si es necesario, para señalizar al emisor que la transferencia de datos se ha completado. Momento en que el emisor puede reutilizar la memoria asociada con la operación completada, para otros propósitos. (En una realización, se utiliza una SBALE para publicar un solo evento de compleción. Dado que una SBAL incluye, por ejemplo, 16 SBALE, el software inalterable puede publicar hasta 16 eventos de compleción (QAOB) dentro de una sola SBAL)

Además, en una realización, el software inalterable tiene otra cola, TPQ 420 en cada destinatario previsto, utilizada para recordar que tiene una solicitud de transferencia de datos pendiente.

Se describen otros detalles en relación con la transformación de una transferencia de datos síncrona a una transferencia de datos asíncrona, haciendo referencia a las figuras 5A-5B. La figura 5A representa una realización de la lógica utilizada por el software inalterable para llevar a cabo la transformación, y la figura 5B representa gráficamente un ejemplo de la transformación. Se hace referencia a ambas figuras en la siguiente discusión.

Haciendo referencia a las figuras 5A a 5B, inicialmente, el software inalterable 350 recibe una indicación del QAOB (por ejemplo, una dirección de un bloque de almacenamiento que puede ser utilizado para una transferencia de datos asíncrona), etapa 500. Por lo tanto, éste sabe que, si lo requiere (o lo desea), puede llevar a cabo de manera asíncrona una transferencia de datos solicitada. En una realización, el software inalterable puede recibir una serie de QAOB que indican que éste puede llevar a cabo hasta dicho número de transferencias de datos de manera asíncrona. La provisión de un QAOB por el emisor es una autorización previa al software inalterable, de que el software inalterable puede llevar a cabo la transferencia de datos de manera asíncrona, si así lo elige.

En un ejemplo particular, el QAOB está incluido como parte de una solicitud de transferencia de datos recibida por el software inalterable desde el emisor 300, etapa 502. En respuesta a la recepción de solicitud de transferencia de datos, el software inalterable intenta enviar los datos al receptor, etapa 504. Si el receptor puede recibir los datos (por ejemplo, hay una memoria tampón vacía en el receptor), pregunta 506, entonces los datos son transferidos de manera síncrona, etapa 508, y la transferencia de datos se completa. A continuación, el emisor puede llevar a cabo otra transferencia de datos, y si el proceso asíncrono es permitido por el emisor, el emisor puede incluir el QAOB en la nueva solicitud.

Sin embargo, si el receptor no puede recibir actualmente los datos (por ejemplo, no hay una memoria tampón vacía en el receptor, según se ha determinado mediante el estado de la memoria tampón, y por lo tanto, el receptor se retarda en la recepción de los datos), entonces la transferencia de datos se transforma automáticamente de una solicitud síncrona a una solicitud asíncrona mediante el software inalterable, suponiendo que se ha proporcionado un QAOB en la solicitud. La solicitud se guarda en el QAOB, etapa 510, y el QAOB se pone en cola en la TPQ 420 en su destino previsto, por ejemplo, colocando un puntero al QAOB en la TPQ, etapa 512. El QAOB incluye ahora los contenidos de la SBAL, y por lo tanto, la SBAL puede ser utilizada para otros procesos.

En una realización, si el QAOB no se especifica, entonces la solicitud falla o queda en espera hasta que pueda ser enviada de manera síncrona.

En un ejemplo, en respuesta a la puesta en cola del QAOB, se proporciona el control a servidor de envío con una indicación de que el QAOB especificado está siendo utilizado para llevar a cabo esta transferencia de datos de manera asíncrona. El servidor de envío puede a continuación configurarse inmediatamente para su próxima transferencia de datos, y opcionalmente, asignar otro QAOB en caso de que sea necesario para la subsiguiente transferencia de datos.

A continuación, se realiza una determinación sobre si el receptor puede recibir los datos, pregunta 514. Como ejemplos, se realiza una determinación sobre si el software inalterable ha recibido del receptor una señal de que está disponible a continuación para aceptar datos (por ejemplo, actualmente hay disponibles memorias tampón

vacías) o sobre si el software inalterable ha determinado que hay disponible una memoria tampón mediante comprobar el estado de las memorias tampón. En un ejemplo particular, el receptor utiliza la instrucción de SIGA para señalizar al software inalterable que ha situado memorias tampón vacías en sus colas de entrada. Se especifica un código de función de lectura SIGA (SIGA-r) en el primer registro general. La función de lectura SIGA hace que el software inalterable transfiera cualesquiera paquetes pendientes desde la TPQ del receptor hasta las memorias tampón de entrada del objetivo.

Si el receptor no puede aceptar los datos, entonces el software inalterable espera. De lo contrario, si el receptor puede recibir los datos (por ejemplo, hay por lo menos una memoria tampón vacía), el software inalterable determina si tiene un puntero a un QAOB en la TPQ del receptor, pregunta 518. Si no, el proceso finaliza. De lo contrario, el software inalterable utiliza el QAOB para llevar a cabo la transferencia de datos. En particular, transmite los datos señalados mediante el QAOB al receptor, colocando los datos en la memoria tampón vacía y situando un puntero a la memoria tampón, ahora llena, en la cola de entrada del receptor.

A continuación, el software inalterable indica al emisor la compleción de la transferencia de datos, etapa 520. En un ejemplo, esta indicación incluye publicar la dirección del QAOB asociado con la transferencia de datos asíncrona completada, en la cola de compleción 410 (figura 5B) del emisor, y generar una interrupción 550, si es necesario, para señalizar al emisor la transferencia de datos completada. Momento en el que el emisor puede reutilizar la memoria asociada con la operación completada, para otros propósitos. El QAOB publicado en la CQ incluye información relativa al proceso asíncrono incluyendo, por ejemplo, información de estado, códigos de compleción, códigos de error, etc.

En un ejemplo particular, cuando la dirección del QAOB se publica en una entrada de la cola de compleción (es decir, la dirección del QAOB está incluida en la SBALE situada en una entrada de la cola de compleción), el software inalterable devuelve la siguiente información al programa en el QAOB, como ejemplo:

- Un código de causas que refleja los resultados de la operación E/S asíncrona, y

- El estado de la memoria tampón: "estado de cola - memoria tampón N (SQBN, State of Queue - Buffer N)" para la transferencia de datos asíncrona. Éste es el mismo valor que se habría situado en un SLSB para una transferencia de datos síncrona. Un SQBN contiene un valor que indica el estado actual del QAOB. El valor de estado incluye, por ejemplo, dos partes: una primera parte que indica si la memoria tampón pertenece al software inalterable del programa; y una segunda parte que indica el estado de proceso actual del QAOB.

Aunque la solicitud particular se lleva a cabo de manera asíncrona, se pueden llevar a cabo de manera síncrona otras solicitudes del emisor a otros receptores, salvo que un receptor particular esté, por ejemplo, retardado en la recepción de datos. Además, otras solicitudes al receptor desde el emisor vuelven automáticamente a ser síncronas salvo que se determine de nuevo que el receptor está retardado en la recepción de datos. Por ejemplo, el software inalterable intenta enviar los datos al receptor, tal como se ha descrito anteriormente, y si el receptor puede recibir los datos, estos se envían de manera síncrona. En una realización, aunque las transmisiones se transformen entre síncronas y asíncronas, los datos permanecen en el mismo orden FIFO en los que fueron transmitidos. El software inalterable transferirá todos los QAOB en cola en la TPQ 420 de los receptores antes de que se reciba cualquier solicitud síncrona futura desde éste o cualquier otro emisor. Si se produce otra transferencia de datos síncrona a un receptor que ya tiene QAOBs en su TPQ 420, será automáticamente transformada por el software inalterable en una solicitud asíncrona (si lo autoriza el emisor) o fallará con una respuesta de memorias tampón no disponibles. Preservar el orden evita un proceso costoso de reordenación, mejorando el rendimiento de la pila TCP/IP del receptor y la utilización global de CPU.

Anteriormente se ha descrito en detalle la capacidad de transformar automáticamente una transferencia de datos síncrona en una transferencia de datos asíncrona, en respuesta a la determinación por parte de procesador (por ejemplo, software inalterable) de que dicha transformación tiene que tener lugar. Como ejemplo, la transformación se lleva a cabo en respuesta a un retardo en la capacidad del receptor para recibir los datos (por ejemplo, no hay memoria tampón disponible, lentitud en la respuestas, etc.). Además, dicha capacidad permite que se lleven a cabo de manera síncrona otras transferencias al mismo receptor.

Uno o varios aspectos de este soporte proporcionan la capacidad de ajustar, de manera transitoria o en un plazo mayor, la transmisión de datos sin retransmisiones costosas con el destino objetivo acoplado con un efecto mínimo, o sin efecto, sobre otros destinos en funcionamiento en la red. En un aspecto, se proporciona la capacidad de conmutar en un sentido y otro entre transferencia de datos síncrona y asíncrona, sin recurrir al programa del emisor. En un ejemplo particular que utiliza cola de compleción (CQ) de HiperSockets, el controlador del dispositivo no interrumpe el funcionamiento E/S (SIGA no bloquea el proceso). Con CQ, el proceso de escritura se reserva y permite al emisor seguir emitiendo escrituras adicionales (SIGA(s)) al mismo objetivo o a otros objetivos, donde algunos se pueden completar de manera síncrona y otros se completan de manera asíncrona. Este aspecto de no bloqueo hace la CQ asíncrona.

De acuerdo con un aspecto de la presente invención, los datos se copian solamente una vez del emisor al receptor (no hay memoria tampón interna), independientemente de si la transferencia es síncrona o asíncrona.

Tal como apreciará un experto en la materia, los aspectos de la presente invención se pueden realizar como un sistema, un procedimiento o un producto de programa informático. Por consiguiente, los aspectos de la presente invención pueden adoptar la forma de una realización íntegramente de hardware, una realización íntegramente de software (incluyendo software inalterable, software residente, microcódigo, etc.) o una realización que combine aspectos de software y hardware que, en general, se puede denominar en conjunto en la presente memoria como un "circuito", "módulo" o "sistema". Además, los aspectos de la presente invención pueden adoptar la forma de un producto de programa informático incorporado en uno o varios medios legibles por ordenador que tienen incorporado en el mismo el código de programa legible por ordenador.

Se puede utilizar cualquier combinación de uno o varios medios no transitorios legibles por ordenador. El medio legible por ordenador puede ser un medio de almacenamiento legible por ordenador. Un medio de almacenamiento legible por ordenador puede ser, por ejemplo, de forma no limitativa, un sistema, aparato o dispositivo electrónico, magnético, óptico, electromagnético, por infrarrojos o semiconductor, o cualquier combinación adecuada de los anteriores. Más ejemplos específicos (en una lista no exhaustiva) del medio de almacenamiento legible por ordenador incluyen los siguientes: una conexión eléctrica que tiene uno o varios cables, un disquete informático portátil, un disco duro, una memoria de acceso aleatorio (RAM, random access memory), una memoria de sólo lectura (ROM, read-only memory), una memoria de sólo lectura programable borrable (EPROM o memoria flash), una fibra óptica, una memoria de sólo lectura de disco compacto (CD-ROM, compact disc read-only memory) portátil, un dispositivo de almacenamiento óptico, un dispositivo de almacenamiento magnético o cualquier combinación adecuada de los anteriores. En el contexto de este documento, un medio de almacenamiento legible por ordenador puede ser cualquier medio tangible que pueda contener o almacenar un programa para ser utilizado por, o en relación con un sistema, aparato o dispositivo de ejecución de instrucciones.

Haciendo referencia a continuación a la figura 6, en un ejemplo, un producto de programa informático 600 incluye, por ejemplo, uno o varios medios de almacenamiento 602 no transitorios legibles por ordenador para almacenar lógica 604 o medios de código de programa legible por ordenador con el fin de proporcionar y facilitar uno o varios aspectos de la presente invención.

El código de programa incorporado en un medio legible por ordenador puede ser transmitido utilizando un medio apropiado, que incluye de forma no limitativa, inalámbrico, cableado, cable de fibra óptica, RF, etc., o cualquier combinación adecuada de los anteriores.

El código de programa informático para llevar a cabo operaciones para aspectos de la presente invención se puede escribir en cualquier combinación de uno o varios lenguajes de programación, incluyendo un lenguaje de programación orientado a objetos, tal como Java, Smalltalk, C++ o similares, y lenguajes convencionales de programación por procedimientos, tales como el lenguaje de programación "C", ensamblador o lenguajes de programación similares. El código de programa se puede ejecutar íntegramente en el ordenador del usuario, parcialmente en el ordenador del usuario, como un paquete de software independiente, parcialmente en el ordenador del usuario y parcialmente en un ordenador remoto, o íntegramente en el ordenador remoto o servidor. En el último escenario, el ordenador remoto puede estar conectado al ordenador del usuario por medio de cualquier tipo de red, incluyendo una red de área local (LAN, local area network) o una red de área extensa (WAN, wide area network), o la conexión se puede realizar a un ordenador externo (por ejemplo, a través de internet utilizando un proveedor de servicios de internet).

En la presente memoria se describen aspectos de la presente invención haciendo referencia a ilustraciones de diagramas de flujo y/o a diagramas de bloques de procedimientos, aparatos (sistemas) y productos de programa informático según las realizaciones de la invención. Se comprenderá que cada bloque de ilustraciones de diagrama de flujo y/o diagrama de bloque, y combinaciones de bloques en las ilustraciones de diagrama de flujo y/o de diagramas de bloque, se pueden implementar mediante instrucciones de programa informático. Estas instrucciones de programa informático se pueden proporcionar mediante un procesador de un ordenador de propósito general, un ordenador de propósito especial u otro aparato programable de procesamiento de datos para producir una máquina, de tal modo que las instrucciones, que se ejecutan por medio del procesador del ordenador u otro aparato programable de procesamiento de datos, crean medios para implementar las funciones/acciones especificadas en el diagrama de flujo y/o en el bloque o bloques de los diagramas de bloques.

Estas instrucciones de programa informático se pueden almacenar asimismo en un medio legible por ordenador que pueda dar instrucciones a un ordenador, otro aparato programable de procesamiento de datos u otros dispositivos que funcionen de manera particular, de tal modo que las instrucciones almacenadas en el medio legible por ordenador produzcan un artículo de fabricación que incluye instrucciones que implementan la función/acción especificada en el diagrama de flujo y/o en el bloque o bloques de los diagramas de bloques.

Las instrucciones de programa informático se pueden cargar asimismo en un ordenador, otro aparato programable de procesamiento de datos u otros dispositivos, para hacer que se lleven a cabo una serie de etapas operativas en el ordenador, otros aparatos programables u otros dispositivos, con el fin de producir un proceso implementado por ordenador, de tal modo que las instrucciones que se ejecuten en el ordenador u otro aparato programable proporcionen procesos para implementar las funciones/acciones especificadas en el diagrama de flujo y/o en el bloque o bloques de los diagramas de bloques.

El diagrama de flujo y los diagramas de bloques en las figuras muestran la arquitectura, funcionalidad y funcionamiento de posibles implementaciones de sistemas, procedimientos y productos de programa informático, según diversas realizaciones de la presente invención. A este respecto, cada bloque en el diagrama de flujo o en los diagramas de bloques puede representar un módulo, segmento o parte de código que comprende una o varias instrucciones ejecutables para implementar la función o funciones lógicas especificadas. Se debe observar asimismo que, en algunas implementaciones alternativas, las funciones indicadas en el bloque pueden no producirse en el orden indicado en las figuras. Por ejemplo, dos bloques mostrados en sucesión pueden, de hecho, ejecutarse de manera sustancialmente simultánea, o en ocasiones los bloques se pueden ejecutar en orden inverso, dependiendo de la funcionalidad involucrada. Cabe señalar asimismo que cada bloque de los diagramas de bloques y/o de la ilustración de diagrama de flujo, y combinaciones de bloques en los diagramas de bloque y/o en la ilustración del diagrama de flujo se pueden implementar mediante sistemas basados en hardware de propósito especial que llevan a cabo funciones o acciones especificadas, o combinaciones de hardware de propósito especial e instrucciones de ordenador.

Además de lo anterior, uno o varios aspectos de la presente invención se pueden proporcionar, ofrecer, desplegar, gestionar, mantener, etc., mediante un proveedor de servicio que ofrezca gestión de entornos del cliente. Por ejemplo, el proveedor de servicio puede crear, mantener, soportar, etc. código informático y/o infraestructura informática que lleve a cabo uno o varios de los aspectos de la presente invención para uno o varios clientes. A cambio, el proveedor de servicio puede recibir un pago del cliente bajo un abono y/o un acuerdo de tarifas, como ejemplos. Adicional o alternativamente, el proveedor de servicio puede recibir el pago de la venta de contenido publicitario a una o varias terceras partes.

En un aspecto de la presente invención, se puede desplegar una aplicación para llevar a cabo uno o varios aspectos de la presente invención. Como ejemplo, el despliegue de una aplicación comprende proporcionar infraestructura informática que puede funcionar para llevar a cabo uno o varios aspectos de la presente invención.

Como un aspecto adicional de la presente invención, se puede desplegar una infraestructura informática que comprende integrar en un sistema informático código legible por ordenador, en el que el código en combinación con el sistema informático puede llevar a cabo uno o varios aspectos de la presente invención.

Como otro aspecto más de la presente invención, se puede proporcionar un proceso para integrar infraestructura informática que comprende integrar en un sistema informático código legible por ordenador. El sistema informático comprende un medio legible por ordenador, donde el medio legible por ordenador comprende uno o varios aspectos de la presente invención. El código en combinación con el sistema informático puede llevar a cabo uno o varios aspectos de la presente invención.

Aunque se han descrito anteriormente varias realizaciones, estas son solamente ejemplos. Por ejemplo, entornos informáticos de otras arquitecturas pueden incorporar y utilizar uno o varios aspectos de la presente invención. Como ejemplos, servidores diferentes a zEnterprise pueden incluir, utilizar y/o beneficiarse de uno o varios aspectos de la presente invención. Además, la transformación de síncrono a asíncrono puede ser en respuesta a consideraciones diferentes de si hay memoria tampón disponible. Asimismo, uno o varios aspectos de la presente invención pueden ser utilizados para cualesquiera transferencias de memoria a memoria entre espacios de memoria aislados. Son posibles asimismo muchas otras variaciones.

Además, otros tipos de entornos informáticos se pueden beneficiar de uno o varios aspectos de la presente invención. Como ejemplo, se puede utilizar un sistema de procesamiento de datos adecuado para almacenar y/o ejecutar código de programa, que incluye por lo menos dos procesadores acoplados directa o indirectamente a elementos de memoria por medio de un bus del sistema. Los elementos de memoria incluyen, por ejemplo, memoria local utilizada durante la ejecución real del código de programa, almacenamiento de gran capacidad y almacenamiento de memoria caché que puede el proporcionar almacenamiento temporal, por lo menos, de algún código de programa con el fin de reducir el número de veces que es necesario recuperar código desde el almacenamiento de gran capacidad durante la ejecución.

Pueden estar acoplados al sistema dispositivos de entrada/salida o E/S (incluyendo, de forma no limitativa, teclados, pantallas, dispositivos de puntero, DASD, cinta, CDs, DVDs, dispositivos USB y otros medios de memoria, etc.), ya sea directamente o a través de controladores de E/S intermedios. Se pueden asimismo acoplar al sistema adaptadores de red para permitir que el sistema de procesamiento de datos se acople a otros sistemas de procesamiento de datos o impresoras remotas o dispositivos de almacenamiento, por medio de redes intermedias privadas o públicas. Los módems, módems de cable y tarjetas Ethernet son tan sólo unos pocos de los tipos de adaptadores de red disponibles.

Haciendo referencia a la figura 7, se muestran componentes representativos de un sistema de ordenador principal 5000 para implementar uno o varios aspectos de la presente invención. El ordenador principal representativo 5000 comprende una o varias CPU 5001 en comunicación con memoria informática (es decir, almacenamiento central) 5002, así como interfaces de E/S a dispositivos de medios de almacenamiento 5011 y redes 5010 para comunicar con otros ordenadores o SANs y similares. La CPU 5001 es compatible con arquitectura que tiene un conjunto de instrucciones diseñado y funcionalidad diseñada. La CPU 5001 puede tener traducción dinámica de direcciones (DAT, dynamic address translation) 5003 para transformar direcciones de programa (direcciones virtuales) en

direcciones reales de memoria. Una DAT incluye habitualmente una memoria tampón de traducción anticipada (TLB, translation lookaside buffer) 5007 para almacenar en memoria caché las traducciones, de tal modo que los últimos accesos al bloque de memoria informática 5002 no requieren el retardo de la traducción de dirección. Habitualmente, se utiliza una memoria caché 5009 entre la memoria 5002 del ordenador y el procesador 5001. La memoria caché 5009 puede ser jerárquica con una gran memoria caché disponible para más de una CPU, y memorias caché menores, más rápidas (de menor nivel) entre la memoria caché grande y cada CPU. En algunas implementaciones, las memorias caché de menor nivel están divididas para proporcionar memorias caché de bajo nivel independientes, para búsqueda y carga de instrucciones y accesos a datos. En una realización, una instrucción es buscada y cargada desde la memoria 5002 mediante una unidad de búsqueda y carga de instrucciones 5004 a través de una memoria caché 5009. La instrucción se descodifica en una unidad 5006 de descodificación de instrucciones y es expedida (con otras instrucciones, en algunas realizaciones) a una o varias unidades 5008 de ejecución de instrucciones. Habitualmente se utilizan varias unidades 5008 de ejecución, por ejemplo una unidad de ejecución aritmética, una unidad ejecución de punto flotante y una unidad de ejecución de derivación. La instrucción es ejecutada por la unidad ejecución, accediendo a operandos a partir de registros especificados por instrucción o de memoria, según se requiera. Si se tiene que acceder (cargar o almacenar) un operando desde la memoria 5002, habitualmente una unidad de carga/almacenamiento 5005 gestiona el acceso bajo el control de la instrucción que se está ejecutando. Las instrucciones se pueden ejecutar en circuitos de hardware o en microcódigo interno (software inalterable), o mediante una combinación de ambos.

Tal como se ha indicado, un sistema informático incluye información en almacenamiento local (o principal), así como direccionamiento, protección, y registro de referencias y cambios. Algunos aspectos de direccionamiento incluyen el formato de las direcciones, el concepto de espacios de direcciones, los diversos tipos de direcciones y el modo en que un tipo de dirección es traducido a otro tipo de dirección. Parte del almacenamiento principal incluye ubicaciones de almacenamiento asignadas permanentemente. El almacenamiento principal proporciona al sistema almacenamiento de datos de rápido acceso, direccionable directamente. Tanto los datos como los programas se tienen que cargar en el almacenamiento principal (desde dispositivos de entrada) antes de que puedan ser procesados.

El almacenamiento principal puede incluir uno o varios almacenamientos menores de memoria tampón, de rápido acceso, en ocasiones denominadas memorias caché. Una memoria caché está habitualmente asociada físicamente con una CPU o un procesador de E/S. Los efectos, excepto en cuanto al rendimiento, de la construcción física y utilización de los distintos medios de almacenamiento no son generalmente observables por el programa.

Se pueden mantener memorias caché independientes para instrucciones y para operandos de datos. La información dentro de una memoria caché se mantiene en bytes contiguos en un contorno integral denominado un bloque de memoria caché o una línea de memoria caché (o línea, para abreviar). Un modelo puede proporcionar una instrucción de EXTRACT CACHE ATTRIBUTE (extraer atributos de la memoria caché) que devuelve el tamaño de una línea de memoria caché en bytes. Un modelo puede asimismo proporcionar instrucciones de PREFETCH DATA y PREFETCH DATA RELATIVE LONG (búsqueda y carga previas de datos, y búsqueda y carga previa de datos relativas largas), que efectúan la búsqueda y carga previas del almacenamiento en la memoria caché de datos o instrucciones, o la liberación de los datos desde la memoria caché.

El almacenamiento es considerado como una larga cadena horizontal de bits. Para la mayor parte de las operaciones, los accesos al almacenamiento avanzan en secuencia de izquierda a derecha. La cadena de bits se subdivide en unidades de ocho bits. Una unidad de ocho bits se denomina un byte, que es el bloque de construcción básico de todos los formatos de información. Cada ubicación de byte en el almacenamiento está identificado por un entero no negativo único, que es la dirección de dicha ubicación de byte o, simplemente, la dirección del byte. Las ubicaciones de byte adyacentes tienen direcciones consecutivas, empezando con 0 a la izquierda y avanzando en una secuencia de izquierda a derecha. Las direcciones son números enteros binarios sin signo y tienen 24, 31 ó 64 bits.

La información se transmite entre el almacenamiento y una CPU o un subsistema de canales en un byte, o un grupo de bytes, cada vez. Salvo que se especifique lo contrario, por ejemplo, en la z/Architecture®, un grupo de bytes en el almacenamiento es direccionado mediante el byte situado más al izquierda del grupo. El número de bytes en el grupo está implicado, o bien especificado explícitamente mediante la operación a llevar a cabo. Cuando se utiliza en una operación de CPU, un grupo de bytes se denomina un campo. Dentro de cada grupo de bytes, por ejemplo, en la z/Architecture®, los bits se numeran en una secuencia de izquierda a derecha. En la z/Architecture®, los bits situados más a la izquierda se denominan en ocasiones los bits "de orden superior" y los bits situados más a la derecha los bits "de orden inferior". Sin embargo, los números de bits no son direcciones de almacenamiento. Solamente se pueden direccionar bytes. Para funcionar con bits individuales en un byte en el almacenamiento, se accede al byte completo. Los bits en un byte están numerados de 0 a 7, de izquierda a derecha (por ejemplo, en la z/Architecture®). Los bits en una dirección se pueden numerar de 8 a 31 o de 40 a 63 para direcciones 24 bits, o de 1 a 31 o de 33 a 63 para direcciones de 31 bits; se numeran de 0 a 63 para direcciones de 64 bits. Dentro de cualquier otro formato de longitud fija de múltiples bytes, los bits que constituyen el formato se numeran consecutivamente empezando desde 0. Con propósitos de detección de errores, y preferentemente para corrección de errores, se pueden transmitir uno o varios bits de control con cada byte o con un grupo de bytes. Dichos bits de control son generados automáticamente por la máquina y no pueden ser controlados directamente por el programa.

Las capacidades de almacenamiento se expresan en un número de bytes. Cuando la longitud de un campo de almacenamiento-operando está implicada por el código de operación de la instrucción, se dice que el campo tiene una longitud fija, que puede ser de uno, dos, cuatro, ocho o dieciséis bytes. Pueden estar involucrados campos mayores para algunas instrucciones. Cuando la longitud de un campo de almacenamiento-operando no está implicada pero se indica explícitamente, se dice que el campo tiene una longitud variable. Los operandos de longitud variable pueden variar de longitud a incrementos de un byte (o con algunas instrucciones, en múltiplos de dos bytes u otros múltiplos). Cuando la información se sitúa en el almacenamiento, se sustituyen solamente los contenidos de aquellas ubicaciones de byte que están incluidas en el campo indicado, aunque la anchura del la trayectoria física al almacenamiento pueda ser mayor que la longitud del campo que está siendo almacenado.

Ciertas unidades de información tienen que estar en un límite integral en el almacenamiento. Se dice que un límite es integral para una unidad de información cuando su dirección de almacenamiento es un múltiplo de la longitud de la unidad en bytes. Se dan nombres especiales a los campos de 2, 4, 8 y 16 bytes en un límite integral. Media palabra es un grupo de dos bytes consecutivos en un límite de dos bytes, y es el bloque de construcción básico de las instrucciones. Una palabra es un grupo de cuatro bytes consecutivos en un límite de cuatro bytes. Una palabra doble es un grupo de ocho bytes consecutivos en un límite de ocho bytes. Una palabra cuádruple es un grupo de 16 bytes consecutivos en un límite de 16 bytes. Cuando las direcciones de almacenamiento designan palabras medias, palabras, palabras dobles y palabras cuádruples, la representación binaria de la dirección contiene uno, dos, tres o cuatro bits cero a la derecha, respectivamente. Las instrucciones tienen que estar en límites integrales de dos bytes. Los operandos de almacenamiento de la mayor parte de las instrucciones no tienen requisitos de alineamiento-límite.

En los dispositivos que implementan memorias caché independientes para instrucciones y operandos de datos, se puede experimentar un retardo significativo si el programa se almacena en una línea de memoria caché desde la que posteriormente se buscan y cargan instrucciones, independientemente de si el almacenamiento altera las instrucciones que se buscan y cargan a continuación.

En una realización, la invención se puede poner en práctica mediante software (en ocasiones, denominado código interno con licencia, software inalterable, microcódigo, milicódigo, picocódigo y similares, cualquiera de los cuales podría ser coherente con uno o varios aspectos de la presente invención). Haciendo referencia a la figura 7, el código de programa de software que incorpora uno o varios aspectos de la presente invención puede ser accedido por el procesador 5001 del sistema principal 5000 desde dispositivos 5011 de medios de almacenamiento a largo plazo, tal como una unidad de CD-ROM, una unidad de cintas o un disco duro. El código de programa de software puede estar incorporado en cualquiera de diversos medios conocidos para su utilización con el sistema de procesamiento de datos, tales como un disquete, un disco duro o un CD-ROM. El código se puede distribuir en dichos medios, o se puede distribuir a los usuarios desde una memoria de ordenador 5002 o un almacenamiento de un sistema informático sobre una red 5010 a otros sistemas informáticos para su utilización por los usuarios de dichos otros sistemas.

El código de programa de software incluye un sistema operativo que controla el funcionamiento y la interacción de los diversos componentes informáticos y de uno o varios programas de aplicación. El código de programa es normalmente paginado desde el dispositivo de medios de almacenamiento 5011 al almacenamiento informático 5002 relativamente más rápido, donde está disponible para su procesamiento por el procesador 5001. Las técnicas y procedimientos para incorporar código de programa de software en memoria, en medios físicos y/o en código de software de distribución por medio de redes son bien conocidos y no se discutirán más en la presente memoria. El código de programa, cuando es creado y almacenado en un medio tangible (que incluye, de forma no limitativa, módulos de memoria electrónica (RAM), memoria flash, discos compactos (CDs, Compact Discs), DVDs, cinta magnética y similares, se denomina a menudo un "producto de programa informático". El medio del producto de programa informático es habitualmente legible por un circuito de procesamiento, preferentemente en un sistema informático, para su ejecución por el circuito de procesamiento.

La figura 8 muestra una estación de trabajo o sistema de hardware de servidor, representativo, en el que se pueden practicar uno o varios aspectos de la presente invención. El sistema 5020 de la figura 8 comprende un sistema informático básico representativo 5021, tal como un ordenador personal, una estación de trabajo o un servidor, que incluye dispositivos periféricos opcionales. El sistema informático básico 5021 incluye uno o varios procesadores 5026 y un bus utilizado para conectar y permitir la comunicación entre el procesador o procesadores 5026 y los otros componentes del sistema 5021, de acuerdo con técnicas conocidas. El bus conecta el procesador 5026 con memoria 5025 y almacenamiento a largo plazo 5027 que puede incluir un disco duro (que incluye cualquiera de medios magnéticos, CD, DVD y memoria flash, por ejemplo) o una unidad de cinta, por ejemplo. El sistema 5021 puede incluir asimismo un adaptador de interfaz de usuario, que conecta el microprocesador 5026 por medio del bus a uno o varios dispositivos de interfaz, tales como un teclado 5024, un ratón 5023, una impresora/escáner 5030 y/u otros dispositivos de interfaz, que pueden ser cualquier dispositivo de interfaz de usuario, tal como una pantalla táctil, un panel de entrada digitalizada, etc. El bus conecta asimismo un dispositivo de visualización 5022, tal como un monitor o pantalla LCD, al microprocesador 5026 por medio de un adaptador de visualización.

El sistema 5021 puede comunicar con otros ordenadores o redes de ordenadores por medio de un adaptador de red apto para comunicación 5028 con una red 5029. Son ejemplos de adaptadores de red los canales de comunicaciones, anillo con paso de testigo, Ethernet o módems.

Alternativamente, el sistema 5021 puede comunicar utilizando una interfaz inalámbrica, tal como una tarjeta de CDPD (cellular digital packet data, datos por paquetes digitales celulares). El sistema 5021 puede estar asociado con dichos otros ordenadores en una red de área local (LAN) o una red de área extensa (WAN), o el sistema 5021 puede ser un cliente en una organización cliente/servidor con otro ordenador, etc. La totalidad de estas configuraciones, así como el hardware y software apropiados de comunicaciones, son conocidos en la técnica.

La figura 9 muestra una red de procesamiento de datos 5040 en la que se pueden practicar uno o varios aspectos de la presente invención. La red de procesamiento de datos 5040 puede incluir una serie de redes individuales, tal como una red inalámbrica y una red cableada, cada una de las cuales puede incluir una serie de estaciones de trabajo individuales 5041, 5042, 5043, 5044. Adicionalmente, tal como apreciarán los expertos en la materia, se pueden incluir una o varias LAN, donde una LAN comprende una serie de estaciones de trabajo inteligentes acopladas a un procesador principal.

También haciendo referencia a la figura 9, las redes pueden incluir asimismo servidores u ordenadores centrales, tal como un ordenador pasarela (servidor cliente 5046) o un servidor de aplicación (un servidor remoto 5048, que puede acceder a un repositorio de datos y puede asimismo ser accedido directamente desde una estación de trabajo 5045). Un ordenador pasarela 5046 sirve como un punto de entrada en cada red individual. Se requiere una pasarela cuando se conecta un protocolo de trabajo en red a otro. La pasarela 5046 puede estar acoplada preferentemente a otra red (por ejemplo, internet 5047) por medio de un enlace de comunicaciones. La pasarela 5046 puede estar asimismo acoplada directamente a una o varias estaciones de trabajo 5041, 5042, 5043, 5044 utilizando un enlace de comunicaciones. El ordenador pasarela se puede implementar utilizando un servidor IBM eServer™ System z® disponible en la firma International Business Machines Corporation.

Haciendo referencia simultáneamente a la figura 8 y la figura 9, un código de programación de software que puede realizar uno o varios aspectos de la presente invención puede ser accedido por el procesador 5026 del sistema 5020 desde medios de almacenamiento a largo plazo 5027, tal como una unidad de CD-ROM o un disco duro. El código de programación de software puede estar incorporado en cualquiera de diversos medios conocidos para su utilización con el sistema de procesamiento de datos, tales como un disquete, un disco duro o un CD-ROM. El código se puede distribuir en dichos medios, o se puede distribuir a los usuarios 5050, 5051 desde la memoria o el almacenamiento de un sistema informático sobre una red a otros sistemas informáticos para su utilización por los usuarios de dichos otros sistemas.

Alternativamente, el código de programación puede estar incorporado en la memoria 5025, y ser accedido por el procesador 5026 utilizando el bus del procesador. Dicho código de programación incluye un sistema operativo que controla el funcionamiento y la interacción de los diversos componentes informáticos, y de uno o varios programas de aplicación 5032. El código de programa es normalmente paginado desde el medio de almacenamiento 5027 a memoria de alta velocidad 5025, donde está disponible para su procesamiento por el procesador 5026. Las técnicas y procedimientos para incorporar código de programación de software en memoria, en medios físicos y/o en código de software de distribución por medio de redes son bien conocidos y no se discutirán más en la presente memoria. El código de programa, cuando es creado y almacenado en un medio tangible (que incluye, de forma no limitativa, módulos de memoria electrónica (RAM), memoria flash, discos compactos (CDs), DVDs, cinta magnética y similares), se denomina a menudo un "producto de programa informático". El medio de producto de programa informático es habitualmente legible mediante un circuito de procesamiento, preferentemente en un sistema informático, para su ejecución por el circuito de procesamiento.

La memoria caché que está más fácilmente disponible para el procesador (normalmente más rápida y más pequeña que otras memorias caché del procesador) es la memoria caché más baja (L1 o nivel uno) y el almacenamiento principal (memoria principal) es la memoria caché de mayor nivel (L3, si existen 3 niveles). La memoria caché de menor nivel se divide a menudo en una memoria caché de instrucciones (memoria caché-I) que contiene instrucciones de máquina que se tienen que ejecutar y una memoria caché de datos (memoria caché-D) que contiene operandos de datos.

Haciendo referencia a la figura 10, se representa una realización de procesador a modo de ejemplo, para el procesador 5026. Habitualmente, se utilizan uno o varios niveles de memoria caché 5053 para bloques de memoria de la memoria tampón con el fin de mejorar el rendimiento del procesador. La memoria caché 5053 es una memoria tampón de alta velocidad que contiene líneas de memoria caché de datos de memoria que es probable sean utilizados. Las líneas habituales de memoria caché son 64, 128 ó 256 bytes de datos de memoria. Las memorias caché son utilizadas a menudo para almacenar en memoria caché instrucciones en lugar de para almacenar en memoria caché datos. La coherencia de las memorias caché (sincronización de copias de las líneas en memoria y las memorias caché) se proporciona a menudo mediante varios algoritmos "espía" bien conocidos en la técnica. La memoria de almacenamiento principal 5025 de un sistema de procesador se denomina asimismo una memoria caché. En un sistema de procesador con 4 niveles de memoria caché 5053, el almacenamiento principal 5025 se denomina en ocasiones la memoria caché de nivel 5 (L5), dado que habitualmente es más rápida y contiene

solamente una parte del almacenamiento no volátil (DASD, cinta, etc.) que está disponible para un sistema informático. El almacenamiento principal 5025 "almacena en memoria caché" páginas de datos paginados dentro y fuera del almacenamiento principal 5025 mediante el sistema operativo.

Un contador de programa (contador de instrucciones) 5061 mantiene un seguimiento de la dirección de la instrucción actual que se tiene que ejecutar. Un contador de programa en un procesador z/Architecture® tiene 64 bits y se puede truncar a 31 ó 24 bits para soportar límites de direccionamiento anteriores. Un contador de programa está incorporado habitualmente en una PSW (program status word, palabra de estado de programa) de un ordenador, de tal modo que persiste durante cambio de contexto. De este modo, un programa en curso, que tiene un valor de contador de programa, puede ser interrumpido, por ejemplo, por el sistema operativo (cambio de contexto desde el entorno de programa al entorno del sistema operativo). La PSW del programa mantiene el valor del contador de programa mientras el programa no está activo, y el contador de programa (en la PSW) del sistema operativo se utiliza mientras se está ejecutando el sistema operativo. Habitualmente, el contador de programa se incrementa en una cantidad igual al número de bytes de la instrucción actual. Las instrucciones RISC (Reduced Instruction Set Computing, computación con conjunto de instrucciones reducidas) tienen habitualmente longitud fija mientras que las instrucciones CISC (Complex Instruction Set Computing, computación con conjunto de instrucciones complejas) tienen habitualmente longitud variable. Las instrucciones de la z/Architecture® de IBM son instrucciones CISC que tienen una longitud de 2, 4 ó 6 bytes. El contador de programa 5061 es modificado por una operación de cambio de contexto o bien por una operación de adopción de derivación de una instrucción de derivación, por ejemplo. En la operación de cambio de contexto, el actual valor del contador de programa se guarda en la palabra de estado de programa junto con otra información de estado sobre el programa que está en ejecución (tal como códigos de ejecución), y se carga un nuevo valor de contador de programa que apunta a una instrucción de un nuevo módulo de programa que se tiene que ejecutar. Se lleva a cabo una operación de derivación adoptada, con el fin de permitir que el programa adopte decisiones o realice un bucle en el interior del programa cargando el resultado de la instrucción de derivación en el contador de programa 5061.

Habitualmente, se utiliza una unidad 5055 de búsqueda y carga de instrucciones para buscar y cargar instrucciones en nombre del procesador 5026. La unidad de búsqueda y carga, busca y carga "siguientes instrucciones secuenciales", instrucciones objetivo de instrucciones de derivación adoptada, o bien primeras instrucciones de un programa después de un cambio de contexto. Las unidades modernas de búsqueda y carga de instrucciones utilizan a menudo técnicas de búsqueda y carga previas para buscar y cargar previamente, de manera especulativa, instrucciones en base a la probabilidad de que las instrucciones buscadas y cargadas previamente puedan ser utilizadas. Por ejemplo, una unidad de búsqueda y carga puede buscar y cargar 16 bytes de instrucción que incluyen la siguiente instrucción secuencial, y bytes adicionales de instrucciones secuenciales adicionales.

Las instrucciones buscadas y cargadas son a continuación ejecutadas por el procesador 5026. En una realización, la instrucción o instrucciones buscadas y cargadas se pasan a una unidad de expedición 5056 de la unidad de búsqueda y carga. La unidad de expedición descodifica la instrucción o instrucciones y envía información sobre la instrucción o instrucciones descodificadas a unidades apropiadas 5057, 5058, 5060. Una unidad de ejecución 5057 recibirá habitualmente información sobre instrucciones aritméticas descodificadas desde la unidad de búsqueda y captura de instrucciones 5055, y llevará a cabo operaciones aritméticas sobre operandos de acuerdo con el código de operación de la instrucción. Los operandos son proporcionados a la unidad de ejecución 5057, preferentemente desde la memoria 5025, desde registros diseñados 5059 o bien desde un campo contiguo a la instrucción que está en ejecución. Los resultados de la ejecución, cuando son almacenados, se almacenan en la memoria 5025, en registros 5059 o bien en otro hardware de la máquina (tal como registros de control, registros PSW y similares).

Un procesador 5026 tiene habitualmente una o varias unidades 5057, 5058, 5060 para ejecutar la función de la instrucción. Haciendo referencia a la figura 11A, una unidad de ejecución 5057 puede comunicar con registros generales diseñados 5059, una unidad de descodificación/expedición 5056, una unidad 5060 de almacenamiento de carga y otras unidades de procesador 5065 por medio de lógica de interconexión 5071. Una unidad de ejecución 5057 puede utilizar varios circuitos de registros 5067, 5068, 5069 para contener información sobre la que operará la unidad aritmética lógica (ALU, arithmetic logic unit) 5066. La ALU lleva a cabo operaciones aritméticas tales como sumar, restar, multiplicar y dividir, así como funciones lógicas tales como y, o y o exclusiva (XOR), rotar y desplazar. Preferentemente, la ALU soporta operaciones especializadas que dependen del diseño. Otros circuitos pueden proporcionar otras instalaciones diseñadas 5072 que incluyen códigos de condición y lógica de soporte de recuperación, por ejemplo. Habitualmente, el resultado de una operación ALU se retiene en un circuito de registro de salida 5070 que puede enviar el resultado a diversas funciones de procesamiento. Hay muchas disposiciones de unidades de procesador, estando la presente descripción destinada solamente a proporcionar una comprensión representativa de una realización.

Una instrucción ADD, por ejemplo, sería ejecutada en una unidad de ejecución 5057 con funcionalidad aritmética y lógica, mientras que una instrucción de punto flotante, por ejemplo, sería ejecutada en una ejecución de punto flotante con capacidad de punto flotante especializada. Preferentemente, una unidad de ejecución funciona sobre operandos identificados por una instrucción, llevando a cabo una función definida por código de operación sobre los operandos. Por ejemplo, una instrucción ADD puede ser ejecutada por una unidad de ejecución 5057 sobre operandos encontrados en dos registros 5059 identificados mediante campos de registro de la instrucción.

La unidad de ejecución 5057 lleva a cabo la suma aritmética sobre dos operandos y almacena el resultado en un tercer operando, donde el tercer operando puede ser un tercer registro o uno de los dos registros de origen. La unidad de ejecución utiliza preferentemente una unidad aritmética lógica (ALU) 5066 que puede llevar a cabo diversas funciones lógicas tales como desplazar, rotar, And (y), Or (o) y XOR así como varias funciones algebraicas incluyendo cualquiera de sumar, restar, multiplicar, dividir. Algunas ALU 5066 están diseñadas para operaciones escalares, y algunas para punto flotante. Los datos pueden ser Big Endian (donde el byte menos significativo está en la dirección de byte más alta) o Little Endian (donde el byte menos significativo está en la dirección de byte más baja), dependiendo de la arquitectura. La z/Architecture® de IBM es Big Endian. Los campos con signo pueden ser signo y magnitud, con complemento a uno o complemento a dos, dependiendo de la arquitectura. Un número con complemento a dos es ventajoso porque la ALU no tiene que diseñar una capacidad de resta, dado que tanto un valor negativo como un valor positivo en complemento a dos requiere solamente una suma con la ALU. Los números se escriben normalmente abreviados, donde un campo de 12 bits define una dirección de un bloque de 4096 bytes y se describe normalmente como un bloque de 4 Kbyte (Kilobyte), por ejemplo.

Haciendo referencia a la figura 11B, se envía habitualmente información de instrucciones de derivación para ejecutar una instrucción de derivación, a una unidad de derivación 5058, que a menudo utiliza un algoritmo de predicción de derivación, tal como una tabla de historial de derivación 5082, para predecir el resultado de la derivación antes de que se completen otras operaciones condicionales. El objetivo de la actual instrucción de derivación será buscado y cargado, y ejecutado especulativamente antes de que se completen las operaciones condicionales. Cuando las operaciones condicionales se han completado, las instrucciones de derivación ejecutadas especulativamente se completan o bien se desechan, en base a las condiciones de la operación condicional y al resultado especulado. Una típica instrucción de derivación puede comprobar códigos de condición y derivar a una dirección objetivo si el código de condición satisface el requisito de derivación de la instrucción de derivación, se puede calcular una dirección objetivo en base a varios números que incluyen números encontrados en campos de registros o en un campo contiguo de la instrucción, por ejemplo. La unidad de derivación 5058 puede utilizar una ALU 5074 que tiene una serie de circuitos 5075, 5076, 5077 de registros de entrada y un circuito 5080 de registros de salida. La unidad de derivación 5058 puede comunicar con registros generales 5059, con la unidad de descodificación expedición 5056 o con otros circuitos 5073, por ejemplo.

La ejecución de un grupo de instrucciones puede ser interrumpida por diversas razones que incluyen un cambio de contexto iniciado por un sistema operativo, un error o excepción de programa que provoque un cambio de contexto, una señal de interrupción de E/S que provoque un cambio de contexto o una actividad multihilo de una serie de programas (en un entorno multihilo), por ejemplo. Preferentemente, una acción de cambio de contexto guarda información de estado sobre un programa actualmente en ejecución, y a continuación carga información de estado sobre otro programa que se está invocando. La información de estado se puede guardar en registros de hardware o en memoria, por ejemplo. La información de estado comprende preferentemente un valor de contador de programa que apunta a una siguiente instrucción que se tiene que ejecutar, códigos de condición, información de traducción de memoria y contenidos de registros diseñados. Una actividad de cambio de contexto puede ser ejercitada por circuitos de hardware, programas de aplicación, programas de sistema operativo o código de software inalterable (microcódigo, picocódigo o código interno con licencia (LIC)), por separado o en combinación.

Un procesador accede a operandos de acuerdo con procedimientos definidos por instrucciones. La instrucción puede proporcionar un operando contiguo utilizando el valor de una parte de la instrucción, puede proporcionar uno o varios campos de registro que apuntan explícitamente a registros de propósito general o bien a registros de propósito especial (registros de punto flotante, por ejemplo). La instrucción puede utilizar registros implicados, identificados mediante un campo de código de operación como operandos. La instrucción puede utilizar ubicaciones de memoria para operandos. Una ubicación de memoria de un operando puede ser proporcionada por un registro, un campo contiguo o una combinación de registros y campos contiguos, tal como se ejemplifica mediante la función de desplazamiento largo de z/Architecture®, en la que la instrucción define un registro de base, un registro de índice y un campo contiguo (campo de desplazamiento) que se suman entre sí para proporcionar la dirección del operando en memoria, por ejemplo. En la presente memoria, una ubicación implica habitualmente una ubicación en memoria principal (almacenamiento principal), salvo que se indique lo contrario.

Haciendo referencia a la figura 11C, un procesador accede al almacenamiento utilizando una unidad de carga/almacenamiento 5060. La unidad de carga/almacenamiento 5060 puede llevar a cabo una operación de carga obteniendo la dirección del operando objetivo en memoria 5053, y cargando el operando en un registro 5059 u otra ubicación de memoria 5053, o puede llevar a cabo una operación de almacenamiento obteniendo la dirección del operando objetivo en memoria 5053 y almacenando datos obtenidos desde un registro 5059 u otra ubicación de memoria 5053 en la ubicación del operando objetivo en la memoria 5053. La unidad de carga/almacenamiento 5060 puede ser especulativa y puede acceder a la memoria en una secuencia que no está en orden en relación con la secuencia de instrucciones, aunque la unidad de carga/almacenamiento 5060 tiene que mantener para los programas la apariencia de que las instrucciones se han ejecutado por orden. Una unidad de carga/almacenamiento 5060 puede comunicar con registros generales 5059, la unidad de descodificación/expedición 5056, la interfaz de memoria caché/memoria 5053 u otros elementos 5083 y comprende varios circuitos de registros, ALUs 5085 y lógica de control 5090 para calcular direcciones de almacenamiento y para proporcionar secuenciación canalizada con el fin de mantener las operaciones en orden. Algunas operaciones pueden no estar en orden pero la unidad de

carga/almacenamiento proporciona funcionalidad para hacer que las operaciones que no están en orden aparezcan para el programa como si se hubieran llevado a cabo por orden, tal como es conocido en la técnica.

Preferentemente, las direcciones que "ve" un programa de aplicación se denominan a menudo direcciones virtuales. La direcciones virtuales se denominan en ocasiones "direcciones lógicas" y "direcciones efectivas". Estas direcciones virtuales son virtuales porque son redirigidas a una ubicación de memoria física mediante una de diversas tecnologías de traducción dinámica de direcciones (DAT) que incluyen, de forma no limitativa, añadir simplemente un prefijo a una dirección virtual con un valor de desplazamiento, traducir la dirección virtual por medio de una o varias tablas de traducción, comprendiendo preferentemente las tablas de traducción por lo menos una tabla de segmentos y una tabla de páginas, por separado o en combinación, preferentemente, teniendo la tabla de segmentos una entrada que apunta a la tabla de páginas. En la z/Architecture®, se proporciona una jerarquía de traducción que incluye una primera tabla de zonas, una segunda tabla de zonas, una tercera tabla de zonas, una tabla de segmentos y una tabla de páginas opcional. El rendimiento de la traducción de direcciones se mejora a menudo utilizando una memoria tampón de traducción anticipada (TLB) que comprende entradas que mapean una dirección virtual a una ubicación de memoria física asociada. Las entradas se crean cuando la DAT traduce una dirección virtual utilizando las tablas de traducción. La utilización posterior de la dirección virtual puede a continuación utilizar la entrada de la TLB rápida en lugar de los lentos accesos a tablas de traducción secuencial. El contenido de la TLB puede ser gestionado por diversos algoritmos de sustitución incluyendo LRU (Least Recently Used, menos usado recientemente).

En caso de que el procesador sea un procesador de un sistema de múltiples procesadores, cada procesador tiene la responsabilidad de mantener estructuras compartidas, tal como E/S, memorias caché, TLBs y memoria, interconectadas por coherencia. Habitualmente, se utilizarán tecnologías "espía" en el mantenimiento de la coherencia de memorias caché. En un entorno espía, la línea de memoria caché se puede marcar como estando en alguno de un estado compartido, un estado exclusivo, un estado modificado, un estado no válido y similares, para facilitar la compartición.

Las unidades de E/S 5054 (figura 10) proporcionan al procesador medios para su acoplamiento a dispositivos periféricos incluyendo cinta, disco, impresoras, pantallas y redes, por ejemplo. A menudo, las unidades de E/S se presentan al programa informático mediante controladores de software. En los ordenadores centrales, tal como el System z® de IBM®, los adaptadores de canal y los adaptadores de sistema abierto son unidades de E/S del ordenador central que proporcionan las comunicaciones entre el sistema operativo y los dispositivos periféricos.

Además, otros tipos de entornos informáticos se pueden beneficiar de uno o varios aspectos de la presente invención. A modo de ejemplo, un entorno puede incluir un emulador (por ejemplo, software u otros mecanismos de emulación), en el que se emula una arquitectura particular (incluyendo, por ejemplo, ejecución de instrucciones, funciones diseñadas, tal como traducción de direcciones, y registros diseñados) o un subconjunto de la misma (por ejemplo, un sistema informático nativo que tiene un procesador y memoria). En dicho entorno, una o varias funciones emuladas del emulador pueden implementar uno o varios aspectos de la presente invención, aunque un ordenador que ejecute el emulador pueda tener una arquitectura diferente a las capacidades que se están emulando. A modo de ejemplo, en modo emulación, la operación o instrucción específica que se está emulando es descodificada, y se construye una función de emulación apropiada con el fin de implementar la operación o instrucción individual.

En un entorno de emulación, un ordenador principal incluye, por ejemplo, una memoria para almacenar instrucciones y datos; una unidad de búsqueda y carga de instrucciones para buscar y cargar instrucciones desde la memoria y, opcionalmente, proporcionar almacenamiento en memoria tampón local para la instrucción buscada y cargada; una unidad de descodificación para recibir las instrucciones buscadas y cargadas, y para determinar el tipo de instrucciones que han sido buscadas y cargadas; y una unidad de ejecución de instrucciones para ejecutar las instrucciones. La ejecución puede incluir cargar datos en un registro desde la memoria; almacenar datos de nuevo en la memoria desde un registro; o llevar a cabo algún tipo de operación aritmética o lógica, según determine la unidad de descodificación. En un ejemplo, cada unidad está implementada en software. Por ejemplo, las operaciones que se están llevando a cabo por las unidades son implementadas como una o varias subrutinas dentro del software del emulador.

Más particularmente, en un ordenador central, las instrucciones de máquina diseñadas son utilizadas por programadores, hoy en día usualmente programadores en "C", a menudo por medio de una aplicación de compilador. Estas instrucciones almacenadas en el medio de almacenamiento pueden ser ejecutadas de forma nativa en un servidor IBM® de z/Architecture®, o alternativamente en máquinas que ejecutan otras arquitecturas. Pueden ser emuladas en los servidores centrales IBM® existentes y futuros, y en otras máquinas de IBM® (por ejemplo, servidores Power Systems y System x® Servers). Pueden ser ejecutadas en máquinas que funcionan con Linux sobre una amplia variedad de máquinas que utilizan hardware fabricado por IBM®, Intel®, AMD™ y otros. Además de su ejecución en dicho hardware bajo una z/Architecture®, se puede utilizar Linux así como máquinas que utilicen emulación mediante Hercules, UMX o FSI (Fundamental Software, Inc), donde la ejecución es generalmente en modo emulación. En modo emulación, un software de emulación es ejecutado por un procesador nativo para emular la arquitectura de un procesador emulado.

El procesador nativo ejecuta habitualmente software de emulación que comprende software inalterable o bien un sistema operativo nativo para llevar a cabo la emulación del procesador emulado. El software de emulación es responsable de la búsqueda y carga, y la ejecución de instrucciones de la arquitectura de procesador emulada. El software de emulación mantiene un contador de programa emulado para mantener un seguimiento de los límites de instrucción. El software de emulación puede buscar y cargar una o varias instrucciones de máquina emuladas cada vez, y transformar dichas una o varias instrucciones de máquina emuladas en un grupo correspondiente de instrucciones de máquina nativas para su ejecución por el procesador nativo. Estas instrucciones transformadas pueden ser almacenadas en memoria caché, de tal modo que se puede conseguir una transformación más rápida. No obstante, el software de emulación tiene que mantener las reglas de diseño del diseño del procesador emulado de tal modo que garantice que los sistemas operativos y las aplicaciones escritas para el procesador emulado funcionen correctamente. Además, el software de emulación tiene que proporcionar recursos identificados por la arquitectura del procesador emulado incluyendo, de forma no limitativa, registros de control, registros de propósito general, registros de punto flotante, función de traducción dinámica de direcciones incluyendo tablas de segmentos y tablas de páginas, por ejemplo, mecanismos de interrupción, mecanismos de cambio de contexto, relojes de la hora del día (TOD, Time of Day) e interfaces diseñados, a sistemas de E/S de tal modo que un sistema operativo o un programa de aplicación diseñado para funcionar en el procesador emulado pueda funcionar en el procesador nativo que tiene el software de emulación.

Una instrucción específica que está siendo emulada es descodificada, y se invoca una subrutina para llevar a cabo la función de instrucción individual. Una función de software de emulación que emula una función de un procesador emulado se implementa, por ejemplo, en un controlador o subrutina "C", o en algún otro procedimiento para proporcionar un controlador para el hardware específico, tal como sabrá un experto en la materia después de comprender la descripción de la realización preferida. Diversas patentes de emulación de software y de hardware que incluyen, de forma no limitativa la patente de invención U.S.A. número 5.551.013, titulada "Multiprocessor for Hardware Emulation", de Beausoleil et al.; y la patente de invención U.S.A. número 6.009.261, titulada "Preprocessing of Stored Target Routines for Emulating Incompatible Instructions on a Target Processor", de Scalzi et al; y la patente de invención U.S.A. número 5.574.873, titulada "Decoding Guest Instruction to Directly Access Emulation Routines that Emulate the Guest Instructions", de Davidian et al; y la patente de invención U.S.A. número 6.308.255, titulada "Symmetrical Multiprocessing Bus and Chipset Used for Coprocessor Support Allowing Non-Native Code to Run in a System", de Gorishek et al; y la patente de invención U.S.A. número 6.463.582, titulada "Dynamic Optimizing Object Code Translator for architecture Emulation and Dynamic Optimizing Object Code Translation method", de Lethin et al; y la patente de invención U.S.A. número 5.790.825, titulada "Method for Emulating Guest Instructions on a host Computer Through Dynamic Recompilation of host Instructions", de Eric Traut, muestran diversas maneras conocidas disponibles para los expertos en la materia para conseguir la emulación de un formato de instrucciones diseñado para una máquina diferente, para una maquina objetivo.

En la figura 12, se proporciona un ejemplo de un sistema de ordenador principal emulado 5092 que emula un sistema de ordenador principal 5000' de una arquitectura de ordenador principal. En el sistema de ordenador principal emulado 5092, el ordenador principal (CPU) 5091 es un procesador principal emulado (o un procesador principal virtual) y comprende un procesador de emulación 5093 que tiene una arquitectura de conjunto de instrucciones nativas diferente a la del procesador 5091 del ordenador principal 5000'. El sistema de ordenador principal emulado 5092 tiene memoria 5094 accesible para el procesador de emulación 5093. En el ejemplo de realización, la memoria 5094 está dividida en una parte de memoria 5096 de ordenador principal y una parte de rutinas 5097 de emulación. La memoria 5096 de ordenador principal está disponible para los programas del ordenador principal emulado 5092, de acuerdo con la arquitectura del ordenador principal. El procesador de emulación 5093 ejecuta instrucciones nativas de un conjunto de instrucciones diseñadas de una arquitectura diferente a la del procesador emulado 5091, las instrucciones nativas obtenidas de la memoria 5097 de rutinas de emulación, y puede acceder a una instrucción principal para su ejecución desde un programa en la memoria 5096 del ordenador principal utilizando una o varias instrucciones obtenidas en una rutina de secuencia y acceso/descodificación que puede descodificar la instrucción o instrucciones principales accedidas, para determinar una rutina de instrucción de ejecución nativa con el fin de emular la función de la instrucción principal accedida. Otras capacidades que se definen para la arquitectura del sistema de ordenador principal 5000' pueden ser emuladas por rutinas de capacidades diseñadas, incluyendo capacidades tales como registros de propósito general, registros de control, traducción dinámica de direcciones y soporte de subsistema de E/S y memoria caché del procesador, por ejemplo. Las rutinas de emulación pueden asimismo aprovechar las funciones disponibles en el procesador de emulación 5093 (tal como registros generales y traducción dinámica de direcciones virtuales) para mejorar el rendimiento de las rutinas de emulación. Se puede proporcionar asimismo hardware especial y motores sin carga para ayudar al procesador 5093 a emular la función del ordenador principal 5000'. La terminología utilizada en la presente memoria tiene el único propósito de describir realizaciones particulares y no está destinada a limitar la invención. Tal como se utilizan en la presente memoria, las formas singulares "un", "una", y "el" y "la" están destinadas a incluir también formas plurales, salvo que el contexto indique claramente lo contrario. Se comprenderá además que los términos "comprende" y/o "que comprende", cuando se utilizan en esta memoria descriptiva, especifican la presencia de características, números enteros, etapas, operaciones, elementos y/u otros componentes indicados, pero no excluyen la presencia o adición de una o varias otras características, números enteros, etapas, operaciones, elementos, componentes y/o grupos de los mismos.

Las correspondientes estructuras, materiales, acciones y equivalentes de todos los medios o etapas más los elementos de función en las siguientes reivindicaciones, donde aparezcan, están destinadas a incluir cualquier estructura, material o acción para llevar a cabo la función en combinación con otros elementos reivindicados, según se reivindique específicamente. La descripción de uno o varios aspectos de la presente invención se ha presentado con propósitos de ilustración y descripción, pero no está destinada a ser exhaustiva ni está limitada a la invención en la forma dada a conocer. Resultarán evidentes a los expertos en la materia muchas modificaciones y variaciones, sin apartarse del alcance de la invención. La realización se ha elegido y descrito para explicar mejor los principios de la invención y la aplicación práctica, y para permitir a otros expertos en la materia comprender la invención para diversas realizaciones con diversas modificaciones cuando son adecuadas para el uso particular contemplado.

REIVINDICACIONES

1. Un procedimiento para facilitar las comunicaciones en un entorno de comunicaciones que comprende un emisor (300), un receptor (310) y software inalterable (350) de un procesador, en el que el emisor y el receptor son espacios de memoria aislados, y en el que tanto el emisor como el receptor funcionan bajo el control del software inalterable dentro de una única máquina física, teniendo el software inalterable acceso a los espacios de memoria aislados del emisor y del receptor, comprendiendo dicho procedimiento:

recibir mediante el software inalterable (350), desde el emisor (300), una solicitud de transferencia de datos para enviar datos al receptor (310), comprendiendo la solicitud de transferencia de datos:

punteros a tampones de memoria que se tienen que enviar en el espacio de memoria aislado del emisor; y

un bloque de funcionamiento de autorización previa (QAOB), indicando el bloque de funcionamiento de autorización previa (QAOB) una autorización previa al software inalterable para que el software inalterable pueda llevar a cabo una transferencia de datos de manera asíncrona;

en respuesta a la recepción de la solicitud de transferencia de datos, determinar (506) mediante el software inalterable si el receptor tiene una memoria tampón vacía para poder recibir los datos;

en respuesta a que el receptor puede recibir los datos, transferir (508) los datos de manera síncrona (508);

en respuesta a que el receptor no puede recibir actualmente los datos debido a que no hay una memoria tampón vacía en el receptor:

transformar automáticamente, mediante el software inalterable, la transferencia de datos de una solicitud síncrona a una solicitud asíncrona, en base al bloque de funcionamiento de autorización previa (QAOB) que se proporciona en la solicitud de transferencia de datos, donde la solicitud de transferencia de datos se guarda en el bloque de funcionamiento de autorización previa (QAOB) (510), el bloque de funcionamiento de autorización previa (QAOB) se pone en cola (512) en una cola TPQ (420) del receptor, y se pasa el control al emisor de tal modo que el emisor se puede configurar inmediatamente para su siguiente transferencia de datos;

recibir otra solicitud desde el emisor, la otra solicitud para ser enviada a otro receptor de manera síncrona antes de la compleción del envío de la solicitud al receptor de manera asíncrona;

determinar (514), mediante el software inalterable, si el receptor tiene una memoria tampón vacía para poder recibir ahora los datos; y

en respuesta a la determinación de que el receptor tiene una memoria tampón vacía para poder recibir ahora los datos (514), determinar (516) mediante el software inalterable si hay un bloque de funcionamiento de autorización previa (QAOB) en cola en la cola (TPQ) del receptor y, en respuesta a determinar que hay un bloque de funcionamiento de autorización previa (QAOB) en cola en la cola (TPQ) del receptor, transferir (518) los datos al receptor utilizando el bloque de funcionamiento de autorización previa (QAOB).

2. El procedimiento según la reivindicación 1, en el que determinar si el receptor tiene una memoria tampón vacía para recibir los datos comprende determinar que el receptor está retardado para poder recibir los datos en el momento en el que los datos están siendo enviados.

3. Un producto de programa informático para facilitar las comunicaciones en un entorno de comunicaciones, comprendiendo dicho producto de programa informático: un medio de almacenamiento legible por ordenador, legible mediante un circuito de procesamiento, e instrucciones de almacenamiento para su ejecución por el circuito de procesamiento con el fin de llevar a cabo un procedimiento según la reivindicación 1 o la reivindicación 2.

4. El producto de programa informático según la reivindicación 3 cuando depende de la reivindicación 2, en el que bloque de funcionamiento de autorización previa (QAOB) comprende un bloque de memoria para el seguimiento de la transferencia de datos asíncrona.

5. El producto de programa informático según la reivindicación 3, en el que el procedimiento comprende además indicar (520) la compleción del envío al emisor.

6. El producto de programa informático según la reivindicación 5, en el que indicar (520) la compleción comprende publicar una dirección del bloque de funcionamiento de autorización previa (QAOB) en una cola de compleción en una memoria accesible para el emisor.

7. El producto de programa informático según la reivindicación 6, en el que el procedimiento comprende además generar una interrupción para el emisor, en respuesta a la publicación de la dirección en la cola de compleción.

8. El producto de programa informático según la reivindicación 3, en el que el procedimiento comprende además recibir una o varias solicitudes adicionales del emisor antes de la compleción del envío de los datos de manera asíncrona al receptor, donde los datos de dichas una o varias solicitudes adicionales pueden ser enviados de manera síncrona o de manera asíncrona a uno o varios receptores.

9. El producto de programa informático según la reivindicación 3, en el que el procedimiento comprende además mantener el orden de llegada de datos al receptor, en respuesta a la transformación de la transferencia de datos síncrona en la transferencia de datos asíncrona.

10. Un sistema informático para facilitar las comunicaciones en un entorno de comunicaciones que comprende un emisor (300), un receptor (310) y software inalterable (350) de un procesador, en el que el emisor y el receptor son espacios de memoria aislados, y en el que tanto el emisor como el receptor funcionan bajo el control del software inalterable dentro de una única máquina física, teniendo el software inalterable acceso a los espacios de memoria aislados del emisor y del receptor, comprendiendo dicho sistema informático:

una memoria (5025); y

un procesador (5026) en comunicación con la memoria, en el que el sistema informático está configurado para llevar a cabo un procedimiento, comprendiendo dicho procedimiento:

recibir mediante el software inalterable, desde el emisor (300), una solicitud de transferencia de datos para enviar datos al receptor (310), comprendiendo la solicitud de transferencia de datos: punteros a tampones de memoria que se tienen que enviar en el espacio de memoria aislado del emisor; y un bloque de funcionamiento de autorización previa (QAOB), indicando el bloque de funcionamiento de autorización previa (QAOB) una autorización previa al software inalterable para que el software inalterable lleve a cabo una transferencia de datos de manera asíncrona;

en respuesta a la recepción de la solicitud de transferencia de datos, determinar (506) mediante el software inalterable si el receptor tiene una memoria tampón vacía para poder recibir los datos;

en respuesta a que el receptor puede recibir los datos, transferir (508) los datos de manera síncrona (508);

en respuesta a que el receptor no puede recibir actualmente los datos debido a que no hay una memoria tampón vacía en el receptor, transformar automáticamente mediante el software inalterable la transferencia de datos de una solicitud síncrona a una solicitud asíncrona, en base al bloque de funcionamiento de autorización previa (QAOB) que se proporciona en la solicitud de transferencia de datos, donde la solicitud de transferencia de datos se guarda en el bloque de funcionamiento de autorización previa (QAOB) (510), el bloque de funcionamiento de autorización previa (QAOB) se pone en cola (512) en una cola TPQ (420) del receptor, y se pasa el control al emisor, de tal modo que el emisor se puede configurar inmediatamente para su siguiente transferencia de datos;

recibir otras solicitud desde el emisor, la otra solicitud para ser enviada a otro receptor de manera síncrona antes de la compleción del envío de la solicitud al receptor de manera asíncrona;

determinar (514), mediante el software inalterable, si el receptor tiene una memoria tampón vacía para poder recibir ahora los datos; y

en respuesta a la determinación de que el receptor tiene una memoria tampón vacía para poder recibir ahora los datos (514), determinar (518) mediante el software inalterable si hay un bloque de funcionamiento de autorización previa (QAOB) en cola en la cola (TPQ) del receptor y, en respuesta a determinar que hay un bloque de funcionamiento de autorización previa (QAOB) en cola en la cola (TPQ) del receptor, transferir (518) los datos al receptor utilizando el bloque de funcionamiento de autorización previa (QAOB).

11. El sistema informático según la reivindicación 10, en el que determinar si el receptor tiene un bufete vacío para poder recibir los datos comprende determinar que el receptor está retardado en poder recibir los datos en el momento en el que los datos se están enviando.

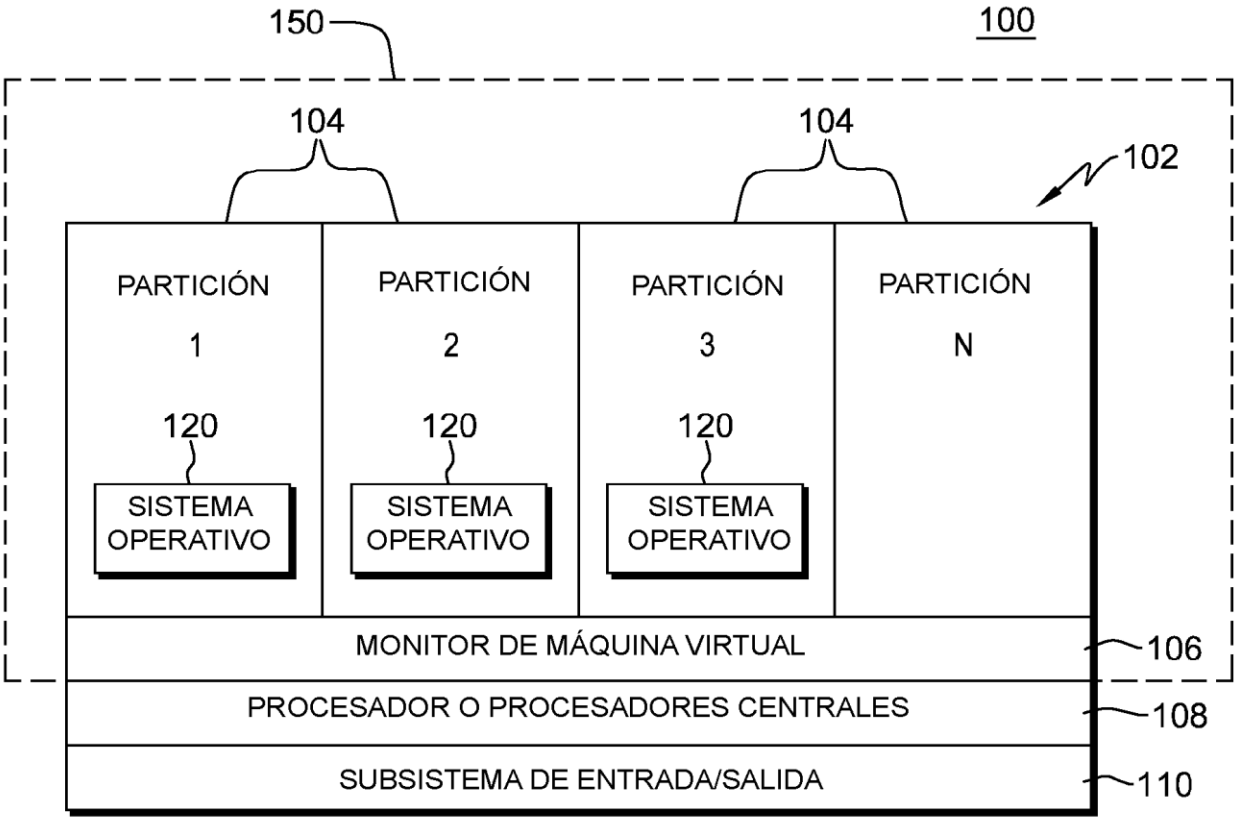
150
100
104
104
102
PARTICIÓN 1
PARTICIÓN 2
PARTICIÓN 3
PARTICIÓN N
120
120
120
SISTEMA OPERATIVO
SISTEMA OPERATIVO
SISTEMA OPERATIVO
MONITOR DE MÁQUINA VIRTUAL
106
PROCESADOR O PROCESADORES CENTRALES
108
SUBSISTEMA DE ENTRADA/SALIDA
110

FIG. 1

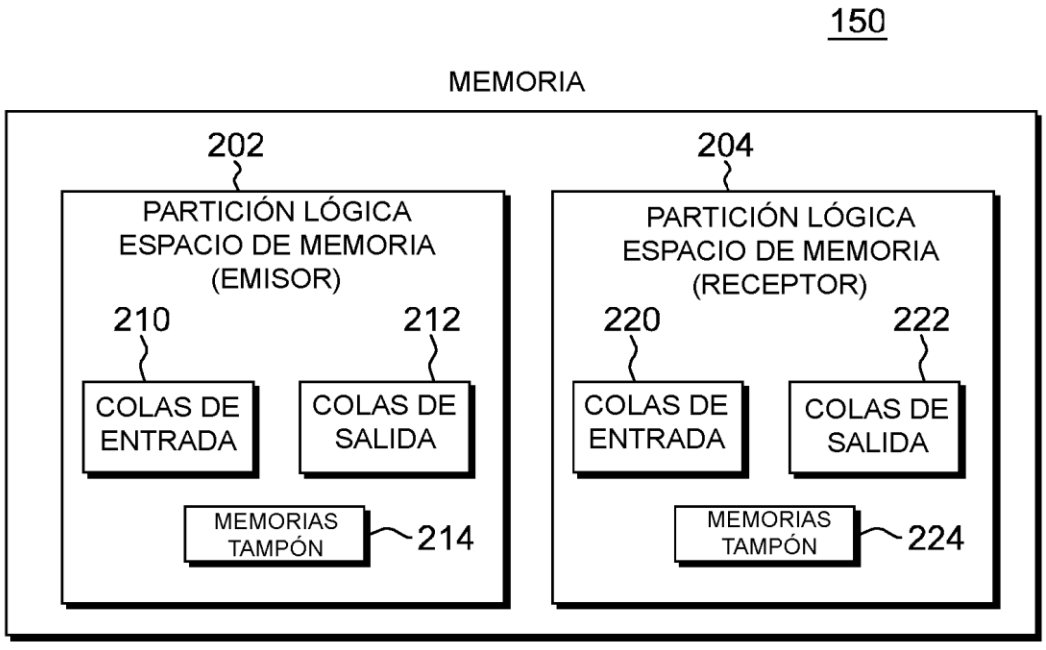
150
MEMORIA
202
204
PARTICIÓN LÓGICA ESPACIO DE MEMORIA (EMISOR)
PARTICIÓN LÓGICA ESPACIO DE MEMORIA (RECEPTOR)
210
212
220
222
COLAS DE ENTRADA
COLAS DE SALIDA
COLAS DE ENTRADA
COLAS DE SALIDA
MEMORIAS TAMPÓN
214
MEMORIAS TAMPÓN
224

FIG. 2

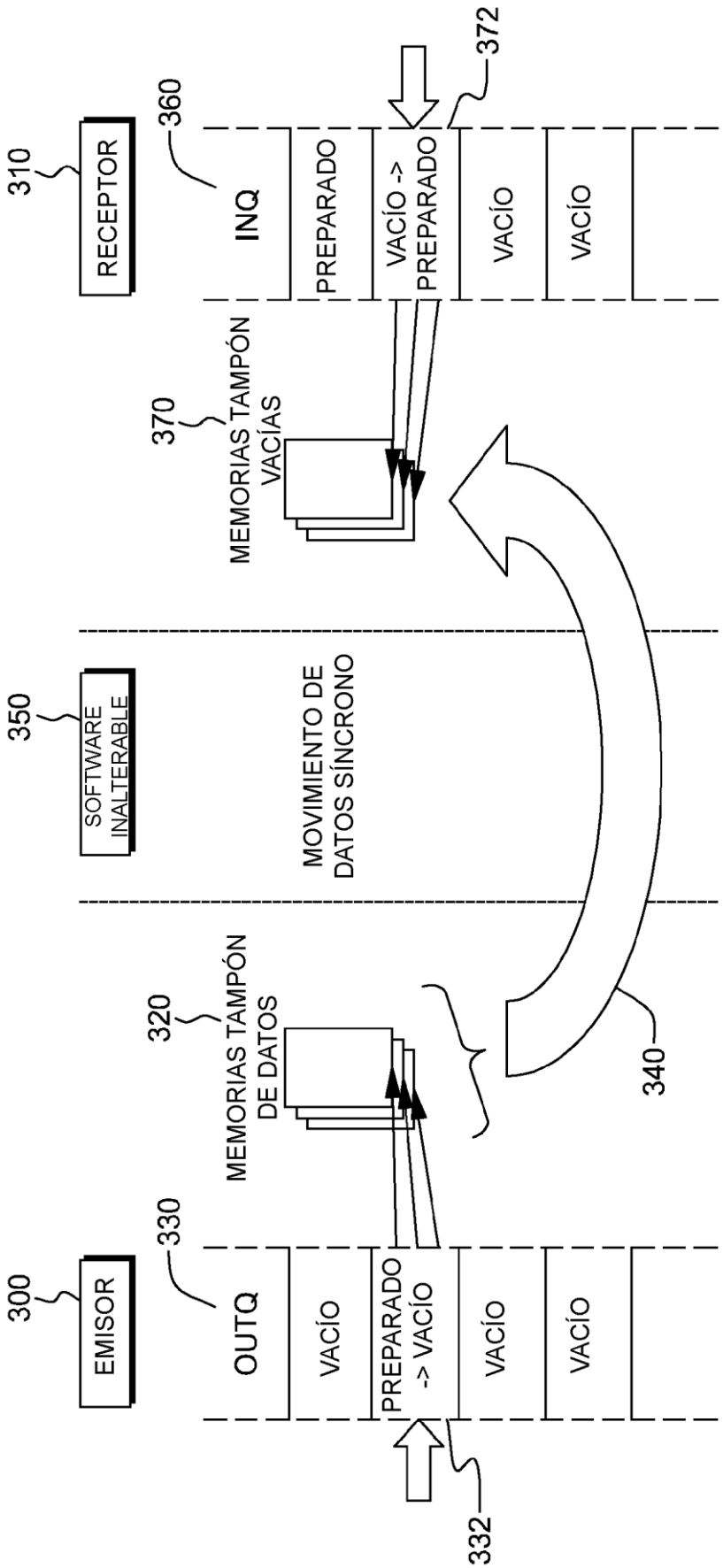
310
RECEPTOR
360
INQ
PREPARADO
VACÍO -> PREPARADO
VACÍO
VACÍO
372
370
MEMORIAS TAMPÓN VACÍAS
350
SOFTWARE INALTERABLE
MOVIMIENTO DE DATOS SÍNCRONO
320
MEMORIAS TAMPÓN DE DATOS
340
300
EMISOR
330
OUTQ
VACÍO
PREPARADO -> VACÍO
VACÍO
VACÍO
332

FIG. 3

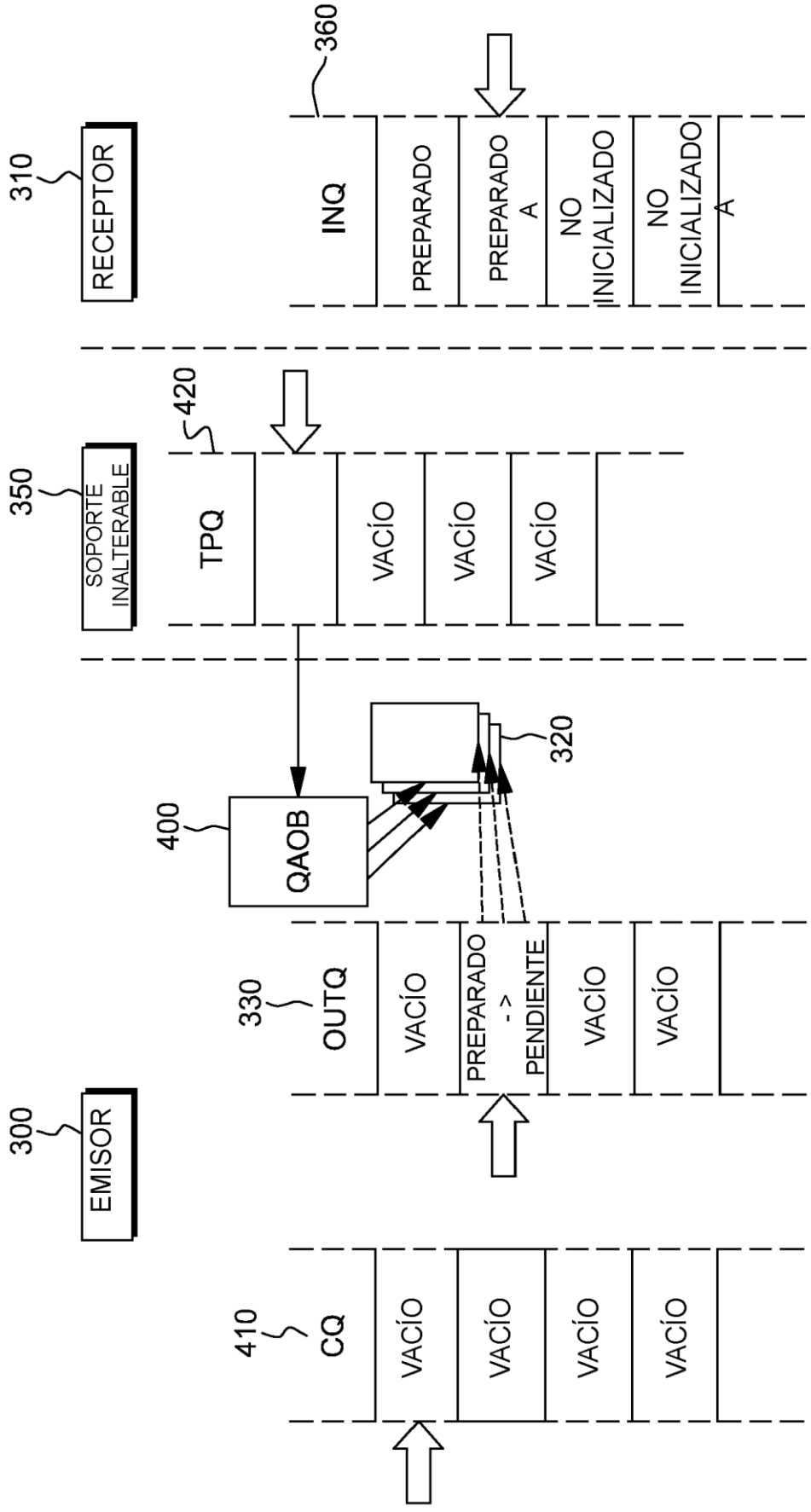
300
EMISOR
310
RECEPTOR
350
SOPORTE INALTERABLE
400
QAOB
410
CQ
VACÍO
VACÍO
VACÍO
VACÍO
330
OUTQ
VACÍO
PREPARADO -> PENDIENTE
VACÍO
VACÍO
320
420
TPQ
VACÍO
VACÍO
VACÍO
360
INQ
PREPARADO
PREPARADO A
NO INICIALIZADO
NO INICIALIZADO A

FIG. 4

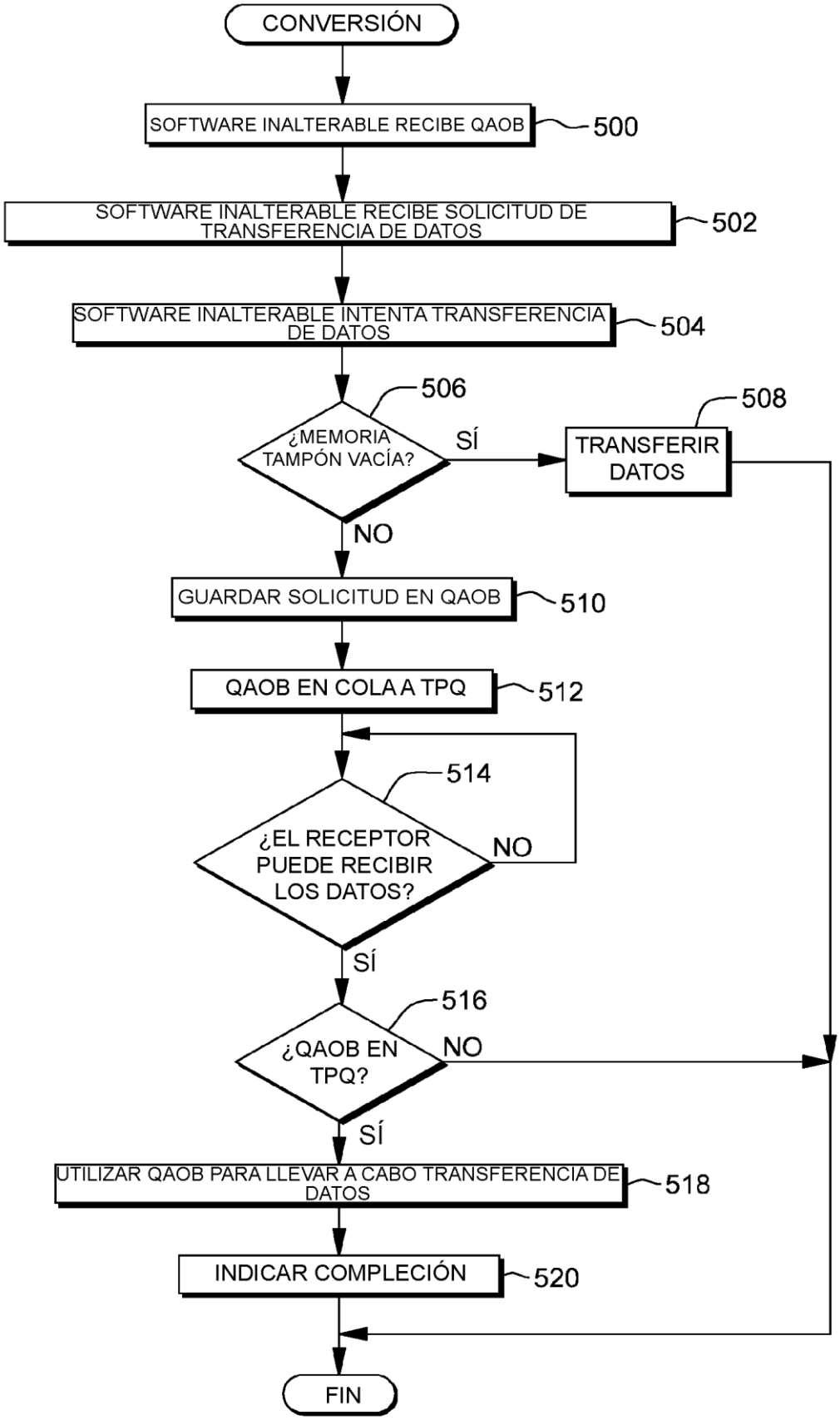
CONVERSIÓN
SOFTWARE INALTERABLE RECIBE QAOB
500
SOFTWARE INALTERABLE RECIBE SOLICITUD DE TRANSFERENCIA DE DATOS
502
SOFTWARE INALTERABLE INTENTA TRANSFERENCIA DE DATOS
504
506
¿MEMORIA TAMPÓN VACÍA?
SÍ
508
TRANSFERIR DATOS
NO
GUARDAR SOLICITUD EN QAOB
510
QAOB EN COLA A TPQ
512
514
¿EL RECEPTOR PUEDE RECIBIR LOS DATOS?
NO
SÍ
516
¿QAOB EN TPQ?
NO
SÍ
UTILIZAR QAOB PARA LLEVAR A CABO TRANSFERENCIA DE DATOS
518
INDICAR COMPLECIÓN
520
FIN

FIG. 5A

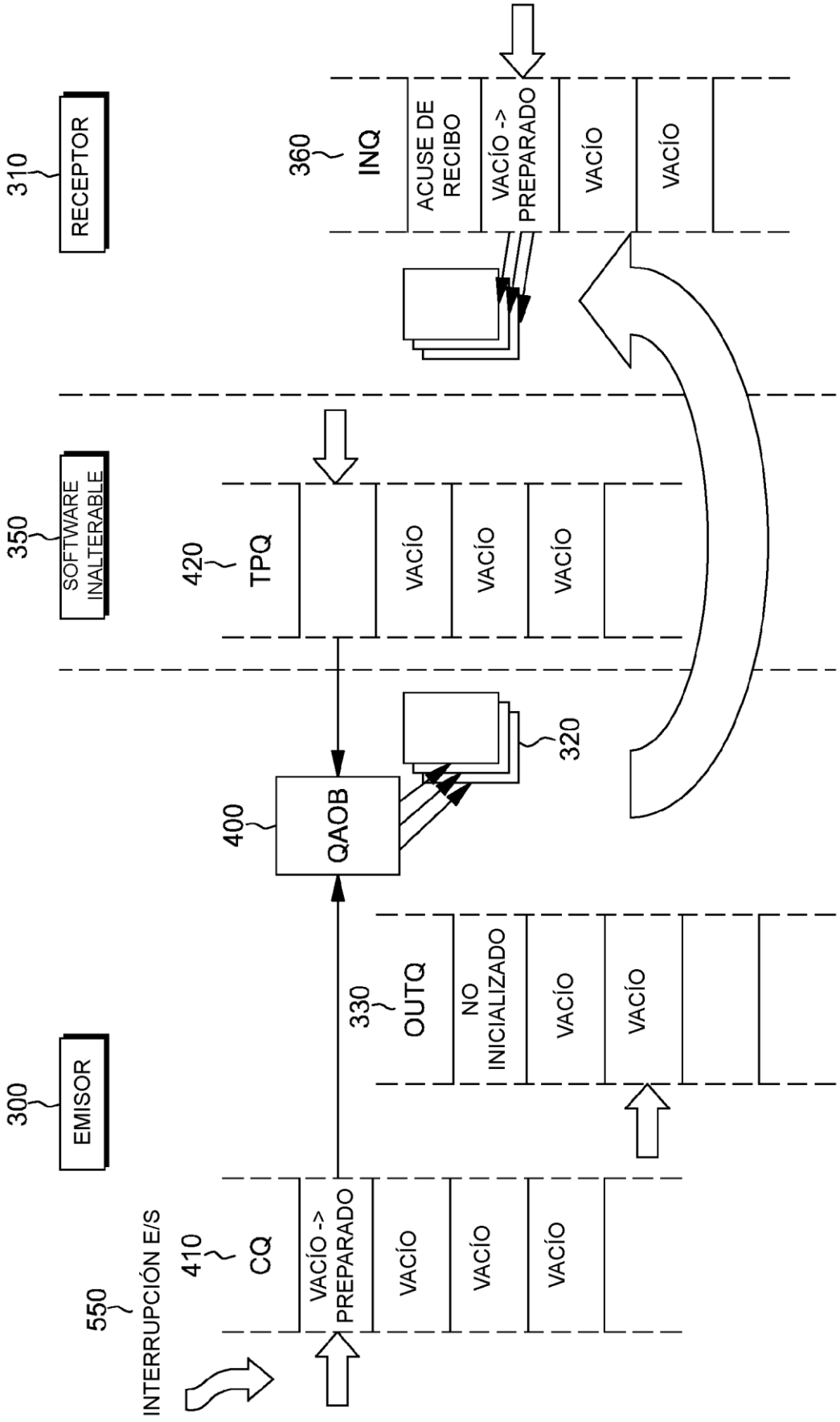
310
RECEPTOR
360
INQ
ACUSE DE RECIBO
VACÍO -> PREPARADO
VACÍO
VACÍO
350
SOFTWARE INALTERABLE
420
TPQ
VACÍO
VACÍO
VACÍO
320
400
QAOB
300
EMISOR
330
OUTQ
NO INICIALIZADO
VACÍO
VACÍO
550
INTERRUPCIÓN E/S
410
CQ
VACÍO -> PREPARADO
VACÍO
VACÍO
VACÍO

FIG. 5B

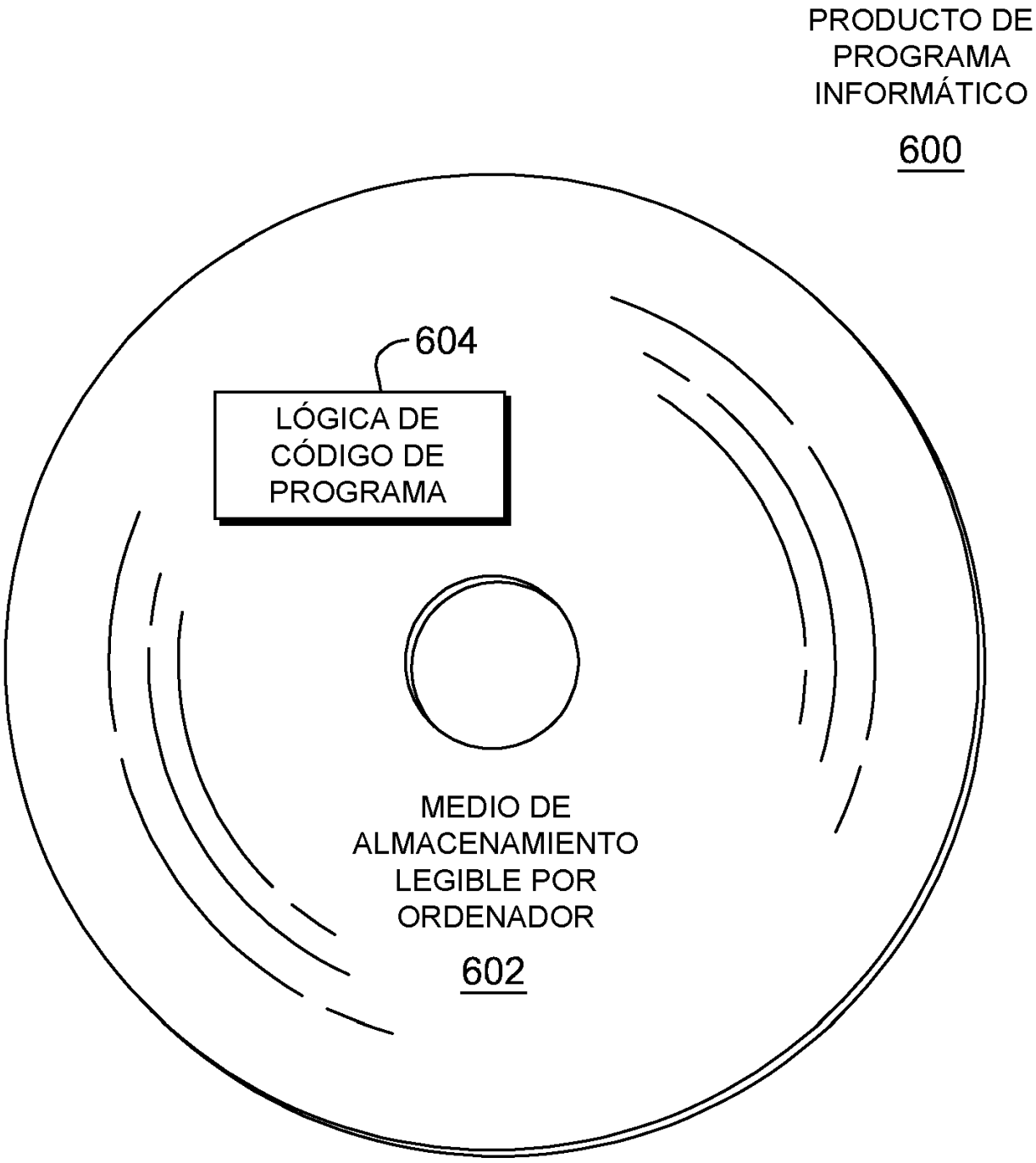
PRODUCTO DE PROGRAMA INFORMÁTICO
600
604
LÓGICA DE CÓDIGO DE PROGRAMA
MEDIO DE ALMACENAMIENTO LEGIBLE POR ORDENADOR
602

FIG. 6

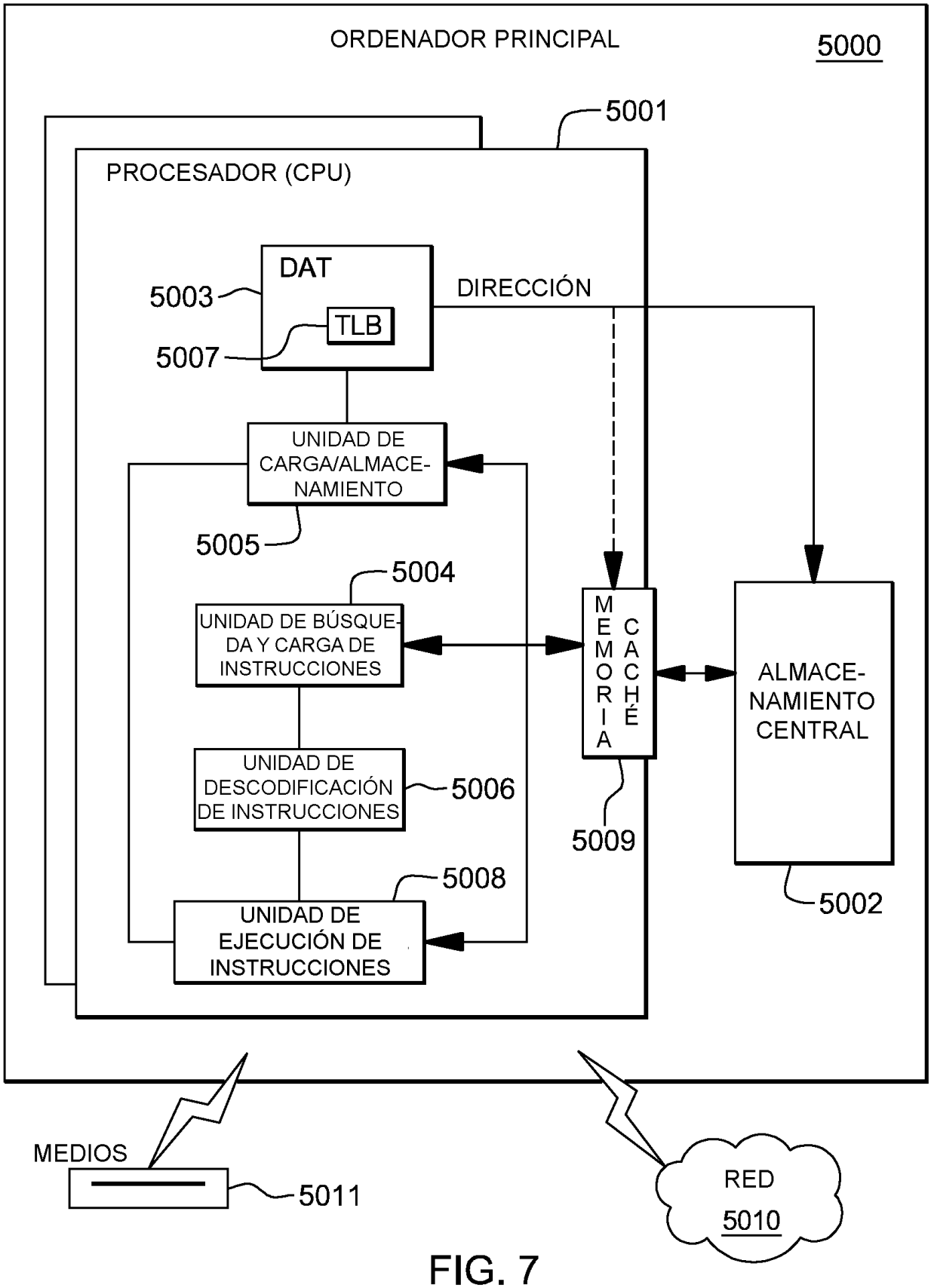
ORDENADOR PRINCIPAL
5000
5001
PROCESADOR (CPU)
DAT
5003
TLB
5007
DIRECCIÓN
UNIDAD DE CARGA/ALMACE-NAMIENTO
5005
5004
UNIDAD DE BÚSQUE-DA Y CARGA DE INSTRUCCIONES
UNIDAD DE DESCODIFICACIÓN DE INSTRUCCIONES
5006
5008
UNIDAD DE EJECUCIÓN DE INSTRUCCIONES
MEMORIA CACHÉ
5009
ALMACE-NAMIENTO CENTRAL
5002
MEDIOS
5011
RED
5010

FIG. 7

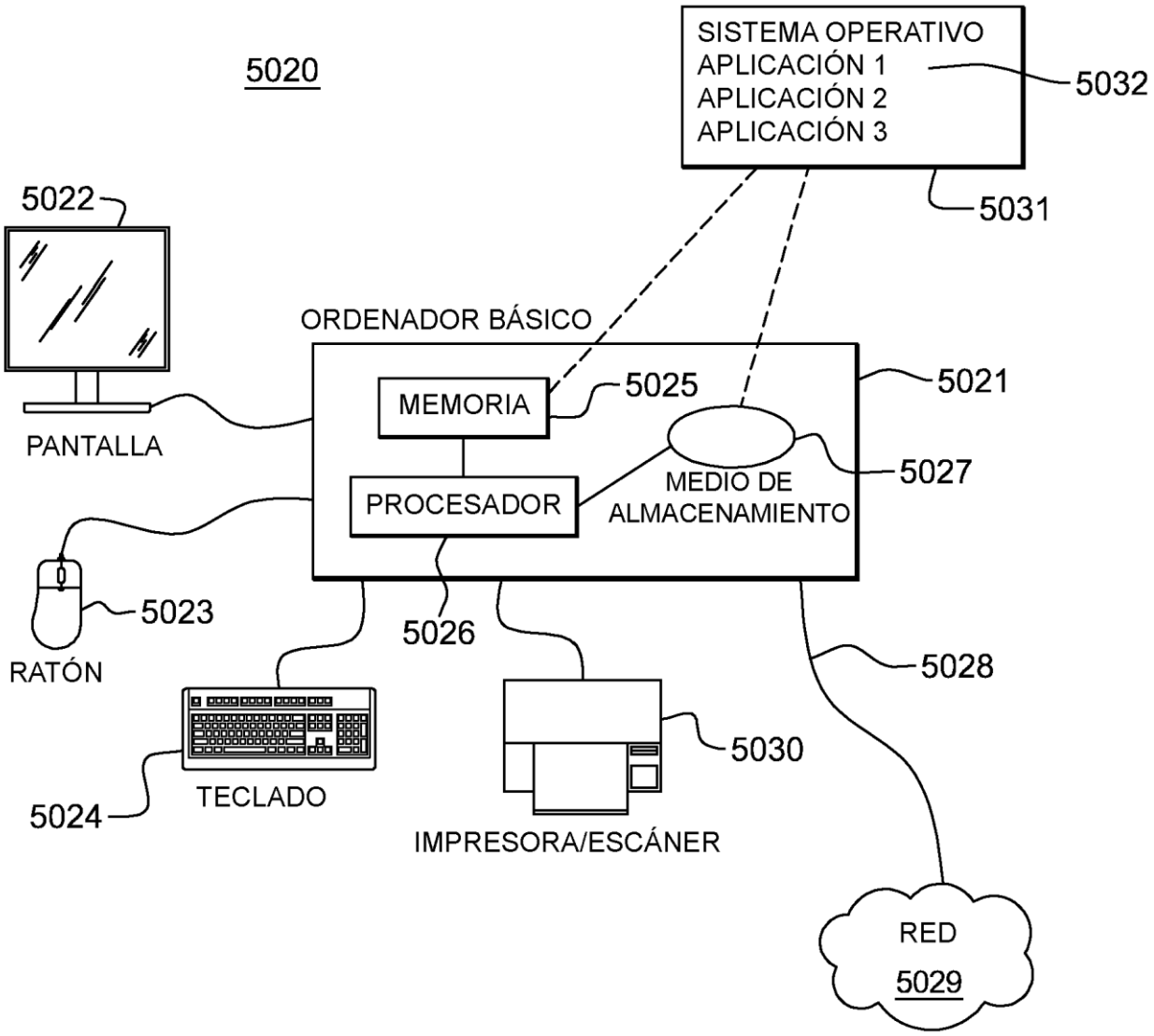
5020
SISTEMA OPERATIVO
APLICACIÓN 1
APLICACIÓN 2
APLICACIÓN 3
5032
5031
5022
ORDENADOR BÁSICO
MEMORIA
5025
5021
PANTALLA
PROCESADOR
MEDIO DE ALMACENAMIENTO
5027
5023
RATÓN
5026
5028
5030
TECLADO
5024
IMPRESORA/ESCÁNER
RED
5029

FIG. 8

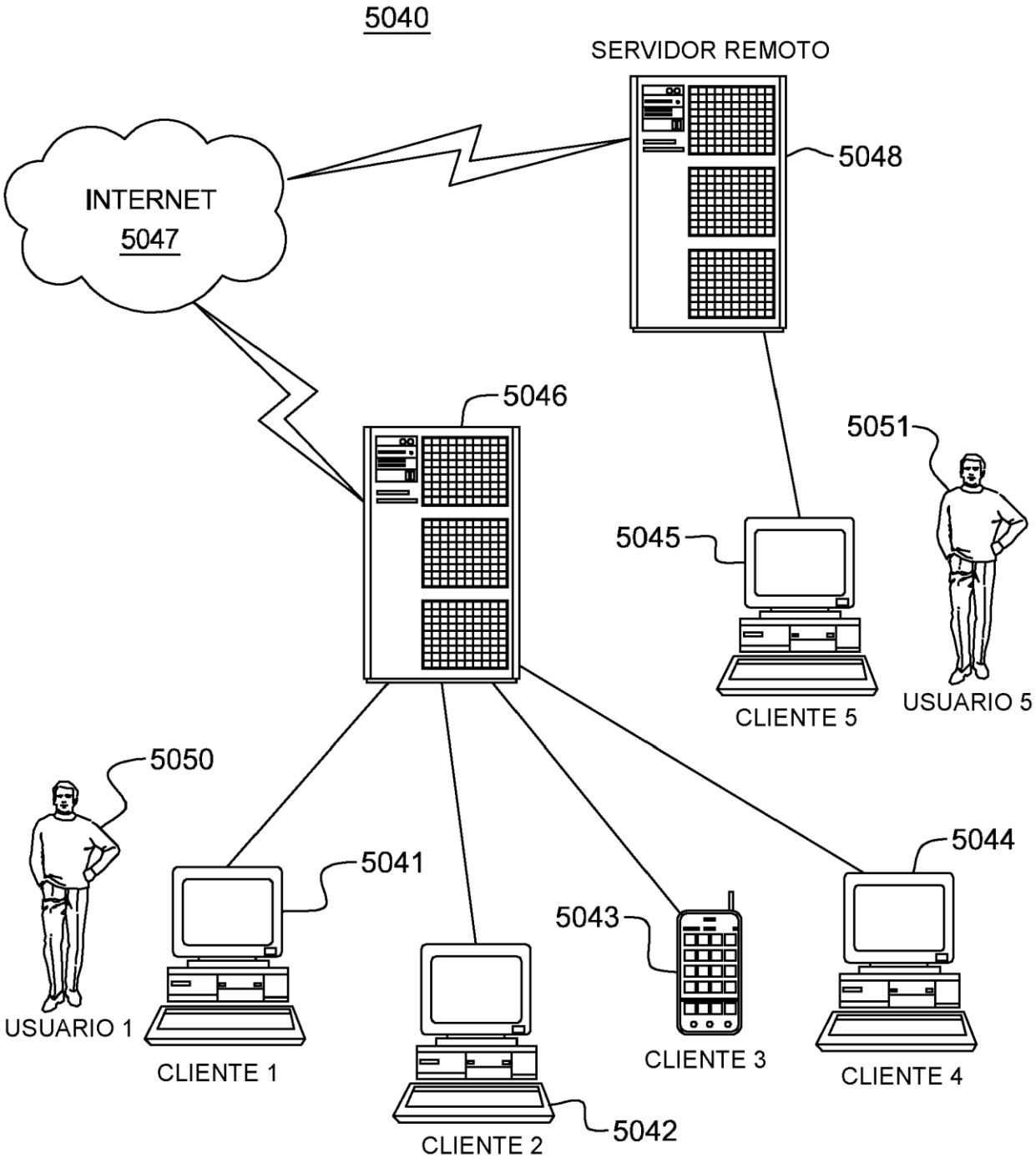
5040
SERVIDOR REMOTO
5048
INTERNET
5047
5046
5051
5045
USUARIO 5
CLIENTE 5
5050
5041
5044
5043
USUARIO 1
CLIENTE 1
CLIENTE 3
CLIENTE 4
CLIENTE 2
5042

FIG. 9

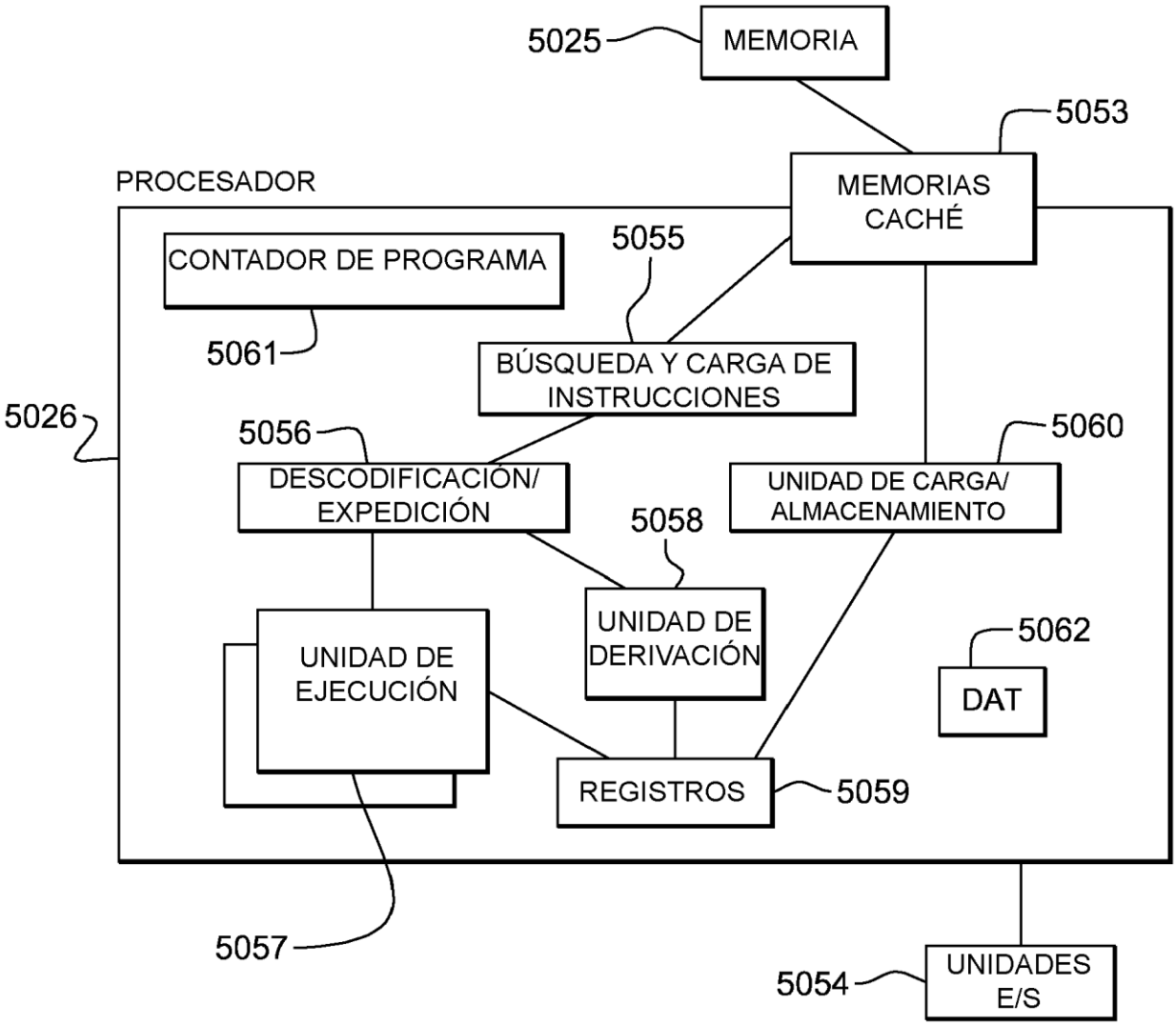
5025
MEMORIA
5053
MEMORIAS CACHÉ
PROCESADOR
CONTADOR DE PROGRAMA
5055
5061
BÚSQUEDA Y CARGA DE INSTRUCCIONES
5026
5056
5060
DESCODIFICACIÓN/ EXPEDICIÓN
UNIDAD DE CARGA/ ALMACENAMIENTO
5058
UNIDAD DE DERIVACIÓN
5062
UNIDAD DE EJECUCIÓN
DAT
REGISTROS
5059
5057
5054
UNIDADES E/S

FIG. 10

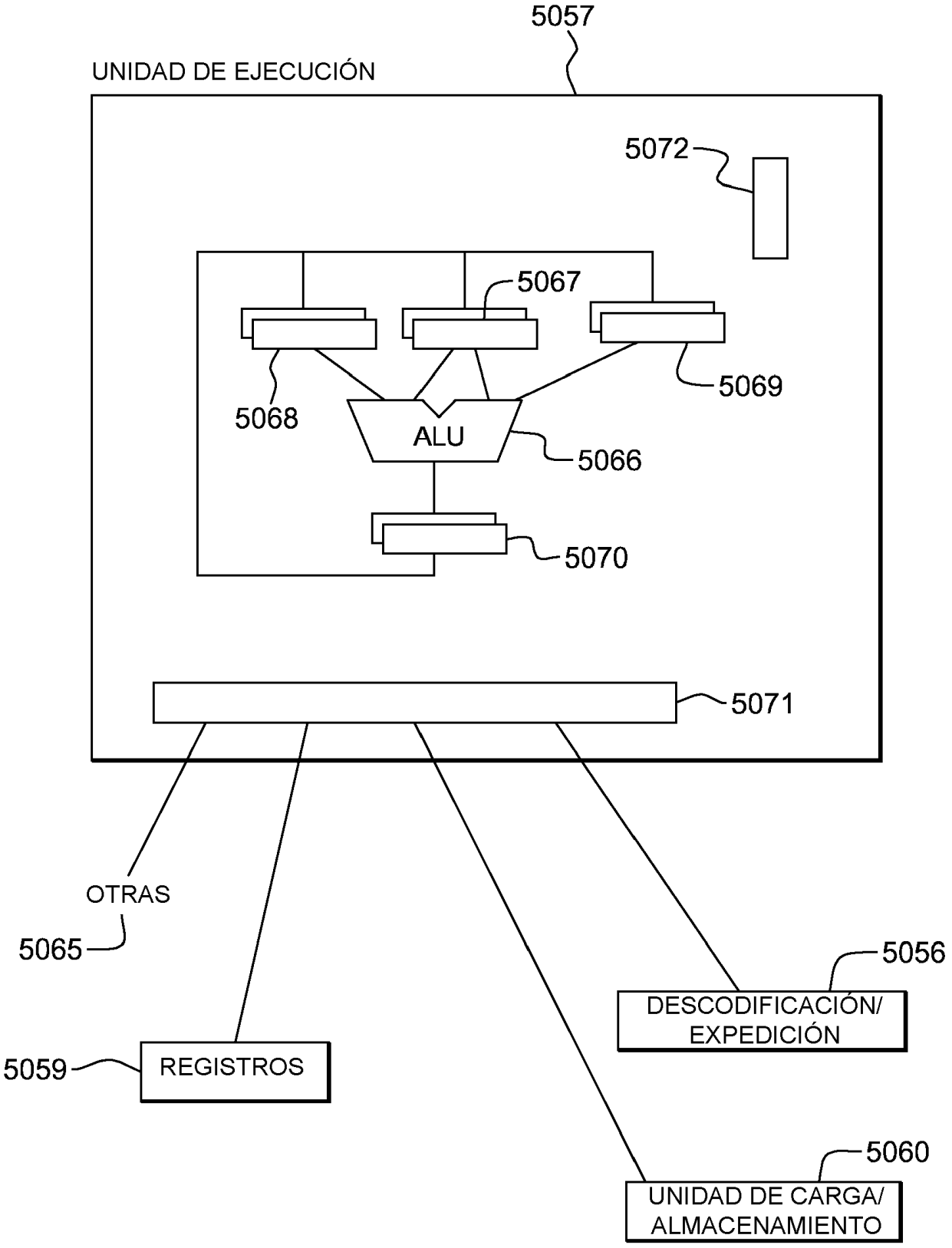
5057
UNIDAD DE EJECUCIÓN
5072
5067
5068
5069
ALU
5066
5070
5071
OTRAS
5065
5056
DESCODIFICACIÓN/
EXPEDICIÓN
5059
REGISTROS
5060
UNIDAD DE CARGA/
ALMACENAMIENTO

FIG. 11A

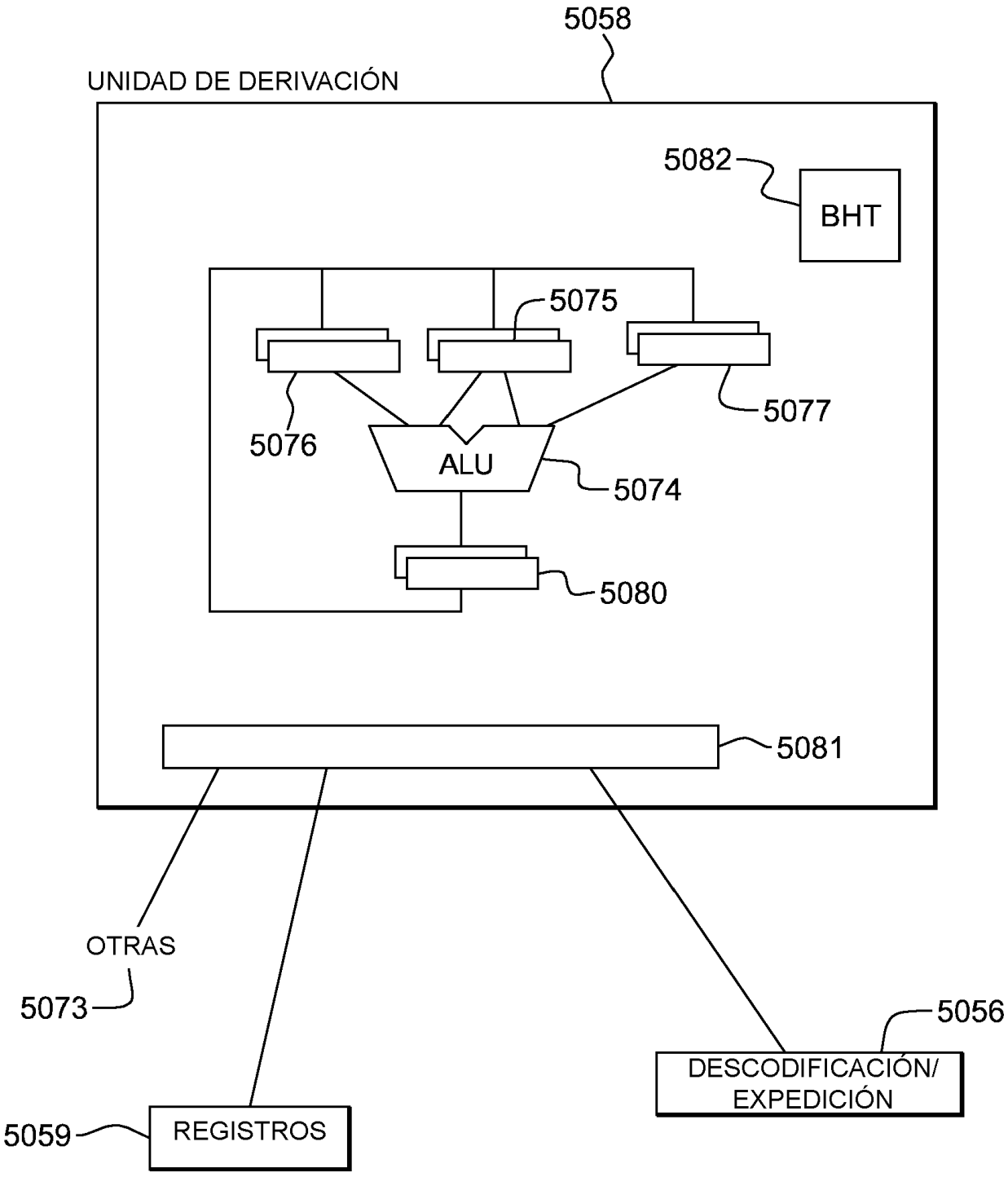
5058
UNIDAD DE DERIVACIÓN
5082
BHT
5075
5076
5077
ALU
5074
5080
5081
OTRAS
5073
5056
DESCODIFICACIÓN/
EXPEDICIÓN
5059
REGISTROS

FIG. 11B

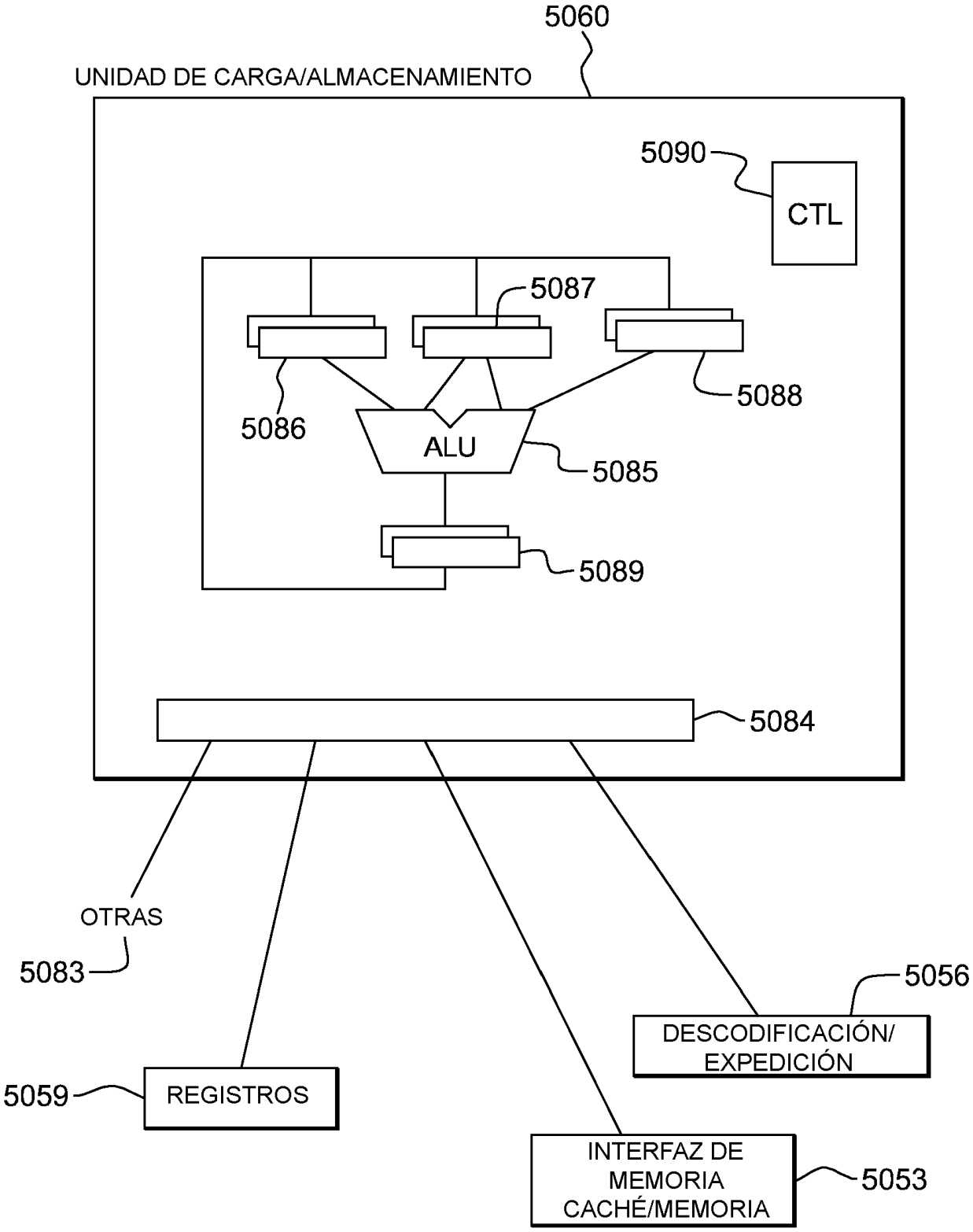
5060
UNIDAD DE CARGA/ALMACENAMIENTO
5090
CTL
5087
5086
5088
ALU
5085
5089
5084
OTRAS
5083
5056
DESCODIFICACIÓN/
EXPEDICIÓN
5059
REGISTROS
INTERFAZ DE
MEMORIA
CACHÉ/MEMORIA
5053

FIG. 11C

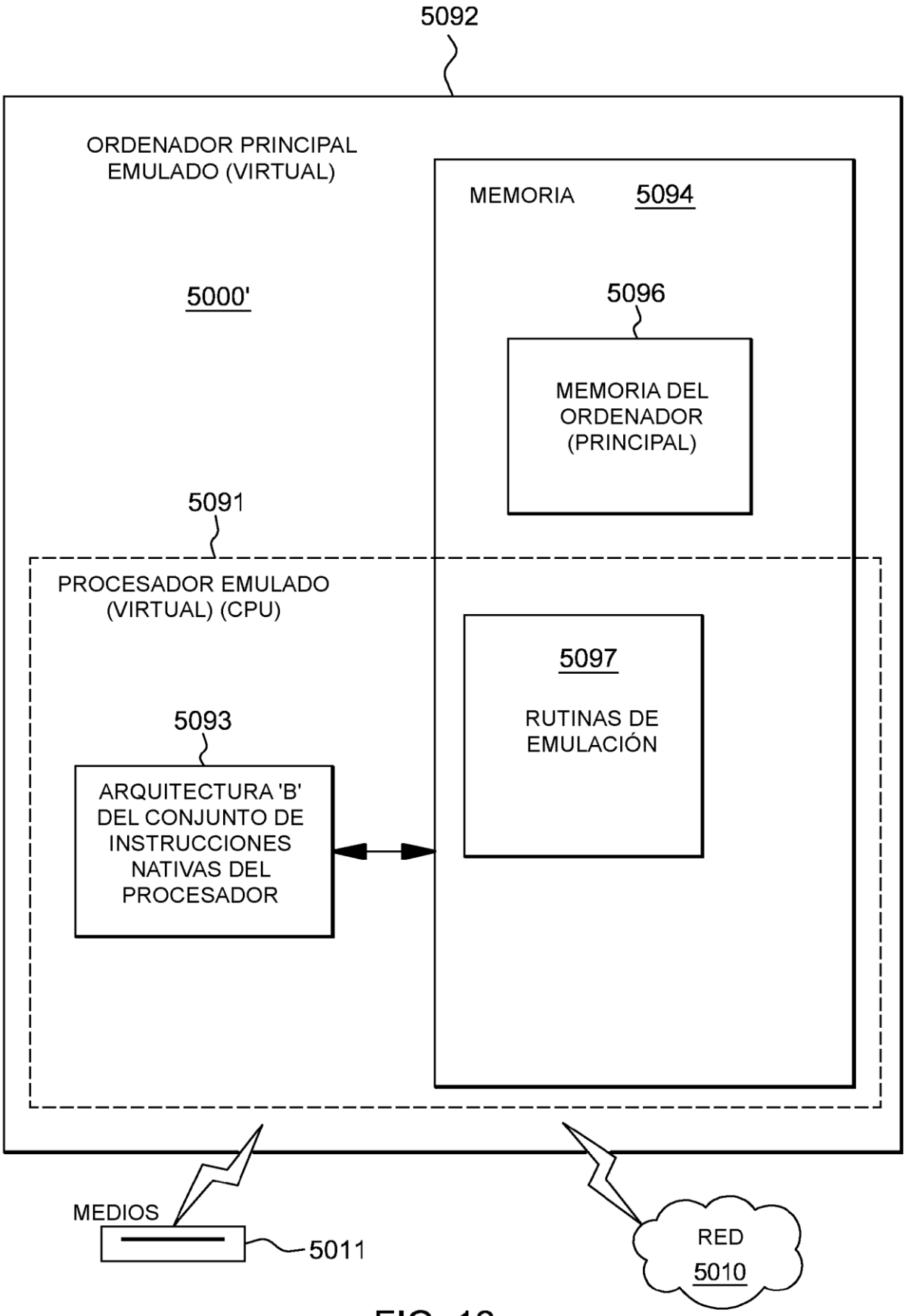
5092
ORDENADOR PRINCIPAL EMULADO (VIRTUAL)
MEMORIA 5094
5000'
5096
MEMORIA DEL ORDENADOR (PRINCIPAL)
5091
PROCESADOR EMULADO (VIRTUAL) (CPU)
5097
RUTINAS DE EMULACIÓN
5093
ARQUITECTURA 'B' DEL CONJUNTO DE INSTRUCCIONES NATIVAS DEL PROCESADOR
MEDIOS
5011
RED
5010

FIG. 12