(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 435 634**

(51) Int. CI.:

G06F 9/455 (2006.01)
G06F 9/30 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **08.11.2010 E 10775820 (3)**

(97) Fecha y número de publicación de la concesión europea: **16.10.2013 EP 2430532**

(54) Título: **Funcionalidad de virtualización de funciones para bloquear una función de instrucción de una instrucción multi-función de un procesador virtual**

(30) Prioridad:

24.06.2010 US 822368

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
20.12.2013

(73) Titular/es:

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

(72) Inventor/es:

GREINER, DAN;
OSISEK, DAMIAN, LEO;
SLEGEL, TIMOTHY y
HELLER, LISA

(74) Agente/Representante:

DE ELZABURU MÁRQUEZ, Alberto

DESCRIPCIÓN

Funcionalidad de virtualización de funciones para bloquear una función de instrucción de una instrucción multi-función de un procesador virtual

Campo de la invención

La presente invención se refiere a sistemas informáticos y, más particularmente, a una funcionalidad de las instrucciones de un procesador de un sistema informático.

Antecedentes de la invención

Marcas comerciales: IBM® es una marca comercial registrada de International Business Machines Corporation, Armonk, Nueva York, EE. UU. S/390, Z900, z990 y z10 y otros nombres de producto pueden ser marcas comerciales registradas o nombres de productos de International Business Machines Corporation o de otras empresas.

Empezando con máquinas conocidas como el sistema IBM® 360 en la década de 1960 hasta el presente, IBM ha creado, gracias al trabajo de muchos ingenieros de gran talento, una arquitectura especial que, debido a su naturaleza esencial a un sistema informático, ha llegado a ser conocida como el "mainframe", cuyos principios de funcionamiento definen la arquitectura de la máquina mediante la descripción de las instrucciones que pueden ser ejecutadas en la implementación "mainframe" de las instrucciones que habían sido inventadas por los inventores de IBM y adoptadas, debido a su importante contribución a mejorar el estado de la máquina de computación representada por el "mainframe", como importantes contribuciones por su inclusión en IBM's Principles of Operation, tal como se ha indicado a lo largo de los años. La octava edición de la IBM® z/Architecture® Principles of Operation, que se publicó en Febrero de 2009, se ha convertido en la referencia estándar publicada como SA22-7832-07 y se incorpora en los servidores mainframe z10® de IBM, incluyendo los servidores z10® con sistema IBM, clase Enterprise.

Con referencia a la Fig. 1A, en la misma se representan los componentes representativos de un sistema 50 informático servidor de la técnica anterior. Pueden emplearse también otras disposiciones de componentes en un sistema informático, que son bien conocidas en la técnica. El servidor 50 representativo comprende una o más CPUs 1 en comunicación con un almacenamiento principal (memoria 2 de ordenador), así como interfaces de E/S a dispositivos 11 de almacenamiento y redes 10 para la comunicación con otros ordenadores o SANs y similares. La CPU 1 es conforme a una arquitectura que tiene un conjunto de instrucciones diseñado y una funcionalidad diseñada. La CPU 1 puede tener una funcionalidad de traducción dinámica de direcciones (Dynamic Address Translation, DAT) 3 para transformar las direcciones de programa (direcciones virtuales) en direcciones de memoria real. Típicamente, una DAT incluye una memoria temporal de traducciones anticipadas (TLB) 7 para almacenar en caché las traducciones, de manera que los accesos futuros al bloque de memoria 2 de ordenador no requieran el retraso debido a la traducción de dirección. Típicamente, se emplea una caché 9 entre una memoria 2 de ordenador y el procesador 1. La caché 9 puede ser jerárquica, y puede tener una gran caché disponible para más de una CPU y cachés más pequeñas y más rápidas (de nivel inferior) entre la caché grande y cada CPU. En algunas implementaciones, las cachés de nivel inferior se dividen para proporcionar cachés de nivel bajo separadas para extracción de instrucciones y accesos a datos. En una realización, una instrucción es extraída de la memoria 2 por una unidad 4 de extracción de instrucciones, a través de una caché 9. La instrucción es decodificada en una unidad (6) de decodificación de instrucciones y es enviada (con otras instrucciones, en algunas realizaciones) a las unidades 8 de ejecución de instrucciones. Típicamente, se emplean varias unidades 8 de ejecución, por ejemplo, una unidad de ejecución aritmética, una unidad de ejecución en coma flotante y una unidad de ejecución de instrucción de bifurcación. La instrucción es ejecutada por la unidad de ejecución, accediendo a los operandos desde registros especificados por las instrucciones o desde la memoria, según sea necesario. Si un operando debe ser accedido (cargado o almacenado) desde la memoria 2, una unidad 5 de carga de almacenamiento gestiona típicamente el acceso, bajo el control de la instrucción que se está ejecutando. Las instrucciones pueden ser ejecutadas en circuitos de hardware o en microcódigo interno (firmware) o por una combinación de ambos.

En la Fig. 1B, se proporciona un ejemplo de un sistema 21 informático servidor emulado de la técnica anterior, que emula un sistema 50 informático servidor de una arquitectura servidor. En el sistema 21 informático servidor emulado, el procesador (CPU) 1 servidor es un procesador servidor emulado (o procesador servidor virtual) y comprende un procesador 27 de emulación que tiene un conjunto de instrucciones nativas diferente al del procesador 1 del servidor 50. El sistema 21 informático servidor emulado tiene una memoria 22 accesible para el procesador 27 de emulación. En la realización ejemplar, la memoria 27 está dividida en una porción de memoria 2 del servidor y una porción de rutinas 23 de emulación. La memoria 2 del servidor está disponible para los programas del servidor 21 emulado, según la arquitectura del servidor. El procesador 27 de emulación ejecuta las instrucciones nativas de un conjunto de instrucciones diseñadas de una arquitectura diferente a la del procesador 1 emulado, siendo obtenidas las instrucciones nativas desde la memoria 23 de rutinas de emulación, y pueden acceder a una instrucción de servidor para su ejecución desde un programa en la memoria 2 de servidor, empleando una o más instrucciones obtenidas en una rutina de secuencia y acceso/decodificación, que puede decodificar la instrucción o las instrucciones de servidor accedidas, para determinar una rutina de ejecución de instrucción nativa para emular la

función de la instrucción de servidor accedida. Otras funcionalidades que se definen para la arquitectura del sistema 50 informático servidor pueden ser emuladas por rutinas de funcionalidades diseñadas, incluyendo funcionalidades tales como registros de propósito general, registros de control, traducción dinámica de direcciones y soporte de subsistema de E/S y caché del procesador, por ejemplo. Las rutinas de emulación pueden aprovechar también una función disponible en el procesador 27 de emulación (tal como registros generales y traducción dinámica de direcciones virtuales) para mejorar el rendimiento de las rutinas de emulación. Pueden proporcionarse también hardware especial y motores de descarga para ayudar al procesador 27 en la emulación de la función del servidor 50.

En un mainframe, las instrucciones máquina diseñadas son usadas por los programadores, normalmente, en la actualidad, programadores en “C”, frecuentemente por medio de una aplicación de compilador. Estas instrucciones, almacenadas en el medio de almacenamiento, pueden ser ejecutadas, de manera nativa, en un servidor IBM de arquitectura z, o, como alternativa, en máquinas que ejecutan otras arquitecturas. Pueden ser emuladas en los servidores mainframe de IBM existentes y futuros y en otras máquinas de IBM (por ejemplo, servidores de la serie p® y servidores de la serie x®). Pueden ser ejecutadas en máquinas que ejecutan Linux en una amplia variedad de máquinas que usan hardware fabricado por IBM®, Intel®, AMD™, Sun Microsystems y otros. Además de la ejecución en ese hardware bajo una Arquitectura z®, se puede usar también Linux, así como máquinas que usan emulación, tal como se describe en http://www.turbohercules.com, http://www.hercules-390.org y http://www.funsoft.com. En modo de emulación, el software de emulación es ejecutado por un procesador nativo para emular la arquitectura de un procesador emulado.

Típicamente, el procesador 27 nativo ejecuta un software 23 de emulación que comprende o bien firmware o bien un sistema operativo nativo, para realizar la emulación del procesador emulado. El software 23 de emulación es responsable de extraer y ejecutar instrucciones de la arquitectura del procesador emulado. El software 23 de emulación mantiene un contador de programa emulado para realizar un seguimiento de los límites de las instrucciones. El software 23 de emulación puede extraer una o más instrucciones máquina emuladas en cada momento y puede convertir la una o más instrucciones máquina emuladas a un grupo correspondiente de instrucciones máquina nativas, para su ejecución por el procesador 27 nativo. Estas instrucciones convertidas pueden ser almacenadas en caché, de manera que puede conseguirse una conversión más rápida. No obstante, el software de emulación debe mantener las normas de arquitectura de la arquitectura del procesador emulado, para garantizar que los sistemas operativos y las aplicaciones escritas para el procesador emulado funcionan correctamente. Además, el software de emulación debe proporcionar recursos identificados por la arquitectura del procesador 1 emulado, incluyendo pero sin limitarse a, registros de control, registros de propósito general, registros en coma 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 que mantienen la hora del día (TOD) e interfaces diseñados para subsistemas de E/S, de manera que un sistema operativo o un programa de aplicación diseñado para ejecutarse en el procesador emulado, pueda ser ejecutado en el procesador nativo que tiene el software de emulación.

Una instrucción específica que está siendo emulada es decodificada, y una subrutina es llamada para realizar la función de la instrucción individual. Una función 23 de software de emulación, que emula una función de un procesador 1 emulado, está implementada, por ejemplo, en un controlador o una subrutina en "C", o algún otro procedimiento para proporcionar un controlador para el hardware específico, tal como estará dentro de las capacidades de las personas con conocimientos en la materia, tras comprender la descripción de la realización preferente. Varias patentes de emulación de software y hardware, incluyendo pero no limitándose a, US 5551013 para un "Multiprocessor for hardware emulation" de Beausoleil et al., y US 6009261: Preprocessing of stored target routines for emulating incompatible instructions on a target processor" de Scalzi et al., y US 5574873: Decoding guest instruction to directly access emulation routines that emulate the guest instructions, de Davidian et al;. US 6308255: Symmetrical multiprocessing bus and chipset used for coprocessor support allowing non-native code to run in a system, de Gorishek et al, y US 6463582: Dynamic optimizing object code translator for architecture emulation and dynamic optimizing object code translation method de Lethin et al; y US 5790825: Method for emulating guest instructions on a host computer through dynamic recompilation of host instructions de Eric Traut. Estas referencias ilustran una diversidad de maneras conocidas para conseguir la emulación de un formato de instrucción diseñado para una máquina diferente para una máquina objetivo disponible para las personas con conocimientos en la materia, así como las técnicas de software comercial usadas por los indicados anteriormente.

En la publicación US Nº 2009/0222813 A1, publicada el 3 de Septiembre de 2009, Astrand, “Selective Exposure to USB Device Functionality for a Virtual Machine”, una máquina virtual (VM) puede ejecutar un sistema operativo (OS) cliente y puede permitir que el OS cliente se conecte a dispositivos USB conectados a un ordenador. La aplicación VM puede filtrar las funciones asociadas con el dispositivo USB de manera que solo algunas de las funciones del dispositivo USB estén expuestas al OS cliente.

La solicitud de patente US US2005/0188171 A1, publicada, describe un cargador de programa dentro de una base de computación confiable que reasigna los códigos de operación (“opcodes”) en el código, usando un mapa de códigos de operación, antes de cargar los códigos de operación en el “pipeline” de ejecución para un procesador. Un código malicioso que intenta ejecutarse directamente en el procesador en lugar de a través del cargador no tendrá los códigos de operación reasignados y, de esta manera, no se ejecutará correctamente.

La solicitud PCT WO006/67115 A1, publicada, describe un microcontrolador en el que un bit de estado es comprobado por un mecanismo habilitador/deshabilitador para determiner si una instrucción está habilitada o no. De manera alternativa, el habilitador compara las instrucciones con una lista predefinida de instrucciones especiales mantenidas por el habilitador.

Advanced Micro Devices: "Live Migration with AMD-V Extended Migration Technology", 30 de Abril de 2008 (2008-04-30), páginas 1-17, describe el control por parte de un supervisor de máquina virtual (Virtual Machine Monitor, VMM) de la información devuelta al software cliente como resultado de una instrucción CPUID. Dicho un VMM, que usa virtualización asistida por hardware, puede usar la function interceptación de CPUID para devolver los bits apropiados siempre que el software cliente ejecute la instrucción CPUID. Esto se implementa usando registros específicos de modelo (Model Specific Registers, MSR) que proporcionan una function de sobreescritura de CPUID para especificar un subconjunto de la información que devuelve la instrucción CPUID. Describe una necesidad de un procedimiento para deshabilitar nuevas instrucciones hasta que todos los procesadores en un conjunto de recursos de procesadores implementen las nuevas instrucciones. Este documento especula con que AMD puede ser compatible con bits de deshabilitación de características para deshabilitar ciertas instrucciones o conjuntos de instrucciones.

Sumario de la invención

En una realización, la ejecución de instrucciones específicas es bloqueada por un procesador lógico que ejecuta una máquina virtual. Un valor de bloqueo de instrucción es establecido para la máquina virtual. Una instrucción es extraída por un procesador lógico para ser ejecutada por el procesador, en la que la instrucción comprende un código de operación, y especifica un código de función de entre una pluralidad de códigos de función, en el que el código de función está configurado para indicar una función a ser realizada por la instrucción, en la que la instrucción es compatible con uno o más procesadores. Se determina un valor de bloqueo de instrucción de la máquina virtual. En respuesta al valor de bloqueo de instrucción que permite la ejecución de la instrucción que tiene el código de operación de la función, la instrucción extraída es ejecutada por el procesador. En respuesta al valor de bloqueo de instrucción que no permite la ejecución de la instrucción, la ejecución de la instrucción extraída es bloqueada y se genera un evento de excepción de programa (excepción de programa, por ejemplo).

En una realización, el valor de bloqueo de instrucción es definido por la máquina virtual para bloquear la ejecución de la instrucción, realizándose el establecimiento del valor de bloqueo de instrucción en respuesta a la habilitación de la máquina virtual a usar el procesador físico; se establece otro valor de bloqueo de instrucción, estando definido el otro valor de bloqueo de instrucción para otra máquina virtual que tiene otro procesador lógico, realizándose el establecimiento del otro valor de bloqueo de instrucción en respuesta a la habilitación de la otra máquina virtual para usar el procesador físico; y en respuesta al otro valor de bloqueo de instrucción que permite la ejecución de la instrucción, se permite la ejecución de la instrucción por el otro procesador lógico; y en respuesta al otro valor de bloqueo de instrucción que no permite la ejecución de la instrucción, no se permite la ejecución de la instrucción por el otro procesador lógico.

En una realización, la instrucción es la instrucción permitida si la instrucción emplea un código de operación permitido, en la que la instrucción es la instrucción no permitida si la instrucción emplea un código de operación no permitido.

En una realización, se realiza una determinación de si la instrucción es o no la instrucción permitida, asociando el código de operación de la instrucción con el valor de bloqueo de instrucción.

En una realización, la instrucción extraída especifica una función a llevar a cabo, el código de operación de la instrucción se usa como índice en una tabla para buscar el valor de bloqueo de instrucción, en el que el valor de bloqueo de instrucción comprende una campo de permiso, en el que el campo de permiso se usa para determinar las funciones permitidas. Si la función es una función permitida, se permite la ejecución de la instrucción, y si la función no es una función permitida, no se permite la ejecución de la instrucción.

Lo indicado anteriormente, así como objetivos, características y ventajas adicionales, serán evidentes a partir de la descripción siguiente.

Breve descripción de los dibujos

Ahora, se describirán, solamente a modo de ejemplo, las realizaciones de la invención, con referencia a los dibujos adjuntos, en los que:

La Fig. 1A es un diagrama que representa un sistema informático servidor ejemplar;

La Fig. 1B es un diagrama que representa un sistema informático servidor de emulación ejemplar;

La Fig. 1C es un diagrama que representa un sistema informático ejemplar;

La Fig. 2 es un diagrama que representa una red de ordenadores ejemplar;

La Fig. 3 es un diagrama que representa unos elementos ejemplares de un sistema informático;

La Fig. 4A es un diagrama que representa una unidad de ejecución ejemplar;

La Fig. 4B es un diagrama que representa una unidad de bifurcación ejemplar;

La Fig. 4C es un diagrama que representa una unidad de carga/almacenamiento ejemplar.

La Fig. 5 es un diagrama que representa una partición lógica ejemplar;

La Fig. 6 es un diagrama que representa los elementos de una partición lógica ejemplar;

La Fig. 7 es un diagrama que representa los elementos de una partición lógica ejemplar;

La Fig. 8 es un diagrama de flujo que representa una tabla de códigos de operación ejemplar;

La Fig. 9 es un diagrama de flujo que representa una técnica de bloqueo ejemplar;

La Fig. 10 es un diagrama de flujo que representa una técnica de bloqueo ejemplar;

La Fig. 11 es un diagrama de flujo que representa una técnica de bloqueo ejemplar; y

Las Figs. 12-15 representan diagramas de flujo de técnicas de bloqueo de instrucciones;

Descripción detallada

Las realizaciones pueden ser llevadas a la práctica mediante software (a veces denominado código interno bajo licencia, firmware, micro-código, mili-código, pico-código y similares, cualquiera de los cuales sería coherente con las enseñanzas de la presente memoria). Con referencia a la Fig. 1A, una realización de código de programa de software es accedida, típicamente, por el procesador, conocido también como CPU (Unidad Central de Procesamiento) 1 del sistema 50 desde un medio 7 de almacenamiento permanente, tal como una unidad de CD-ROM, una unidad de cinta o un disco duro. El código de programa de software puede materializarse en cualquiera de entre una diversidad de medios conocidos para su uso con un sistema de procesamiento de datos, tal como un disquete, un disco duro o un CD-ROM. El código puede ser distribuido en dichos medios, o puede ser distribuido a los usuarios desde la memoria 2 del ordenador o el almacenamiento de un sistema informático sobre una red 10 a otros sistemas informáticos para su uso por los usuarios de dichos otros sistemas.

De manera alternativa, el código de programa puede materializarse en la memoria 2, y puede ser accedido por el procesador 1 usando el bus del procesador. Dicho código de programa incluye un sistema operativo que controla la función y la interacción de los diversos componentes de ordenador y uno o más programas de aplicación. El código de programa es paginado normalmente desde un medio 11 de almacenamiento masivo a una memoria 2 de alta velocidad, donde se encuentra disponible para ser procesado por el procesador 1. Las técnicas y los procedimientos para materializar el código de programa de software en una memoria, en un medio físico y/o distribuir código de software a través de redes son bien conocidos y no se describirán adicionalmente en la presente memoria. El código de programa, cuando es creado y almacenado en un medio tangible (incluyendo, pero sin limitarse a, módulos de memoria electrónica (RAM), memoria flash, discos compactos (CDs), DVDs, cintas magnéticas y similares) es denominado, frecuentemente, un “producto de programa informático". Típicamente, el medio del producto de programa informático puede ser leído por un circuito de procesamiento, preferentemente en un sistema informático para su ejecución por el circuito de procesamiento.

La Fig. 1C ilustra un sistema de hardware de servidor o estación de trabajo representativo. El sistema 100 de la Fig. 1C comprende un sistema 101 informático representativo, tal como un ordenador personal, una estación de trabajo o un servidor, que incluye dispositivos periféricos opcionales. La estación de trabajo 101 incluye uno o más procesadores 106 y un bus empleado para conectar y permitir la comunicación entre el procesador o los procesadores 106 y los otros componentes del sistema 101, según técnicas conocidas. El bus conecta el procesador 106 a la memoria 105 y al almacenamiento 107 permanente, que puede incluir una unidad de disco duro (incluyendo cualquiera de entre un medio magnético, CD, DVD y memoria Flash, por ejemplo) o una unidad de cinta, por ejemplo. El sistema 101 podría incluir también un adaptador de interfaz de usuario, que conecta el microprocesador 106 a través del bus a uno o más dispositivos de interfaz, tales como un teclado 104, un ratón 103, una impresora/escáner 110 y/u otros dispositivos de interfaz, que pueden ser cualquier dispositivo de interfaz de usuario, tales como una pantalla táctil, una tableta de entrada digital, etc. El bus conecta también un dispositivo 102 de visualización, tal como un monitor o una pantalla LCD, al microprocesador 106 a través de un adaptador de pantalla.

El sistema 101 puede comunicarse con otros ordenadores o redes de ordenadores por medio de un adaptador de red capaz de comunicarse 108 con una red 109. Los adaptadores de red ejemplares son canales de comunicación, token ring, ethernet o módems. Como alternativa, la estación de trabajo 101 puede comunicarse usando una interfaz inalámbrica, tal como una tarjeta CDPD (Cellular Digital Packet Data). La estación de trabajo 101 puede estar asociada con dichos otros ordenadores en una red de área local (LAN) o en una red de área extensa (WAN), o la estación de trabajo 101 puede ser un cliente en una disposición cliente/servidor con otro ordenador, etc. Todas estas

configuraciones, así como el hardware y el software de comunicaciones apropiados, son conocidas en la técnica.

La Fig. 2 ilustra una red 200 de procesamiento de datos en la que pueden llevarse a la práctica las realizaciones. La red 200 de procesamiento de datos puede incluir una pluralidad de redes individuales, tales como una red inalámbrica y una red por cable, cada una de las cuales puede incluir una pluralidad de estaciones de trabajo 101, 201, 202, 203, 204 individuales. Además, tal como apreciarán las personas con conocimientos en la materia, pueden incluirse una o más LANs, donde una LAN puede comprender una pluralidad de estaciones de trabajo inteligentes, acopladas a un procesador servidor.

Todavía con referencia a la Fig. 2, las redes pueden incluir también ordenadores o servidores mainframe, tales como un ordenador pasarela (cliente servidor 206) o servidor de aplicaciones (servidor remoto 208 que puede acceder a un repositorio de datos y puede ser accedido también directamente desde una estación de trabajo 205). Un ordenador 206 pasarela sirve como punto de entrada a cada red 207. Una pasarela es necesaria cuando se conecta un protocolo de red a otro. La pasarela 206 puede estar acoplada, preferentemente, a otra red (por ejemplo, Internet 207) por medio de un enlace de comunicaciones. La pasarela 206 puede estar acoplada también directamente a una o más estaciones de trabajo 101, 201, 202, 203, 204, usando un enlace de comunicaciones. El ordenador pasarela puede ser implementado utilizando un IBM eServer™ zSeries® z9® Server, disponible en IBM Corp.

Típicamente, el código de programación de software es accedido por el procesador 106 del sistema 101 desde un medio 107 de almacenamiento permanente, tal como una unidad de CD-ROM o un disco duro. El código de programación de software puede estar materializado en cualquiera de entre una diversidad de medios conocidos para su uso con un sistema de procesamiento de datos, tal como un disquete, un disco duro o un CD-ROM. El código puede ser distribuido sobre dicho medio, o puede ser distribuido a los usuarios 210, 211 desde la memoria o el almacenamiento de un sistema informático sobre una red a otros sistemas informáticos para su uso por los usuarios de dichos otros sistemas.

Como alternativa, el código 111 de programación puede estar materializado en la memoria 105, y es accedido por el procesador 106 usando el bus del procesador. Dicho código de programación incluye un sistema operativo que controla la función y la interacción de los diversos componentes de ordenador y uno o más programas 112 de aplicación. El código de programa es paginado normalmente desde un medio 107 de almacenamiento masivo a una memoria 105 de alta velocidad, donde se encuentra disponible para ser procesado por el procesador 106. Las técnicas y los procedimientos para materializar el código de programación de software en memoria, en un medio físico y/o distribuir el código de software a través de redes son bien conocidos y no se describirán adicionalmente en la presente memoria. El código de programa, cuando es creado y almacenado en un medio tangible (incluyendo, pero sin limitarse a, módulos de memoria electrónica (RAM), memoria flash, discos compactos (CDs), DVDs, cintas magnéticas y similares) se denomina, frecuentemente, un “producto de programa informático". Típicamente, el medio del producto de programa informático puede ser leído por un circuito de procesamiento, preferentemente en un sistema informático, para su ejecución por el circuito de procesamiento.

La caché que está disponible, de manera más rápida, para el procesador (normalmente, más rápida y más pequeña que otras cachés del procesador) es la caché más baja (nivel 1 o L1) y el almacenamiento principal (memoria principal) es la caché de nivel más alto (L3, si hay 3 niveles). La caché de nivel más bajo se divide, frecuentemente, en una caché de instrucciones (I-Caché), que contiene las instrucciones máquina a ser ejecutadas, y una caché de datos (D-Caché), que contiene los operandos de datos.

Con referencia a la Fig. 3, en la misma se representa una realización de un procesador ejemplar para el procesador 106. Típicamente, se emplean uno o más niveles de caché 303 para almacenar en memoria temporal bloques de memoria para mejorar el rendimiento del procesador. La caché 303 es una memoria temporal de alta velocidad, que mantiene líneas de caché de datos de memoria que tienen una alta probabilidad de ser usadas. Las líneas de caché típicas son 64, 128 o 256 bytes de datos de memoria. Las cachés separadas se emplean más frecuentemente para el almacenamiento en caché de instrucciones que para el almacenamiento en caché de datos. La coherencia de caché (sincronización de las copias de líneas en memoria y las cachés) es proporcionada frecuentemente por diversos algoritmos "Snoop", bien conocidos en la técnica. El almacenamiento 105 principal de un sistema de procesamiento de denomina, frecuentemente, caché. En un sistema de procesamiento que tiene 4 niveles de caché 303, el almacenamiento 105 principal se denomina, a veces, caché de nivel 5 (L5), ya que, típicamente, es más rápida y sólo mantiene una porción del almacenamiento no volátil (DASD, cinta, etc.) que está disponible para un sistema informático. El almacenamiento 105 principal almacena en caché páginas de datos paginados desde y hacia el almacenamiento 105 principal por el sistema operativo.

Un contador de programa (contador de instrucciones) 311 realiza un seguimiento de la dirección de la instrucción actual a ejecutar. Un contador de programa en un procesador de arquitectura z es de 64 bits y puede ser truncado a 31 ó 24 bits, para soportar los límites de direccionamiento anteriores. Típicamente, un contador de programa se materializa en una PSW (palabra de estado de programa) de un ordenador, de manera que persiste durante un cambio de contexto. De esta manera, un programa en ejecución, que tiene un valor de contador de programa, puede ser interrumpido, por ejemplo, por el sistema operativo (cambio de contexto desde el entorno del 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 es usado mientras el sistema

operativo se está ejecutando. Típicamente, el contador de programa es incrementado en una cantidad igual al número de bytes de la instrucción actual. Las instrucciones RISC (Reduced Instruction Set Computing) son, típicamente, de longitud fija, mientras que las instrucciones CISC (Complex Instruction Set Computing) son, típicamente, de longitud variable. Las instrucciones de la arquitectura z de IBM son instrucciones CISC, que tienen una longitud de 2, 4 ó 6 bytes. El contador 311 de programa es modificado o bien por una operación de cambio de contexto o bien por una operación de bifurcación tomada de una instrucción de bifurcación, por ejemplo. En una operación de cambio de contexto, el valor actual del contador de programa es salvado en una palabra de estado de programa (PSW), junto con otra información de estado acerca del programa en ejecución (tales como códigos de condición), y se carga un nuevo valor de contador de programa, que apunta a una instrucción de un nuevo módulo de programa a ejecutar. Se realiza una operación de bifurcación tomada para permitir que el programa tome decisiones o realice bucles dentro del programa, cargando el resultado de la instrucción de bifurcación en el contador 311 de programa.

Típicamente, se emplea una unidad 305 de extracción de instrucciones para extraer instrucciones en nombre del procesador 106. La unidad de extracción extrae “las siguientes instrucciones secuenciales”, las instrucciones objetivo de las instrucciones de bifurcación o las primeras instrucciones de un programa después de un cambio de contexto. Frecuentemente, las unidades de extracción de instrucciones modernas emplean técnicas de pre-extracción para extraer instrucciones con antelación, de manera especulativa, en base a la probabilidad de que las instrucciones pre-extraídas puedan ser usadas. Por ejemplo, una unidad de extracción puede extraer 16 bytes de instrucción que incluyen la siguiente instrucción secuencial y bytes adicionales de instrucciones secuenciales adicionales.

A continuación, las instrucciones extraídas son ejecutadas por el procesador 106. En una realización, la instrucción o las instrucciones extraídas son pasadas a una unidad 306 de envío de la unidad de extracción. La unidad de envío decodifica la instrucción o las instrucciones y reenvía información acerca de la instrucción o las instrucciones decodificadas a las unidades 307, 308, 310 apropiadas. Una unidad 307 de ejecución recibirá, típicamente, información acerca de las instrucciones aritméticas decodificadas desde la unidad 305 de extracción de instrucciones y realizará operaciones aritméticas sobre los operandos, según el código de operación de la instrucción. Preferentemente, los operandos son proporcionados a la unidad 307 de ejecución desde una memoria 105, registros 309 diseñados o desde un campo inmediato de la instrucción en ejecución. Los resultados de la ejecución, cuando se almacenan, se almacenan en una memoria 105, registros 309 o en otro hardware de máquina (tales como registros de control, registros de PSW y similares).

Con referencia a la Fig. 5, en la misma se muestra un entorno de máquina virtual (VM) ejemplar. Un programa hipervisor (que puede ser, en sí mismo, un sistema operativo (OS), tal como zVM de IBM) puede estar ejecutándose en un “hardware” multi-procesador que comprende una pluralidad de procesadores físicos, una memoria física principal y adaptadores físicos para la comunicación con dispositivos periféricos de E/S que incluyen almacenamiento, redes, pantallas y similares. El hipervisor crea imágenes de VM (VM1, VM2 y VM3 por ejemplo) de manera que el software, incluyendo un sistema operativo y programas de aplicación, puede ser ejecutado dentro de la máquina virtual, utilizando recursos virtuales. El software que se ejecuta en una máquina virtual no tiene conocimiento de que está siendo ejecutado en una máquina virtual, y funciona usando los recursos virtuales como si fueran recursos físicos. El sistema operativo zVM de IBM puede crear imágenes "Cliente", cada imagen cliente es efectivamente una máquina virtual. Además, cada uno de los clientes zVM puede ejecutar un sistema operativo zVM creando "clientes de segundo nivel". De esta manera, una máquina virtual (imagen cliente) podría ser anidada en una jerarquía de máquinas virtuales, donde cada zVM desempeña un papel hipervisor para sus imágenes cliente. Por otro lado, una plataforma multi-procesador puede ser "particionada físicamente", a cada partición física pueden asignarse recursos (procesadores, memoria, E/S). Cada partición física es una máquina virtual ya que el software que se ejecuta en la partición no es consciente de los recursos de la máquina no asignados a la partición. De esta manera, los recursos de la máquina están "virtualizados". En otra realización, las particiones lógicas son máquinas virtuales.

Los términos clientes, máquinas virtuales (VMs) y particiones lógicas pueden usarse indistintamente en la presente memoria, ya que existen muchos procedimientos conocidos en la técnica para la virtualización de una imagen de sistema informático.

La virtualización se describe, por ejemplo, en un documento técnico de VMware® titulado "Virtualization Overview" y "VMware VMotion and CPU Compatibility" VMware® VMware® Infrastructure 3. Además la publicación de solicitud de patente Nº 2009/0070760 "VIRTUAL MACHINE (VM) MIGRATION BETWEEN PROCESSOR ARCHITECTURES" de Khatri et al., presentada el 6 de Septiembre de 2007, describe la emulación de cierto conjunto de características para permitir una migración de VM entre conjuntos similares de máquinas mediante el enmascaramiento de bits seleccionados de un registro CPUID.

Con referencia a la Fig. 6, cada máquina virtual puede tener un sistema operativo diferente y aplicaciones diferentes. Por ejemplo, el sistema operativo 1 puede ser z/OS de IBM y el sistema operativo 2 puede ser zLinux de IBM, o todos los sistemas operativos pueden ser el mismo sistema operativo, tal como z/OS.

El hipervisor crea características lógicas, recursos y capacidades para cada máquina virtual en base a las

características, recursos y capacidades físicos. En un sistema ejemplar, las porciones de memoria física pueden ser asignadas a cada máquina virtual por medio de una traducción dinámica de direcciones, los procesadores físicos pueden ser compartidos en el tiempo entre las máquinas virtuales, al igual que la capacidad de E/S.

Con referencia a la Fig. 7, cada procesador lógico puede tener acceso a los registros de características físicas por medio de una máscara de características lógicas gestionada por el hipervisor. De esta manera, el software que se ejecuta en los procesadores lógicos puede parecer que funciona en un nivel de arquitectura de procesador común, incluso si los procesadores reales se encuentran en diferentes niveles de arquitectura. En un ejemplo, el registro de característica física puede ser un registro CPUID de Intel, que indica el nivel de la arquitectura del procesador Intel, así como características específicas que están disponibles para el programador. La máscara de característica lógica está programada para proporcionar la totalidad o un subconjunto de los CPUID de los procesadores físicos al software en una máquina virtual (VM) cuando la máquina virtual realiza una consulta al CPUID del procesador lógico correspondiente.

La arquitectura de procesador x86 de Intel®, "Intel® Itanium® Architecture Software Developer's Manual, Volumen 2, Versión 2.2 Enero de 2006" proporciona Registros CPUID para identificar las características soportadas por un procesador. Los registros CPUID no tienen privilegios y se acceden usando la instrucción mov (desde) indirecta. Todos los registros más allá del número de registro CPUID están reservados y generan un aviso Registro/Campo Reservado si se accede a los mismos. No se permiten las escrituras y no existe ninguna instrucción para dicha operación. La información del vendedor está situada en los registros CPUID 0 y 1 y especifican un nombre de vendedor, en ASCII, para la implementación del procesador. Todos los bytes después del final de la cadena hasta el byte 16 son iguales a cero. Los caracteres ASCII anteriores se colocan en los registros de menor numeración y en las posiciones de bytes con menor numeración. El registro CPUID 4 proporciona información general a nivel de aplicación acerca de las características del procesador. Contiene un conjunto de bits indicadores usados para indicar si una característica determinada está soportada en el modelo de procesador. Cuando un bit es uno, la característica está soportada, cuando es 0, la característica no está soportada. Conforme se añaden (o se eliminan) nuevas características de modelos de procesadores futuros, la presencia (o eliminación) de las nuevas características se indicará mediante nuevos bits de características. El registro CPUID 4 está dividido lógicamente en dos mitades, las cuales contienen información general de la característica y la capacidad, pero que tienen diferentes modelos de uso y capacidades de acceso; esta información refleja el estado de cualquier característica habilitada o deshabilitada. Tanto la mitad superior como la mitad inferior del registro CPUID 4 pueden ser accedidas a través de la instrucción move indirecto con registro; dependiendo de la implementación, la latencia para este acceso puede ser grande y este procedimiento de acceso no es apropiado para las versiones de código de baja latencia usando autoselección. Además, la mitad superior del registro CPUID 4 es accesible también usando la instrucción de comprobación de característica; la latencia para este acceso es comparable a la de la instrucción de comprobación de bit y este procedimiento permite versiones de código de baja latencia usando auto-selección.

Los principios de funcionamiento de la arquitectura z proporcionan una instrucción Almacena Lista Extendida de Funcionalidades (Store Facility List Extended, STFLE) que, al igual que el registro CPUID de Intel, proporciona al software el conocimiento de las características (o niveles de arquitectura) de las unidades centrales de procesamiento (CPU) o procesadores subyacentes. La instrucción STFLE tiene el formato mostrado en la Tabla 1 a continuación.

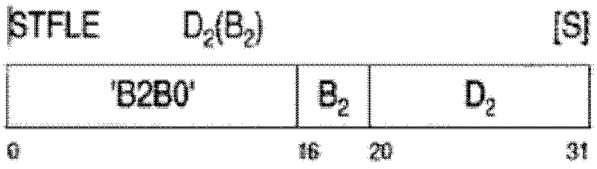


TABLA 1

La instrucción STFLE (TABLA 1) comprende un campo de bits (0-15) de código de operación, un campo B2 (16-19) de registro y un campo D2 (20-31) de desplazamiento (inmediato). La ejecución de la instrucción STFLE por una máquina, almacena una lista de bits que proporcionan información acerca de las funcionalidades en una posición de memoria de programa determinada añadiendo los contenidos del registro especificado por el campo B2 de la instrucción al campo inmediato D2 de 12 bits, en el que la posición de memoria empieza en la doble palabra (8 bytes, una palabra es de 4 bytes) especificada por la dirección del segundo operando ((B2) + D2). La dirección de la posición de memoria de programa en la arquitectura z es sometida a una traducción dinámica de direcciones (Dynamic Address Translation, DAT).

Los bits reservados son bits que no están asignados actualmente para representar una funcionalidad. Para las dobles palabras situadas más a la izquierda en las que se asignan los bits de las funcionalidades, los bits reservados se almacenan como ceros. Las dobles palabras situadas a la derecha de la doble palabra en la que se asigna el bit de funcionalidad de numeración más alta para un modelo pueden almacenarse o no. Las excepciones de acceso y los eventos PER no se reconocen para las dobles palabras que no se almacenan. El tamaño del segundo operando,

en dobles palabras, es uno más que el valor especificado en los bits 56-63 del registro general 0. Los bits restantes del registro general 0 no están asignados y deberían contener ceros; de lo contrario, el programa puede no funcionar de manera compatible en el futuro.

Cuando el tamaño del segundo operando es suficientemente grande para contener todos los bits de las funcionalidades asignadas para un modelo, entonces, la lista completa de funcionalidades es almacenada en la posición del segundo operando, los bits 56-63 del registrador general 0 se actualizan para contener una menos que el número de dobles palabras necesarias para contener todos los bits de las funcionalidades asignadas para el modelo, y se establece el código de condición 0.

Cuando el tamaño del segundo operando no es suficientemente grande para contener todos los bits de las funcionalidades asignadas para un modelo, entonces solo se almacenan el número de dobles palabras especificadas por el tamaño del segundo operando, los bits 56-63 del registro general 0 se actualizan para contener una menos que el número de dobles palabras necesarias para contener todos los bits de las funcionalidades asignadas para el modelo, y se establece el código de condición 3.

La ejecución de la instrucción resulta en el establecimiento de un valor de código de condición, el valor del código de condición es salvado durante un cambio de contexto la palabra de estado del programa (Program Status Word, PSW).

Condiciones especiales

El segundo operando debe ser designado en un límite de doble palabra; si no, se reconoce una excepción de la excepción.

Código de condición resultante:

0 Lista completa de funcionalidades almacenada

1 -

2 -

3 Lista incompleta de funcionalidades almacenada

Excepciones del programa:

- Acceso (almacena, segundo operando)
- Operación (si la funcionalidad store-facility-list-extended no está instalada)
- Especificación

Notas de programación:

El rendimiento de STORE FACILITY LIST EXTENDED/ALMACENA LISTA EXTENDIDA FUNCIONALIDADES puede ser considerablemente más lento que el de simplemente comprobar un byte en el almacenamiento. Los programas que necesitan comprobar con frecuencia la presencia de una funcionalidad (por ejemplo, un código de dos caminos en el que la funcionalidad es usada en un camino pero no en el otro) debería ejecutar la instrucción STORE FACILITY LIST EXTENDED una vez durante la inicialización. Posteriormente, el programa puede comprobar la presencia de la funcionalidad examinando el resultado almacenado, usando una instrucción tal como TEST UNDER MASK.

Cuando se establece el código de condición 0, los bits 56-63 del registro general 0 se actualizan para indicar el número de dobles palabras almacenadas. Si el programa elige ignorar los resultados en el registro general 0, entonces debería asegurarse de que todo el segundo operando en el almacenamiento es establecido a cero antes de ejecutar STORE FACILITY LIST EXTENDED.

La TABLA 2 muestra los bits STFLE asignados por una arquitectura z anterior y sus significados. Un bit se pone a uno independientemente del modo arquitectónico actual si su significado es verdadero. Un significado se aplica al modo de arquitectura actual a menos que se especifique que se aplique a un modo arquitectónico específico.

Los bits no asignados están reservados para indicar nuevas funcionalidades; estos bits pueden ser almacenados como unos en el futuro.

La lista de funcionalidades de la arquitectura z de la técnica anterior se define tal como se muestra en la Tabla 2, a continuación:

TABLA 2

Significado del bit cuando es igual uno:

0 Las instrucciones marcadas "N3" en las figuras de resumen de instrucciones en los capítulos 7 y 10 de la arquitectura z están instaladas.

1 El modo arquitectónico de arquitectura z está instalado.

2 El modo arquitectónico de arquitectura z está activo. Cuando este bit es cero, el modo arquitectónico ESA/390 está activo.

3 La funcionalidad de mejora de DAT está instalada en el modo arquitectónico de arquitectura z. La funcionalidad de mejora de DAT incluye las instrucciones INVALIDATE DAT TABLE ENTRY (IDTE) y COMPARE AND SWAP AND PURGE (CSPG).

4 INVALIDATE DAT TABLE ENTRY (IDTE) realiza la operación de invalidación y limpieza, limpiando selectivamente entradas combinadas de la tabla de regiones y segmentos cuando se invalidan una entrada o entradas de la tabla de segmentos. IDTE realiza también la operación de limpieza mediante ASCE. A menos que el bit 4 sea igual a uno, IDTE simplemente purga todos los TLB. El bit 3 es uno si el bit 4 es uno.

5 INVALIDATE DAT TABLE ENTRY (IDTE) realiza la operación de invalidación y limpieza, limpiando selectivamente las entradas de la tabla combinada de regiones y segmentos cuando se invalidan una entrada o unas entradas en la tabla de regiones. Los bits 3 y 4 son igual a uno si el bit 5 es igual a uno.

6 La funcionalidad de reutilización de ASN y LX está instalada en el modo arquitectónico arquitectura z.

7 La funcionalidad store-facility-list-extended está instalada.

8 La funcionalidad DAT mejorada está instalada en el modo arquitectónico arquitectura z.

9 La funcionalidad sense-running-status está instalada en el modo arquitectónico arquitectura z.

10 La funcionalidad condicional-SSKE está instalada en el modo arquitectónico arquitectura z.

11 La funcionalidad de configuration-topology está instalada en el modo arquitectónico arquitectura z.

16 La funcionalidad extended_translation 2 está instalada.

17 La asistencia message-security está instalada.

18 La funcionalidad long_displacement está instalado en el modo arquitectónico arquitectura z.

19 La funcionalidad long_displacement tiene un alto rendimiento. El bit 18 es uno si el bit 19 es uno.

20 La funcionalidad HFP-multiply-and-add/subtract está instalada.

21 La funcionalidad extended-inmmediate está instalada en el modo arquitectónico arquitectura z.

22 La funcionalidad extended_translation 3 está instalada en el modo arquitectónico arquitectura z.

23 La funcionalidad HFP-unnormalized-extesion está instalada en el modo arquitectónico arquitectura z.

24 La funcionalidad ETF2_enhancement está instalada.

25 La funcionalidad store-clock-fast está instalada en el modo arquitectónico arquitectura z.

26 La funcionalidad parsing-enhancement está instalada en el modo arquitectónico arquitectura z.

27 La funcionalidad move-with-optional-especifications está instalada en el modo arquitectónico arquitectura z.

28 La funcionalidad TOD-clock-steering está instalada en el modo arquitectónico arquitectura z.

30 La funcionalidad ETF3-enhancement está instalada en el modo arquitectónico arquitectura z.

31 La funcionalidad extract-CPU-time está instalada en el modo arquitectónico arquitectura z.

32 La funcionalidad compare-and-swap-and-store está instalada en el modo arquitectónico arquitectura z.

33 La funcionalidad compare-and-swap-and-store 2 está instalada en el modo arquitectónico arquitectura z.

34 La funcionalidad general-instructions-extension está instalada e en el modo arquitectónico arquitectura z.

35 La funcionalidad execute-extensions está instalada en el modo arquitectónico arquitectura z.

39 Asignado para el uso interno de IBM.

41 Las funcionalidades floating-point-support-enhancement (FPR-GR-transfer, FPS-sign-handling y DFP rounding) están instaladas en el modo arquitectónico arquitectura z.

42 La funcionalidad DFP (decimal-floating-point) está instalada en el modo arquitectónico arquitectura z.

43 La funcionalidad DFP (decimal-floating-point) tiene un alto rendimiento. El bit 42 es uno si el bit 43 es uno.

44 La instrucción PFPO está instalada en el modo arquitectónico arquitectura z.

Una instrucción puede realizar una única función en una arquitectura o, en algunos casos, cualquiera de entre una pluralidad de funciones seleccionables. Las funciones seleccionables definidas para una instrucción pueden ser diferentes de una máquina a otra. Por ejemplo, una instrucción multifunción, cuando es introducida por primera vez en un conjunto de instrucciones, puede tener sólo unas pocas funciones seleccionables. Un conjunto de instrucciones posterior puede introducir más funciones seleccionables a la instrucción multifunción introducida previamente. En una realización, a una máquina virtual puede asignarse un subconjunto de la función seleccionable del procesador físico de manera que una instrucción que se ejecuta en un procesador lógico de la máquina virtual puede consultar una lista de las funciones disponibles del procesador lógico y sólo se devuelven las funciones asignadas a la máquina virtual, aunque el procesador físico puede realizar más funciones seleccionables. En una realización, esto se consigue mediante una funcionalidad Function-Indicating-Instruction Interception Facitity (FIIIF) que permite a un hipervisor atrapar o interceptar la ejecución de esta función de consulta por un cliente (máquina virtual), con el fin de presentar la lista reducida de las funciones disponibles. En otra realización, el hipervisor específica, por ejemplo mediante una máscara de bits, el conjunto de funciones a indicar al cliente, y la función de consulta de la instrucción multifunción entrega esta lista. Además, en una realización, una instrucción que se ejecuta en el procesador lógico experimentará una excepción de programa si intenta realizar una función seleccionable seleccionada.

Un ejemplo de una instrucción que tiene funciones seleccionables es la instrucción CIPHER MESSAGE de la arquitectura z.

La instrucción CIPHER MESSAGE (KM) puede realizar cualquiera de entre una pluralidad de funciones de cifrado de mensaje. Una de las funciones proporcionadas por el CIPHER MESSAGE es realizar una consulta al procesador para obtener una lista de bits relacionados con las funciones de cifrado de mensaje soportadas por el procesador.

El formato de la instrucción CIPHER MESSAGE (TABLA 3) es tal como se indica a continuación, en la que R1 designa un primer Registro General, y R2 designa un segundo Registro General.

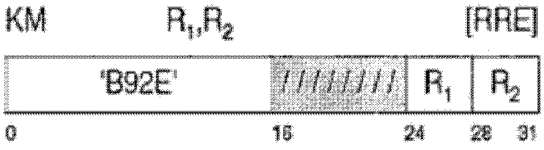


TABLA 3

La ejecución de la instrucción CIPHER MESSAGE (TABLA 3) es tal como se indica a continuación.

Se realiza una función especificada por el código de función en el registro general 0 implícito.

Los bits 16-23 de la instrucción se ignoran.

Las posiciones de bit 57-63 del registro general 0 contienen el código de función.

Los códigos de función asignados actualmente para CIPHER MESSAGE y CIPHER MESSAGE WITH CHAINING, respectivamente (0-3 y 18-20) se muestran en la Tabla 4. Todos los demás códigos de función no están asignados. Para las funciones de cifrado, el bit 56 es el bit modificador que especifica si debe realizarse una operación de cifrado o de descifrado. El bit modificador se ignora para todas las demás funciones. Todos los demás bits del registro general 0 se ignoran.

El registro general 1 implícito contiene la dirección lógica del byte situado más a la izquierda del bloque de parámetros en el almacenamiento. En el modo de direccionamiento de 24 bits, el contenido de las posiciones de bit 40-63 del registro general 1 constituye la dirección, y el contenido de las posiciones de bit 0-39 se ignoran. En el modo de direccionamiento de 31 bits, el contenido de las posiciones de bit 33-63 del registro general 1 constituye la dirección, y el contenido de las posiciones de bit 0-32 se ignora. En el modo de direccionamiento de 64 bits, el contenido de las posiciones de bit 0-63 del registro general 1 constituye la dirección.

La función de consulta proporciona los medios para indicar la disponibilidad de las otras funciones. Los contenidos

de los registros generales especificados por los campos de la instrucción (R1, R2), y R2+1 se ignoran para la función de consulta.

Para todas las demás funciones, el segundo operando (especificado por R2) es cifrado tal como se especifica por el código de función, usando una clave criptográfica en el bloque de parámetros, y el resultado es colocado en la posición del primer operando.

Para CIPHER MESSAGE WITH CHAINING, el cifrado usa también un valor de encadenamiento inicial en el bloque de parámetros, y el valor de encadenamiento es actualizado como parte de la operación. El uso del registro para el direccionamiento de 24 bits se muestra en la TABLA 5.

El campo R1 designa un registro general y debe designar un registro con numeración par; de lo contrario, se reconoce una excepción de especificación.

El campo R2 designa un par par-impar de registros generales y debe designar un registro con numeración par; de lo contrario, se reconoce una excepción de especificación.

La posición del byte situado más a la izquierda del primer operando y el segundo operando es especificada por el contenido de los registros generales R1 y R2, respectivamente. El número de bytes en la posición del segundo operando se especifica en el registro general R2+1. El primer operando tiene la misma longitud que el segundo operando.

Como parte de la operación, las direcciones en los registros generales R1 y R2 se incrementan en el número de bytes procesados, y la longitud en el registro general R2+1 se decrementa en el mismo número. La formación y actualización de las direcciones y la longitud depende del modo de direccionamiento.

En el modo de direccionamiento de 24 bits, los contenidos de las posiciones de bit 40-63 de los registros generales R1 y R2 constituyen las direcciones del primer operando y el segundo operando, respectivamente, y los contenidos de las posiciones de bit 0-39 se ignoran; los bits 40-63 de las direcciones actualizadas reemplazan los bits correspondientes en los registros generales R1 y R2, los acarreos fuera de la posición de bit 40 de la dirección actualizada se ignoran, y el contenido de las posiciones de bit 32-39 de los registros generales R1 y R2 se establecen a ceros. En el modo de direccionamiento de 31 bits, los contenidos de las posiciones de bit 33-63 de los registros generales R1 y R2 constituyen las direcciones del primer operando y el segundo operando, respectivamente, y se ignoran los contenidos de las posiciones de bit 0-32; los bits 33-63 de las direcciones actualizadas reemplazan los bits correspondientes en los registros generales R1 y R2, los acarreos fuera de la posición de bit 33 de la dirección actualizada se ignoran, y el contenido de la posición de bit 32 de los registros generales R1 y R2 se establece a cero. En el modo de direccionamiento de 64 bits, los contenidos de las posiciones de bit 0-63 de los registros generales R1 y R2 constituyen las direcciones del primer operando y el segundo operando, respectivamente; los bits 0-63 de las direcciones actualizadas reemplazan los contenidos de los registros generales R1 y R2, los acarreos fuera de la posición de bit 0 se ignoran.

En los modos de direccionamiento de 31 bits y de 24 bits, el contenido de las posiciones de bit 32-63 del registro R2+1 forma un entero binario sin signo de 32 bits que especifica el número de bytes en el primer operando y el segundo operando, y los contenidos de las posiciones de bit 0-31 se ignoran; los bits 32-63 del valor actualizado reemplazan los bits correspondientes en el registro general R2+1. En el modo de direccionamiento de 64 bits, el contenido de las posiciones de bit 0-63 del registro general R2+1 forma un entero binario sin signo de 64 bits que especifica el número de bytes en el primer operando y el segundo operando; y el valor actualizado reemplaza los contenidos del registro general R2+1.

En el modo de direccionamiento de 24 bits o de 31 bits, el contenido de las posiciones de bit 0-31 de los registros generales R1, R2, y R2+1, siempre permanecen inalterados. La Tabla 5 muestra los contenidos de los registros generales descritos anteriormente.

En el modo de acceso de registro, los registros de acceso 1, R1 y R2 especifican los espacios de direcciones que contienen el bloque de parámetros, el primer operando y el segundo operando, respectivamente.

El resultado se obtiene como si el procesamiento se iniciara en el extremo izquierdo del primer operando y el segundo operando y procediera hacia la derecha, bloque por bloque. La operación se termina cuando el número de bytes en el segundo operando tal como se especifica en el registro general R2+1 han sido procesados y colocados en la posición del primera operando (llamada terminación normal) o cuando un número de bloques, determinado por la CPU, que es menor que la longitud del segundo operando, han sido procesados (denominada terminación parcial). El número de bloques, determinado por la CPU, depende del modelo, y puede ser un número diferente cada vez que se ejecuta la instrucción. Generalmente, el número de bloques, determinado por la CPU, es distinto de cero. En ciertas situaciones inusuales, este número puede ser cero, y el código de estado 3 puede establecerse sin progreso. Sin embargo, la CPU proporciona protección contra la recurrencia sin fin de este caso sin progreso.

Los resultados en la posición de primer operando y el campo valor de encadenamiento son impredecibles si se produce alguna de las situaciones siguientes:

El campo clave criptográfica se superpone a cualquier parte del primer operando.

El campo valor de encadenamiento se superpone sobre cualquier parte del primer operando o el segundo operando.

El primero operando y el segundo operando se superponen de manera destructiva. Se dice que los operandos se superponen de manera destructiva cuando la posición del primer operando sería usada como una fuente después de que los datos hayan sido movidos a la misma, suponiendo que el procesamiento se realiza de izquierda a derecha y un byte en cada momento.

Cuando la operación termina debido a una terminación normal, el código de condición se pone a 0 y el valor resultante en R2+1 es cero. Cuando la operación termina debido a una terminación parcial, se establece el código de condición 3 y el valor resultante en R2+1 es distinto de cero.

Cuando se reconoce un evento PER de alteración de almacenamiento, se almacenan menos de 4K bytes adicionales en las posiciones del primer operando antes de informar acerca del evento.

Cuando la longitud del segundo operando es inicialmente cero, el bloque de parámetros, el primer operando y el segundo operando no son accedidos, los registros generales R1, R2 y R2+1 no se cambian y se establece el código de condición 0.

Cuando los contenidos de los campos R1 y R2 son iguales, los contenidos de los registros designados se incrementan sólo en el número de bytes procesados, no en dos veces el número de bytes procesados.

Tal como lo observan otras CPUs y programas de canal, las referencias al bloque de parámetros y a los operandos de almacenamiento pueden ser referencias de acceso múltiple, los accesos a estas posiciones de almacenamiento no son necesariamente bloques concurrentes, y la secuencia de estos accesos o referencias no está definida.

En ciertas situaciones inusuales, la ejecución de la instrucción puede completarse estableciendo el código de condición 3 sin actualizar los registros y el valor de encadenamiento para reflejar la última unidad del primer operando y el segundo operando procesados. El tamaño de la unidad procesada en este caso depende de la situación y del modelo, pero está limitado de manera que la parte del primer operando y el segundo operando que han sido procesados y no informados no se solapen en el almacenamiento. En todos los casos, se establecen los bits de cambio y se comunican los eventos de alteración de almacenamiento PER, cuando sea aplicable, para las posiciones de todos los primeros operandos procesados.

Las excepciones de acceso pueden ser comunicadas para una parte mayor de un operando que es procesado en una única ejecución de la instrucción; sin embargo, las excepciones de acceso no son reconocidas para las posiciones más allá de la longitud de un operando ni para posiciones alejadas más de 4K bytes de la posición actual que está siendo procesada.

Los códigos de función para CIPHER MESSAGE son los siguientes.

Código	**Función**	**Tamaño bloque parm. (bytes)**	**Tamaño bloque de datos (bytes)**
0	KM-Query	16	-
1	KM-DEA	8	8
2	KM-TDEA-128	16	8
3	KM-TDEA-192	24	8
18	KM-AES-128	16	16
19	KM-AES-192	24	16
20	KM-AES-256	32	16
Explicación: - No aplicable			

TABLA 4

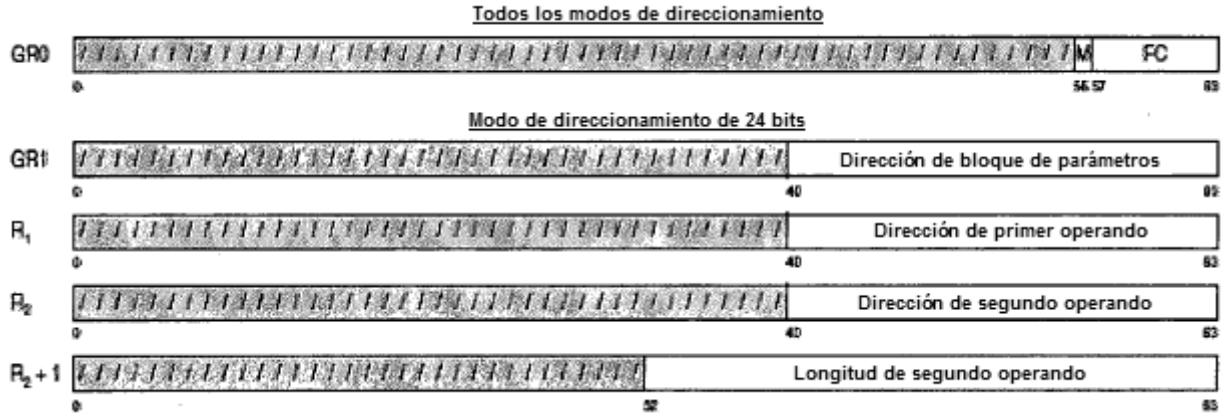


TABLA 5

Usando la instrucción CIPHER MESSAGE como un ejemplo, una máquina ejemplar puede implementar funciones CIPHER MESSAGE. En la implementación ejemplar, los procesadores servidores pueden implementar todas las funciones mostradas (código de función 0-3 y 18-20). Un sistema operativo (OS) servidor (o hipervisor) puede crear una o más máquinas virtuales para sistemas operativos cliente. Una máquina virtual podría ser definida para una arquitectura de nivel anterior, que no tiene instrucciones CIPHER MESSAGE.

Según una realización, si hubiese instalada una funcionalidad de bloqueo de instrucciones y si las instrucciones CIPHER MESSAGE se designaran como instrucciones bloqueadas para una máquina virtual, la máquina virtual no permitiría la ejecución de la instrucción CIPHER MESSAGE por los programas que se están ejecutando en la máquina virtual, a pesar de que la máquina servidor subyacente soporta instrucciones CIPHER MESSAGE. Un intento de ejecutar una instrucción CIPHER MESSAGE en la máquina virtual resultaría en una verificación de programa (excepción de programa).

Según otra realización, si hubiese instalada una funcionalidad de bloqueo de funciones y sólo se permite un subconjunto de las funciones CIPHER MESSAGE (códigos de función 0-3 por ejemplo) en una máquina virtual, la máquina virtual permitiría la ejecución de CIPHER MESSAGE pero no permitiría la ejecución de la instrucción CIPHER MENSAJE de las instrucciones CIPHER MESSAGE que tienen un código de función diferente de 0-3 por los programas que se están ejecutando en la máquina virtual, a pesar de que la máquina servidor subyacente soporta instrucciones CIPHER MESSAGE que soportan los códigos de función (0-3 y 18-20). Un intento de ejecutar una instrucción CIPHER MESSAGE que tiene códigos de función diferentes de 0-3, tal como cualquiera de entre 18-20) resultaría en una verificación de programa (excepción de programa).

En otra realización, si hubiese instalada una funcionalidad de comprobación/consulta de funciones y sólo se permite un subconjunto de las funciones CIPHER MESSAGE (códigos de función 0-3 por ejemplo) en una máquina virtual, la ejecución de una consulta CIPHER MESSAGE de las funciones CIPHER MESSAGE devolvería sólo los códigos de función 0-3, a pesar de que la máquina servidor subyacente soporta los códigos de función 0-3 y 18-20.

Funcionalidad de bloqueo de instrucciones:

Con referencia a la Fig. 8, en la misma se muestra la función de una funcionalidad de bloqueo de instrucciones de nivel de arquitectura virtual (Virtual Architecture Level, VAL). Cada instrucción a ejecutar en la máquina virtual (tal como se muestra en las instrucciones de la columna de almacenamiento), incluye un código de operación. En algunas implementaciones, el código de operación es un único campo en la instrucción 901 902 903 904. En otras implementaciones, los códigos de operación pueden estar distribuidos en más de un campo de la instrucción 905 (OpCode || OpCode) 906 (OpCode || OpCode). Preferiblemente, los circuitos, los microcódigo o una combinación de los mismos, determinarían, en base al código de operación, si la instrucción a ejecutar está soportada o no por la máquina virtual actual. Si no está soportada, se indicaría una interrupción del programa, por ejemplo, una excepción de programa, y la instrucción se suprimiría.

En una implementación, el código de operación de la instrucción a ejecutar sería usado como índice para una tabla 907 de códigos de operación para localizar una entrada asociada con el código de operación. La entrada encontrada incluiría un código que indica el nivel de máquina (ML) soportado por el código de operación. En otra implementación, cada máquina virtual tendría una tabla de códigos de operación y la entrada en la tabla indicaría si el código de operación está soportado por la máquina virtual.

Con referencia a la Fig. 9, el código (nivel de máquina (ML)) 1002 obtenido de la tabla 907 sería comparado 1008 con una entrada de descripción de estado (IBC) 1008 de una tabla 1004 de descripción de estado de la máquina virtual, y si el código 1002 de nivel de máquina es mayor que la entrada 1008 de descripción de estado IBC, la instrucción se ejecutaría normalmente 1007, de lo contrario, el intento de ejecución resultaría en una excepción de

programa 1006. En otra realización, campos de la instrucción además, o diferentes, del campo de código de operación pueden ser usados como índice en la tabla 907 de códigos de operación. Por ejemplo, un código de operación puede tener campos reservados (para ser 0 o ignorados) en una arquitectura de máquina anterior, que se emplean en niveles de arquitectura más nuevos para proporcionar nuevas funciones. Una realización incluiría estos bits con el código de operación como índice en la tabla 907 de código de operación. En otra realización, la tabla 907 de códigos de operación puede tener campos adicionales al campo ML usado para indicar el uso permitido de bits reservados en la instrucción asociada. Por ejemplo, si la instrucción tiene 4 bits de reserva, la tabla ML puede contener 0000 si todos los bits deben ser 0, o 1 en bits seleccionados en los que el valor 1 indica que los bits correspondientes previamente reservados del campo pueden ser 0 ó 1 (permitiendo la función recién introducida de la instrucción para la máquina virtual).

Funcionalidad de comprobación/consulta de instrucción:

Si hay instalada una funcionalidad FUNCTION BLOCKING FACILITY de la funcionalidad de comprobación/consulta de instrucción (Fig. 10), la entrada 1001 de la tabla de códigos de operación puede incluir además, en una realización, un campo de código de función (FCx) 1003 (o un puntero a una tabla 1108 de códigos de función). El campo 1003 de código de función (o la entrada 1107 de la tabla 1108 de códigos de función) se compara 1103 con el código de función a ejecutar 1102. Si el código de función es igual, se permite 1105 que la instrucción use el código de función, si el código de función no es igual 1103, la ejecución de la instrucción causa una interrupción del programa, tal como una excepción de programa o una excepción de especificación (verificación de programa) 1104.

Con referencia a la Fig. 11, si hay instalada una funcionalidad FUNCTION TEST/QUERY BLOCKING FACILITY de la funcionalidad de comprobación/consulta de instrucción, si se ejecuta cualquier instrucción 1109 de consulta para determinar la función instalada de la instrucción, sólo se devuelven 1205 los códigos de las funciones permitidas por la máquina virtual. En una realización, la máquina virtual proporciona una tabla 1108 con bits significativos, que es usada por la máquina virtual para responder a dichas consultas. En otra realización, se proporciona una máscara (no mostrada) a la máquina virtual con la que puede realizarse una función AND con los códigos de las funciones instaladas de la máquina servidor para crear un resultado de los códigos 1107 de función permitidos de la instrucción en la máquina virtual.

Con referencia a la Fig. 8, en la misma se muestran ejemplos de formatos de instrucciones de arquitectura z. El formato 901 representa un formato de 2 bytes en el que el código de operación (Op) ocupa el byte de orden alto, y los campos R1 y R2 de registro general ocupan 4 bits respectivos del byte restante. El formato 902 representa un formato de instrucción de solo un código de operación de 2 bytes. El formato 903 representa una instrucción 4 bytes (palabra) que tiene un código de operación (Operation code, Op) de 1 byte, seguido por 3 campos de registro, (R1, X2 y B2) y a continuación un campo inmediato denominado campo de desplazamiento (D2). El formato 904 representa una instrucción de 4 bytes que tiene un código de operación de 4 bytes (Op), seguido de un campo de registro (B2) de 4 bits y, a continuación, un campo inmediato denominado campo de desplazamiento (D2). El formato 905 representa una instrucción de 4 bytes que tiene un código de operación (Op) de 1 byte seguido de una máscara M1 de 4 bits, seguido por una extensión de código de operación (Op) de 4 bits y un campo de 4 bit reservado, seguido de un campo inmediato (I2) de 12 bits. El formato 906 representa una instrucción de 6 bytes que tiene un código de operación (Op) de 1 byte seguido por 3 campos de registro, (R1, X2 y B2) y, a continuación, un campo inmediato denominado campo de desplazamiento (DL2) seguido de un campo inmediato (DH2) de 8 bits y una extensión de código de operación (Op) de 8 bits.

Con referencia a las Figs. 8 y 9, en una realización, cuando una instrucción es extraída para su ejecución por un procesador lógico que ejecuta una máquina virtual, se realiza una consulta a una tabla 907 de códigos de operación, usando el código o códigos de operación de la instrucción como un argumento de búsqueda. Si se encuentra una entrada 1001 para la instrucción, la entrada incluye información 1002 1003 para determinar la información de permiso de instrucción. En una realización preferida, una entrada incluye un campo 1002 que especifica un código (ML) que indica el nivel de máquina de la arquitectura que soporta la instrucción. Una descripción 1004 de estado es proporcionada para cada máquina virtual. La descripción del estado incluye un campo (IBC) 1008 que representa el nivel de máquina de la arquitectura que la máquina virtual debe simular. Si 1005, el nivel de máquina de la arquitectura que soporta la instrucción (ML) es mayor que el nivel de máquina de la arquitectura que la máquina virtual debe simular (IBC), se señaliza una excepción de programa (verificación de programa) y, en una realización, puede suprimirse la ejecución de la instrucción. Por otro lado, si el nivel de máquina de la arquitectura que soporta la instrucción (ML) no es mayor que el nivel de máquina de la arquitectura que la máquina virtual debe simular (IBC), se permite la ejecución de la instrucción.

En algunos entornos, se proporcionan instrucciones que son capaces de ejecutar cualquiera de entre una pluralidad de funciones (tal como la instrucción CIPHER MESSAGE descrita anteriormente). La selección de la función por una instrucción puede ser realizada especificando un código de función (FC) que representa la función. El código de función puede ser especificado indirectamente por la instrucción o puede ser especificado explícitamente por los bits o los campos de la instrucción, por ejemplo. En algunos casos, ciertos códigos de función pueden ser implementados inicialmente (0-3 por ejemplo) en un nivel de arquitectura de máquina, y los códigos de función adicionales pueden ser añadidos en niveles de arquitectura de máquinas posteriores. La máquina virtual puede ser provista con la capacidad de permitir sólo la ejecución de los códigos de función de un nivel de arquitectura más

antiguo, y bloquear (prevenir) la ejecución de las funciones de un nivel de arquitectura más reciente.

Con referencia a la Fig. 10, esto puede conseguirse teniendo un campo de código de función (FCx) 1003 en la entrada 1001 de la tabla de códigos de operación. Cuando una instrucción está a punto de ser ejecutada, el campo FCx 1003 especifica la lista de códigos de función permitidos a devolver en lugar de los códigos de función actualmente soportados por el procesador servidor. En una realización, el campo FCx 1003 de la entrada de la tabla de códigos de operación es concatenado con el campo IBC 1008 para formar un índice 1006 para una tabla 1108 de FCx para localizar una entrada que comprende los códigos de las funciones permitidas (FCs) 1107. Los FCs 1107 permitidos se comparan con el FC especificado por la instrucción 1102 (en la instrucción Cipher Message, los bits 1102 del registro general 0 1101 contienen el FC 1102 especificado). Si 1103 el valor FC está permitido 1105, se permite la ejecución normal de la función representada por el bit FC. Si 1103 el valor FC no está permitido 1104, se lleva a cabo un evento de excepción de programa, tal como una excepción de especificación (verificación de programa). De manera similar, cuando se ejecuta una operación 1109 de consulta/comprobación de función (tal como la operación QUERY de la instrucción CIPHER MESSAGE), los bits FCX de la entrada 1003 de la tabla de códigos de operación se concatenan 1106 con los bits IBC 1008 para formar un índice para la tabla FCX 1108 para localizar los FCs 1107 permitidos para la instrucción cuyo código de operación localiza la entrada 1001 de la tabla de códigos de operación. A continuación, los FCc permitidos son devueltos 1105 a la posición especificada por la operación consulta/comprobación de función.

En una realización, cuando los bits FCX son 0, no se realiza ningún acceso a la tabla FCx 1108 y cualquier código de función indicado por la instrucción correspondiente es usado sin traducción.

En una realización, otras modificaciones de la arquitectura para las instrucciones pueden usar el mismo mecanismo descrito para los códigos de función. En este caso, por ejemplo, la instrucción 905 a un nivel arquitectónico tiene los bits entre el campo extensión de código de operación y el campo I2, reservados (0000). Preferiblemente, se comprueba si los bits reservados son 0 para garantizar que la instrucción se realizará apropiadamente en un entorno en el que los bits no nulos todavía no soportan la función soportada. Una arquitectura más nueva implementa una nueva función usando uno o más de los bits reservados para identificar la nueva función. En un ejemplo, estos 4 bits reservados (Res) pueden formar un índice para la Tabla FCx 1108 con el fin de determinar si están soportados, tal como se muestra para los bits FC 1103 en la Fig. 10. En este caso, la concatenación sería 0||IBC||FCx para los códigos de función, y 1||IBC||FCx para la comprobación 1103 de permiso de la nueva función. En lugar de comparar el FC 1102 con los FCs 1107 permitidos, el campo Res de la instrucción 905 sería comparado con los bits FCS 1107 permitidos para determinar 1103 si la función está permitida.

En otra realización, el campo Res de la instrucción 905 podría ser concatenado como si se tratara de una tercera extensión de código de operación de los códigos de operación 905 para formar un índice para la tabla 907 de códigos de operación para determinar si la función introducida con el campo está permitida.

Como parte de, o después de, la extracción de una instrucción, una CPU puede determinar ciertos atributos de la instrucción, por ejemplo, el número de operandos, el tipo de operandos (almacenamiento o registro), requisitos de alineación de operandos y requisitos de autorización. En un entorno de emulación, esta determinación puede ser el resultado de una tabla de consulta sencilla que usa el código de operación como un índice; en una implementación de hardware de CPU de alto rendimiento, la determinación puede ser integrada en la circuitería de decodificación de instrucciones del procesador. De esta manera, según se está decodificando una instrucción, el nivel de máquina para esa instrucción puede ser comparado con un valor programable que indica el nivel de máquina permitido. Para una instrucción que está siendo decodificada, que tiene un nivel de máquina mayor que el valor permitido, su extracción, ejecución o terminación sería bloqueada dependiendo de la implementación y puede generarse una excepción dependiente de máquina para un código de operación no válido.

La funcionalidad de nivel de arquitectura virtual puede introducir un atributo adicional asociado con cada instrucción: el nivel de máquina en el que la instrucción fue introducida por primera vez a la arquitectura. Este nivel de máquina puede ser un punto numérico codificado en un continuo (por ejemplo, 10,2, que significa la décima generación de máquina en el segundo nivel de firmware), o puede ser simplemente un valor con relación al nivel de máquina más reciente (por ejemplo, 2 [o -2], que significa que la instrucción fue introducida dos generaciones de máquina antes de la máquina actual).

Con referencia a la Fig. 12, en una realización, se bloquean instrucciones 1258 específicas de manera que no sean ejecutadas por un procesador. Se establece 1251 un valor de bloqueo instrucción. Una instrucción es extraída 1252 para ser ejecutada por el procesador, comprendiendo la instrucción un código de operación, siendo soportada la instrucción por el procesador. Cuando la instrucción debe ser ejecutada, se realiza una comparación del valor de bloqueo de instrucción con la instrucción (o el código de operación de la instrucción) para determinar si se permite o no la ejecución. En respuesta 1254 al valor de bloqueo de instrucción que permite la ejecución de la instrucción, se ejecuta 1255 la instrucción extraída por el procesador, y en respuesta 1254 al valor de bloqueo de instrucción que no permite la ejecución 1256 de la instrucción, se bloquea la ejecución de la instrucción extraída y se genera un evento de excepción de programa.

Con referencia a la Fig. 13, en una realización, el procesador es un procesador lógico que ejecuta una máquina

virtual, en el que la extracción es realizada por el procesador lógico. Se realiza una determinación 1254 del valor de bloqueo de instrucción de la máquina virtual, en el que el valor de bloqueo de instrucción se establece en el procesador lógico que tiene uno o más procesadores físicos, en el que la instrucción está soportada por el uno o más procesadores físicos, en el que en respuesta al valor de bloqueo de instrucción se permite la ejecución de la instrucción, la ejecución es realizada 1352 por el procesador lógico. Si la instrucción está bloqueada 1256 se comunica un evento de excepción de programa.

Con referencia a la Fig. 14, en una realización, el procesador es uno o más procesadores físicos de un procesador lógico que ejecuta una máquina virtual, en el que el valor de bloqueo de instrucción se establece 1451 en el uno o más procesadores físicos, en el que la extracción es realizada por el uno o más procesadores físicos. El procesador físico compara 1452 el valor de bloqueo de instrucción con la instrucción a ejecutar para determinar si la instrucción debe ser bloqueada, y el procesador físico o bien realiza la instrucción 1454 o bien causa un evento 1455 de excepción de programa.

Con referencia a la Fig. 15, en una realización, se define el valor de bloqueo de instrucción para que la máquina virtual bloquee la ejecución de la instrucción, estableciendo 1551 el valor de bloqueo de instrucción en respuesta a la habilitación de la máquina virtual para usar el procesador 1553 físico; se establece 1552 otro valor de bloqueo de instrucción, definiéndose el otro valor de bloqueo de instrucción para otra máquina virtual que tiene otro procesador lógico, estableciéndose el otro valor de bloqueo de instrucción en respuesta a la habilitación de la otra máquina virtual para usar el procesador 1553 físico; y en respuesta a otro valor de bloqueo de instrucción que permite la ejecución 1254 de la instrucción, se permite la ejecución 1255 de la instrucción por el otro procesador lógico; y en respuesta al otro valor de bloqueo de instrucción que no permite la ejecución 1254 de la instrucción , no se permite la ejecución 1256 de la instrucción por el otro procesador lógico.

En una realización, se define el valor de bloqueo de instrucción para la máquina virtual para bloquear la ejecución de la instrucción, estableciendo el valor de bloqueo de instrucción en respuesta a la habilitación de la máquina virtual para usar el procesador físico, se establece otro valor de bloqueo de instrucción, definiéndose el otro valor de bloqueo de instrucción para otra máquina virtual que tiene otro procesador lógico, estableciendo el otro valor de bloqueo de instrucción en respuesta a la habilitación de la otra máquina virtual para usar el procesador físico; y en respuesta al otro valor de bloqueo de instrucción que permite la ejecución de la instrucción , se permite la ejecución de la instrucción por el procesador físico mientras la otra máquina virtual está habilitada para usar el procesador físico, y en respuesta al otro valor de bloqueo de instrucción que no permite la ejecución de la instrucción, no se permite la ejecución de la instrucción por el procesador físico mientras la otra máquina virtual está habilitada para usar el procesador físico.

Con referencia a la Fig. 12, en una realización, la instrucción 1258 es la instrucción permitida si la instrucción 1258 que emplea un código de función permitido asociado con una función 1259 seleccionada de entre una pluralidad de funciones seleccionables, en el que la instrucción es la instrucción no si la instrucción emplea un código de función no permitido, en el que los códigos de función son especificados por la instrucción.

En una realización, se realiza una determinación acerca de si la instrucción es la instrucción permitida asociando el código de operación de la instrucción con el valor de bloqueo de instrucción.

Lo indicado anteriormente puede ser útil para comprender la terminología y la estructura de una realización del sistema informático. Las realizaciones pueden no limitarse a la arquitectura z o a la descripción proporcionada de la misma. Las realizaciones pueden aplicarse ventajosamente a otras arquitecturas de ordenador de otros fabricantes de ordenadores con las enseñanzas de la presente memoria.

REIVINDICACIONES

1. Un procedimiento implementado por ordenador para bloquear que funciones específicas a ser realizadas por una instrucción sean ejecutadas por un procesador lógico que ejecuta una máquina virtual, en el que el procedimiento comprende:

establecer (1251) un valor (1008) de bloqueo de instrucción definido para la máquina virtual, para bloquear la ejecución de una instrucción, en el que el valor de bloqueo de instrucción indica las funciones permitidas;

extraer (1252), por dicho procesador lógico, una instrucción (1101) a ser ejecutada por el procesador lógico, en el que la instrucción comprende un código de operación y especifica un código (1102) de función de entre una pluralidad de códigos de función, en el que el código de función se usa para seleccionar la función a realizar por la instrucción, en el que la instrucción es soportada por uno o más procesadores físicos;

determinar (1253) un valor de bloqueo de instrucción para la instrucción extraída que tiene el código de función, comparando (1253) el valor de bloqueo de instrucción definido para la máquina virtual con el código de operación de la instrucción extraída;

en respuesta al valor de bloqueo de instrucción que permite (1007, 1255) la ejecución de la instrucción que tiene el código de función, ejecutar la instrucción extraída por el procesador lógico; y

en respuesta al valor de bloqueo de instrucción que no permite (1006, 1256) la ejecución de la instrucción que tiene el código de función, bloquear la ejecución de la instrucción extraída y causar un evento de excepción de programa.

2. Procedimiento según la reivindicación 1, que comprende además:

que el valor (1008) de bloqueo de instrucción es definido para la máquina virtual para bloquear la ejecución de funciones a ser realizadas por la instrucción, estableciéndose el valor de bloqueo de instrucción en respuesta a la habilitación de la máquina virtual para ser ejecutada en el procesador físico;

establecer (1351) otro valor de bloqueo de instrucción definido para otra máquina virtual que se ejecuta en otro procesador lógico, estableciéndose el otro valor de bloqueo de instrucción en respuesta a la habilitación de la otra máquina virtual para ser ejecutada en el procesador físico; y

en respuesta al otro valor de bloqueo de instrucción que permite (1352) la ejecución de la instrucción que tiene el código de función, permitir la ejecución de la instrucción por el otro procesador lógico; y

en respuesta al otro valor de bloque de instrucción que no permite (1256) la ejecución de la instrucción que tiene el código de función, bloquear la ejecución de la instrucción por el otro procesador lógico.

3. Procedimiento según la reivindicación 1, en el que la determinación de un valor de bloqueo de instrucción comprende además las etapas de:

usar el código de operación como índice en una tabla (907) para localizar el valor de bloqueo de instrucción definido para la máquina virtual, en el que el valor de bloqueo de instrucción comprende un campo de permiso (1002, 1003);

usar el campo de permiso para determinar las funciones permitidas, comparándolo con el código de función de la instrucción extraída; y

si la función es una función permitida, determinar (1007) que se permite la ejecución de la instrucción; y

si la función es una función no permitida, determinar (1006) que no se permite la ejecución de la instrucción.

4. Procedimiento según la reivindicación 1, que comprende además las etapas de:

determinar si la instrucción es la instrucción permitida, asociando el código de operación de la instrucción con el valor de bloqueo de instrucción.

5. Sistema informático para bloquear la ejecución de instrucciones específicas por un procesador; que comprende:

una memoria;

un procesador en comunicación con la memoria, en el que el procesador comprende un elemento de extracción de instrucciones para extraer instrucciones desde una memoria y uno o más elementos de ejecución para ejecutar las instrucciones extraídas;

en el que el sistema informático está configurado para realizar el procedimiento según se reivindica en las reivindicaciones 1 a 4.

6. Programa de ordenador que puede ser cargado en la memoria del sistema informático según la reivindicación 5 que comprende partes de código de software para realizar, cuando dicho programa es ejecutado en el sistema informático, las etapas del procedimiento según se reivindica en las reivindicaciones 1 a 4.

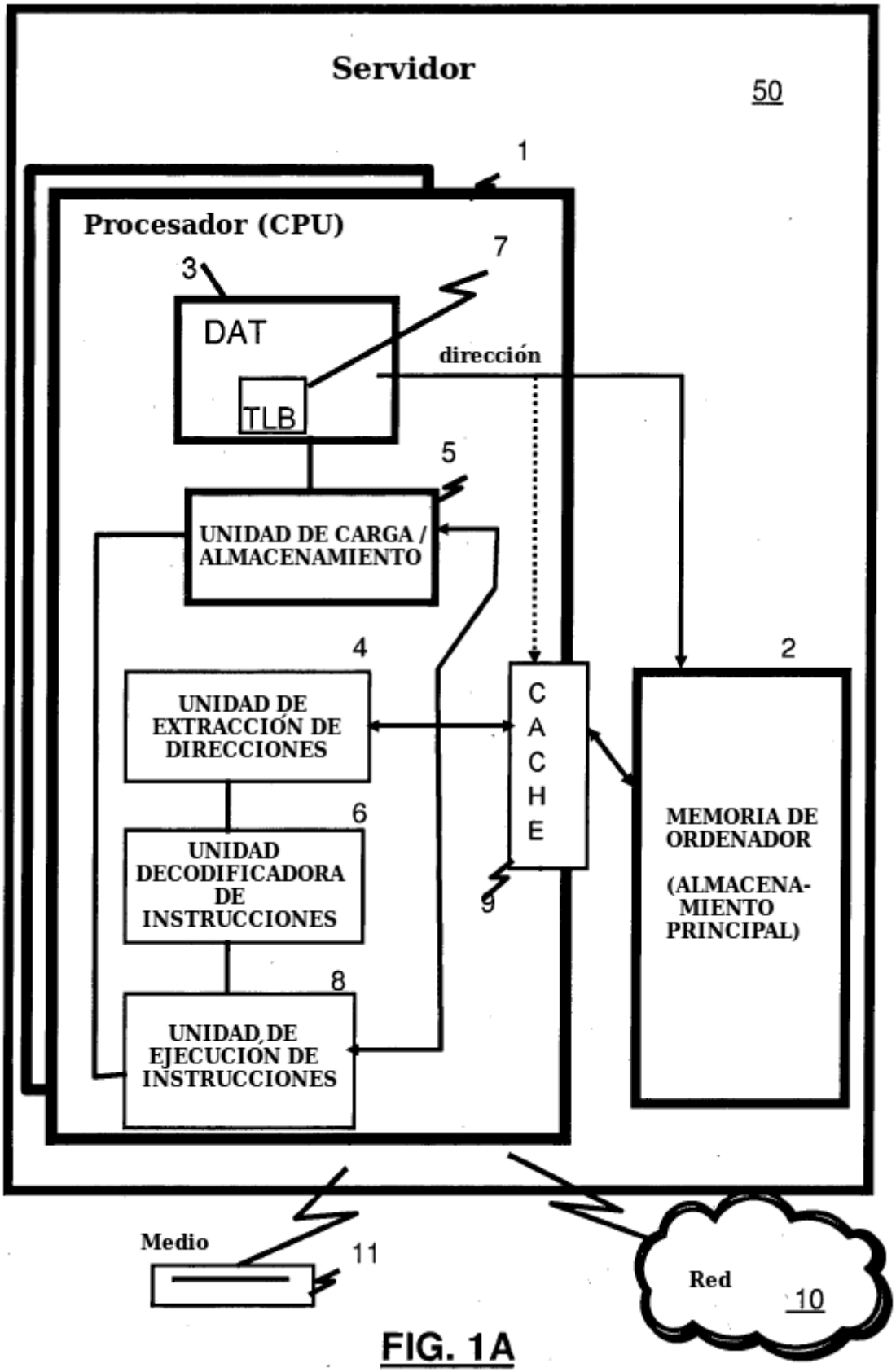

Servidor
50
1
Procesador (CPU)
7
3
DAT
dirección
TLB
5
UNIDAD DE CARGA / ALMACENAMIENTO
4
2
UNIDAD DE EXTRACCIÓN DE DIRECCIONES
C A C H E
6
UNIDAD DECODIFICADORA DE INSTRUCCIONES
MEMORIA DE ORDENADOR
(ALMACENA-MIENTO PRINCIPAL)
9
8
UNIDAD DE EJECUCIÓN DE INSTRUCCIONES
Medio
11
Red
10

FIG. 1A

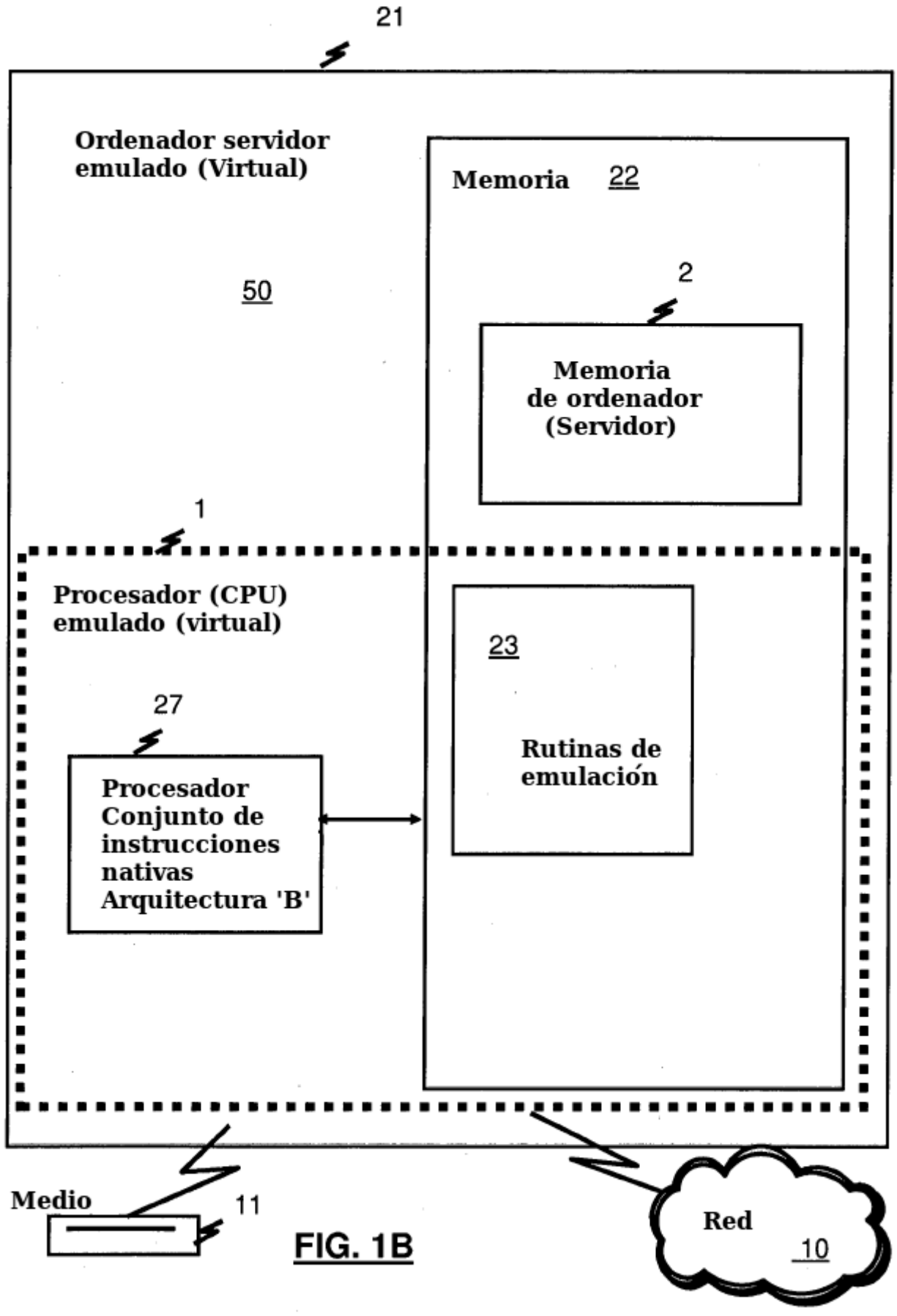

21
Ordenador servidor emulado (Virtual)
Memoria 22
50
2
Memoria de ordenador (Servidor)
1
Procesador (CPU) emulado (virtual)
23
Rutinas de emulación
27
Procesador Conjunto de instrucciones nativas Arquitectura 'B'
Medio
11
Red
10

FIG. 1B

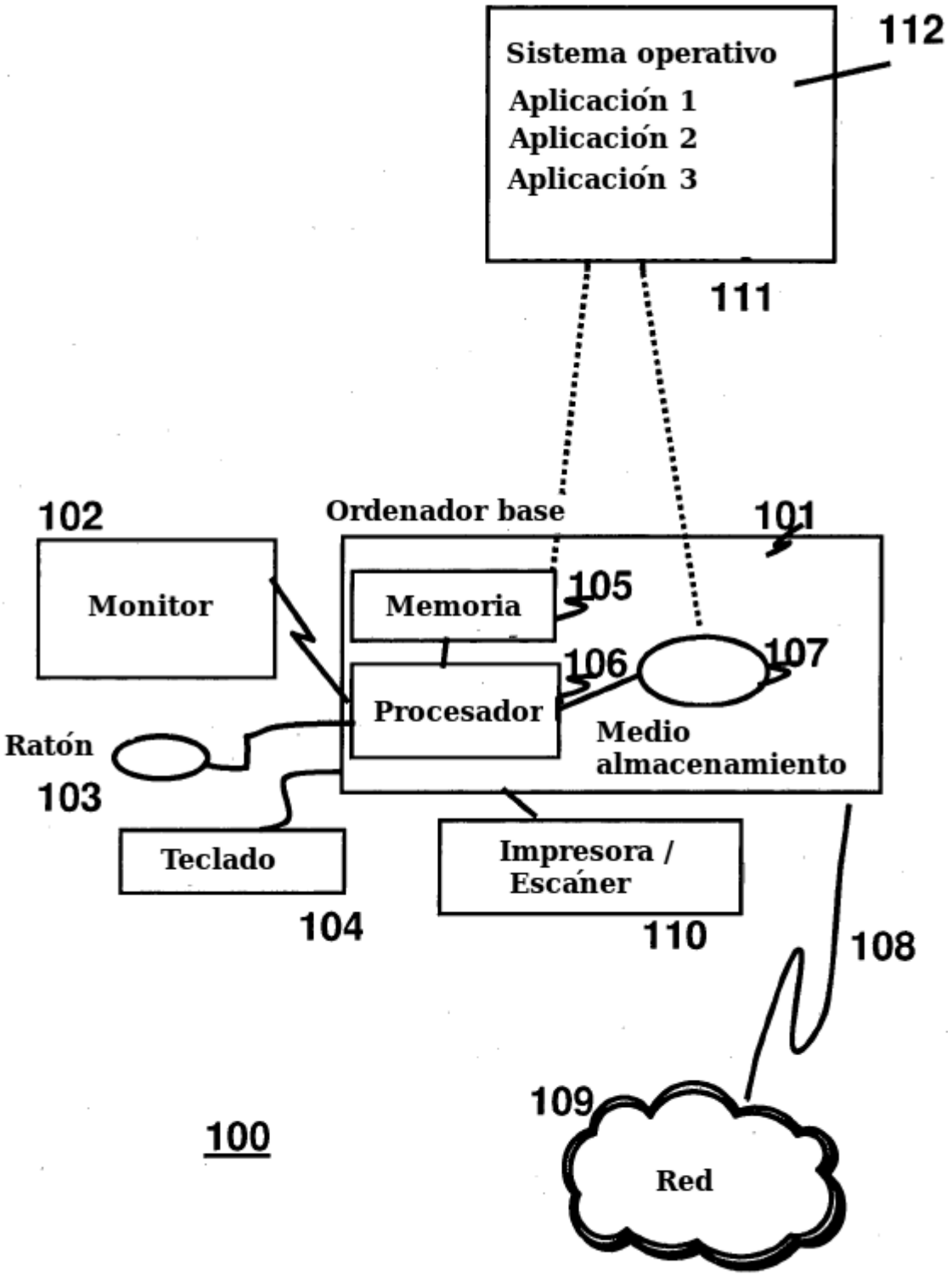

Sistema operativo
Aplicación 1
Aplicación 2
Aplicación 3
112
111
102
Ordenador base
101
Monitor
Memoria
105
106
107
Procesador
Medio almacenamiento
Ratón
103
Teclado
104
Impresora / Escáner
110
108
109
Red
100

FIG. 1C

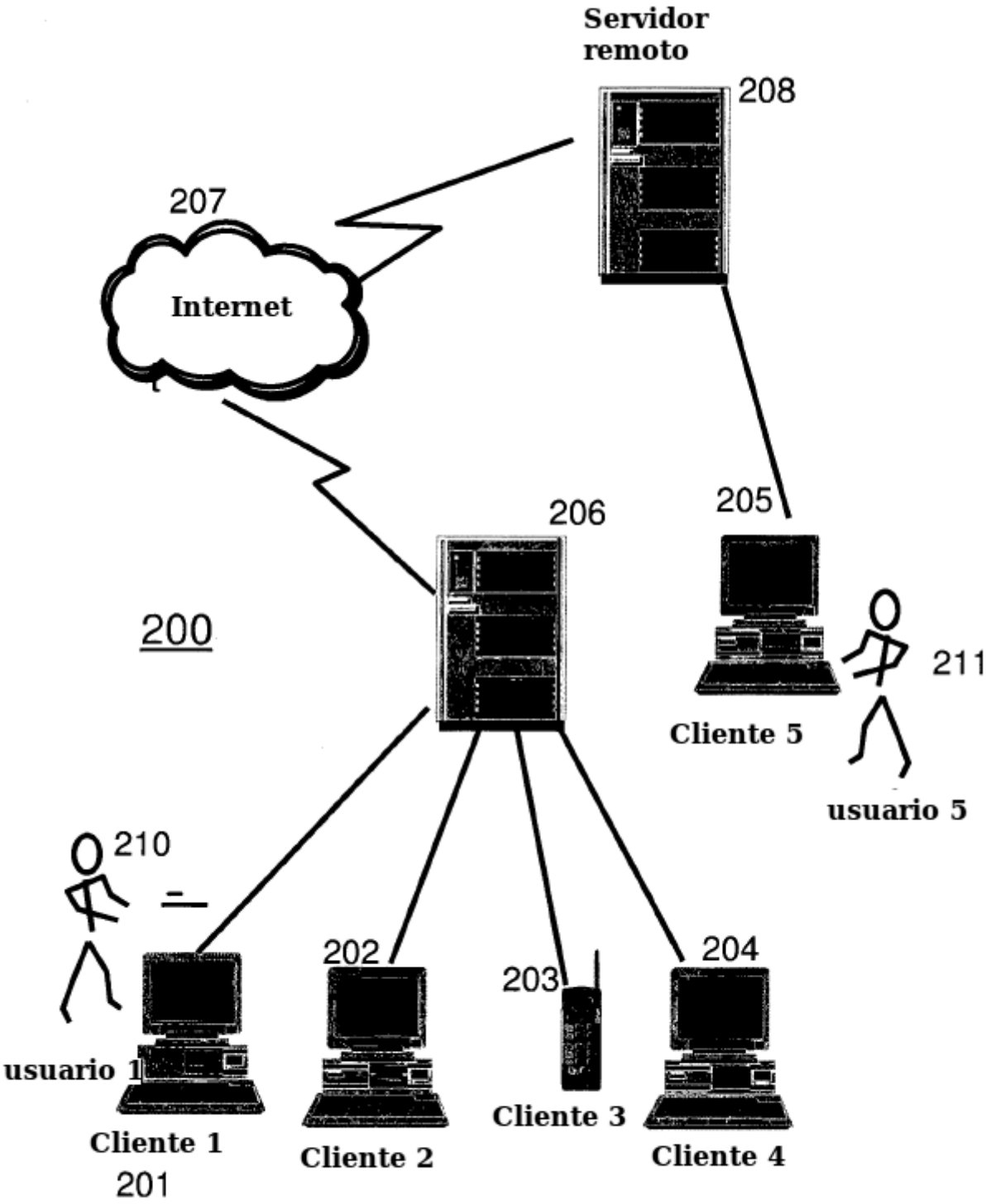

Servidor remoto
208
207
Internet
206
205
200
211
Cliente 5
usuario 5
210
202
203
204
usuario 1
Cliente 1
201
Cliente 2
Cliente 3
Cliente 4

FIG. 2

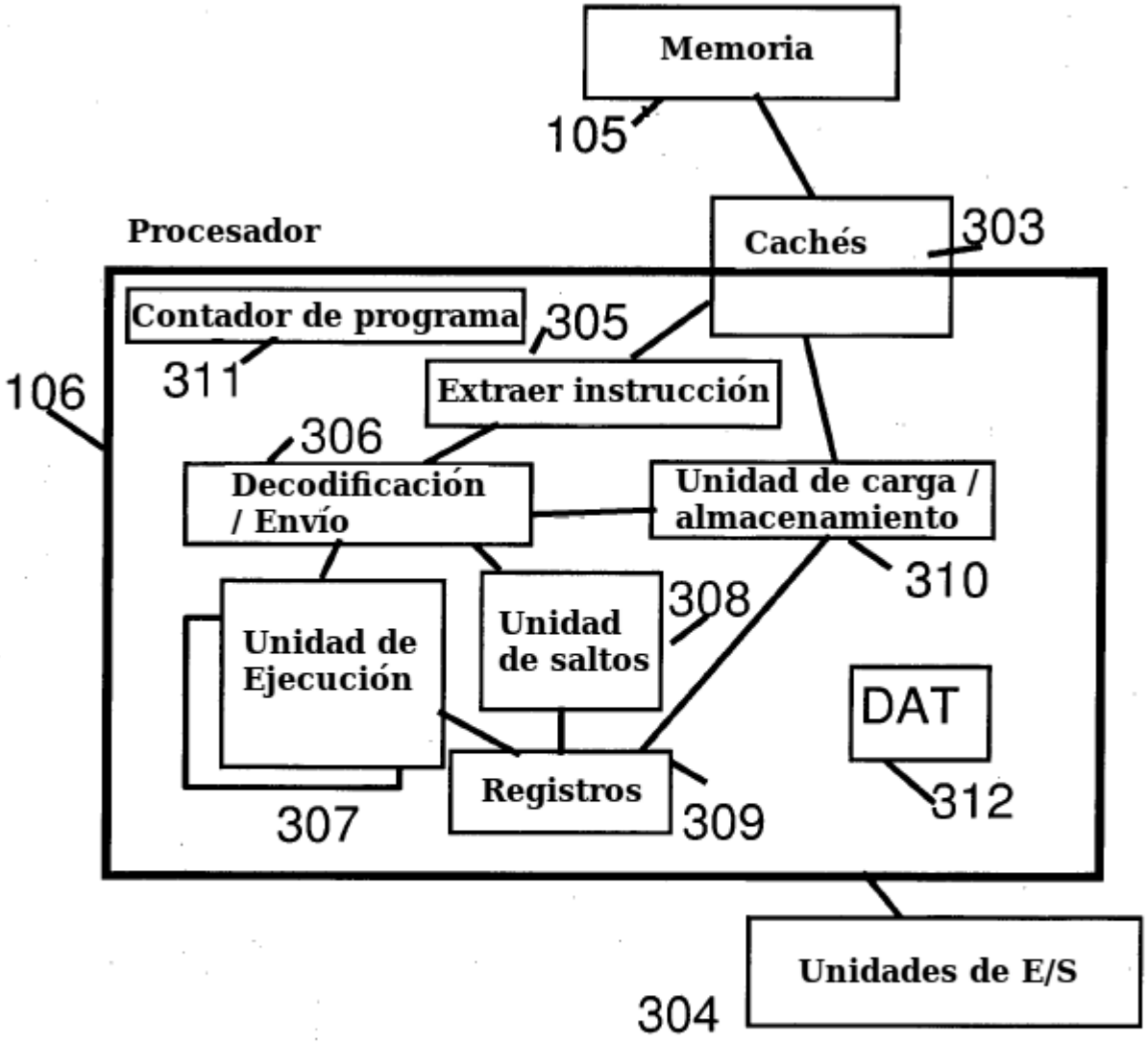

Memoria
105
Procesador
Cachés
303
Contador de programa
305
311
106
Extraer instrucción
306
Decodificación / Envío
Unidad de carga / almacenamiento
310
308
Unidad de Ejecución
Unidad de saltos
DAT
Registros
307
309
312
Unidades de E/S
304

FIG. 3

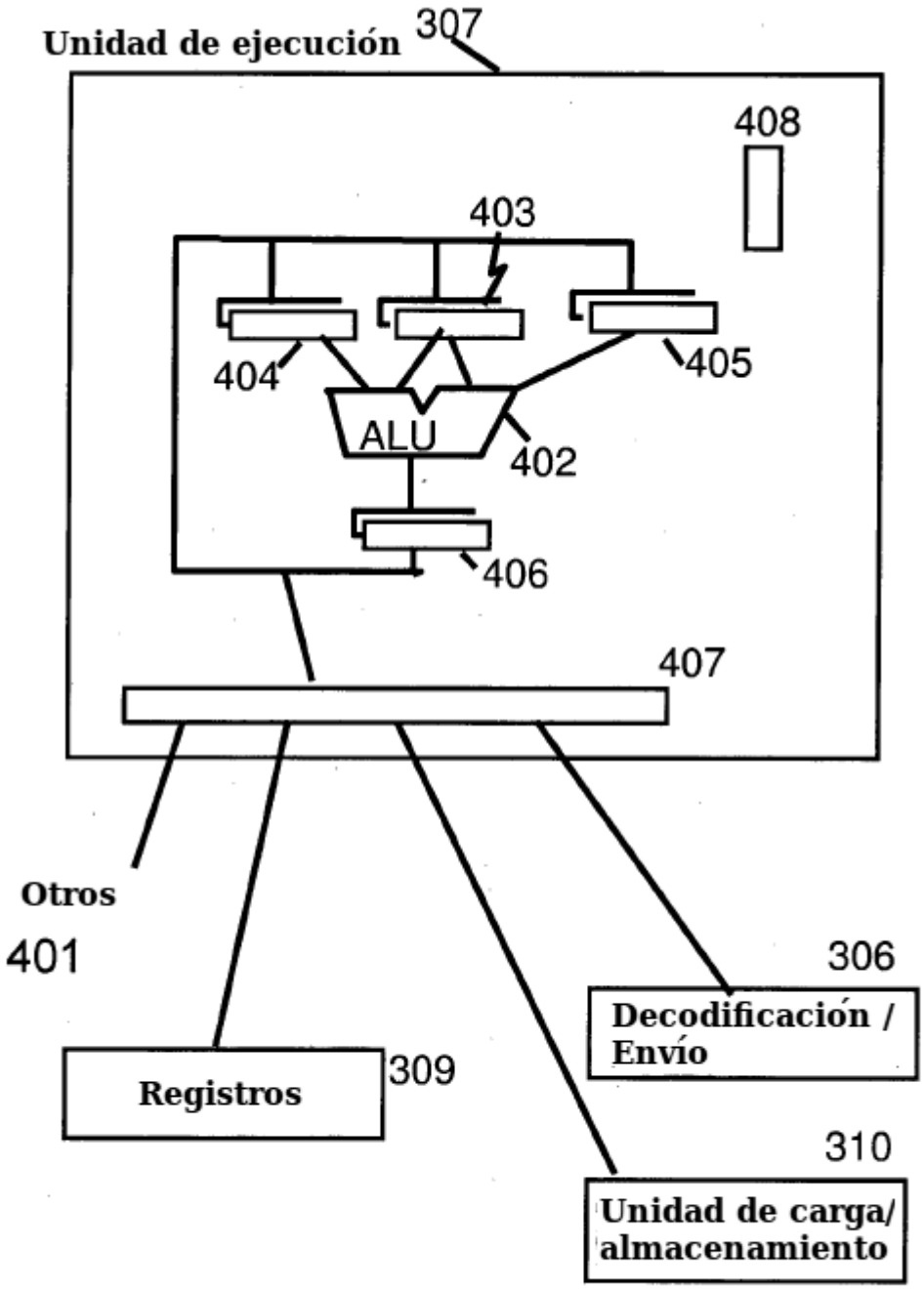

Unidad de ejecución
307
408
403
404
405
ALU
402
406
407
Otros
401
306
Decodificación / Envío
Registros
309
310
Unidad de carga/ almacenamiento

FIG. 4A

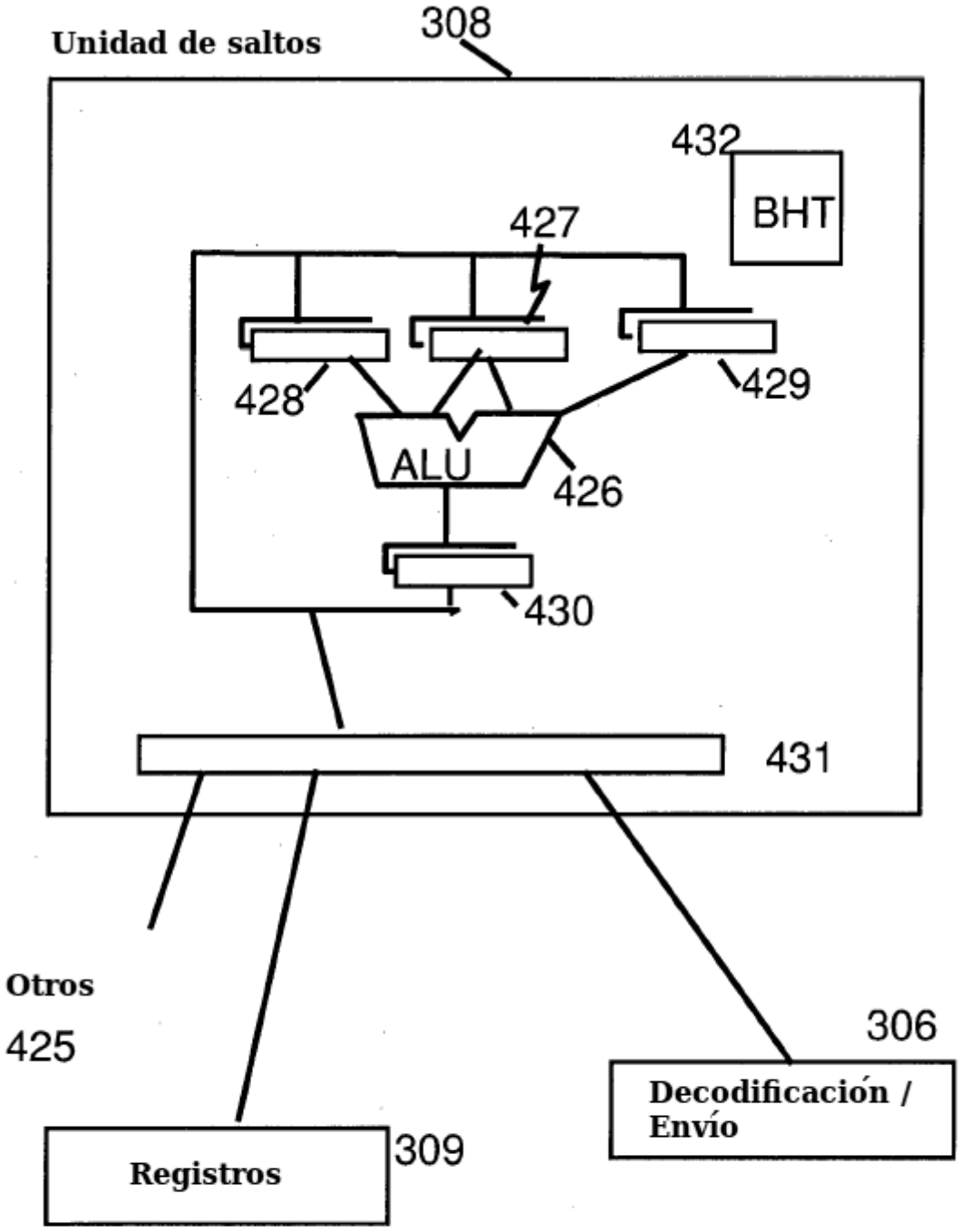

Unidad de saltos
308
432
BHT
427
428
429
ALU
426
430
431
Otros
425
306
Decodificación / Envío
Registros
309

FIG. 4B

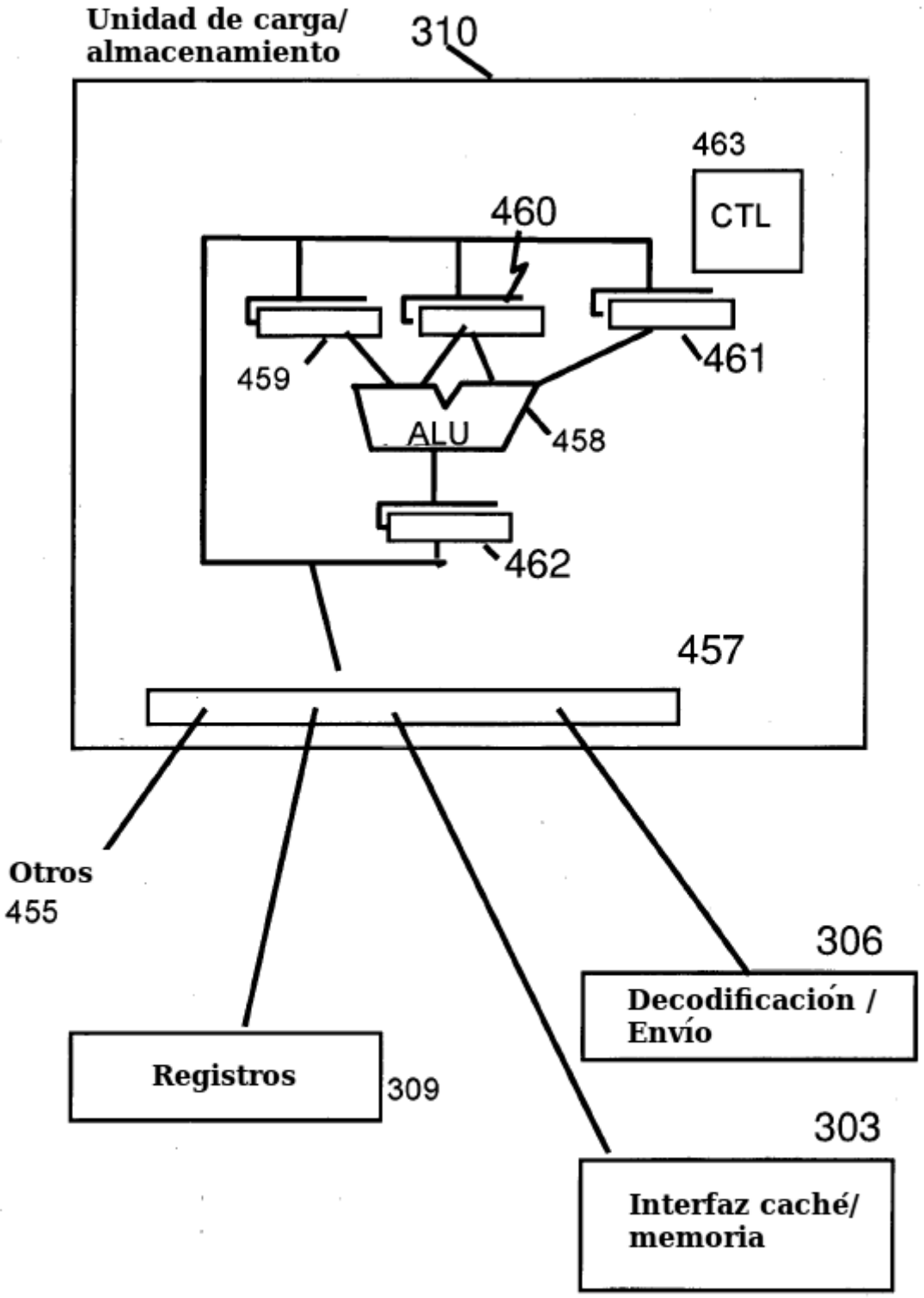

Unidad de carga/
almacenamiento
310
463
CTL
460
459
461
ALU
458
462
457
Otros
455
306
Decodificación /
Envío
Registros
309
303
Interfaz caché/
memoria

FIG. 4C

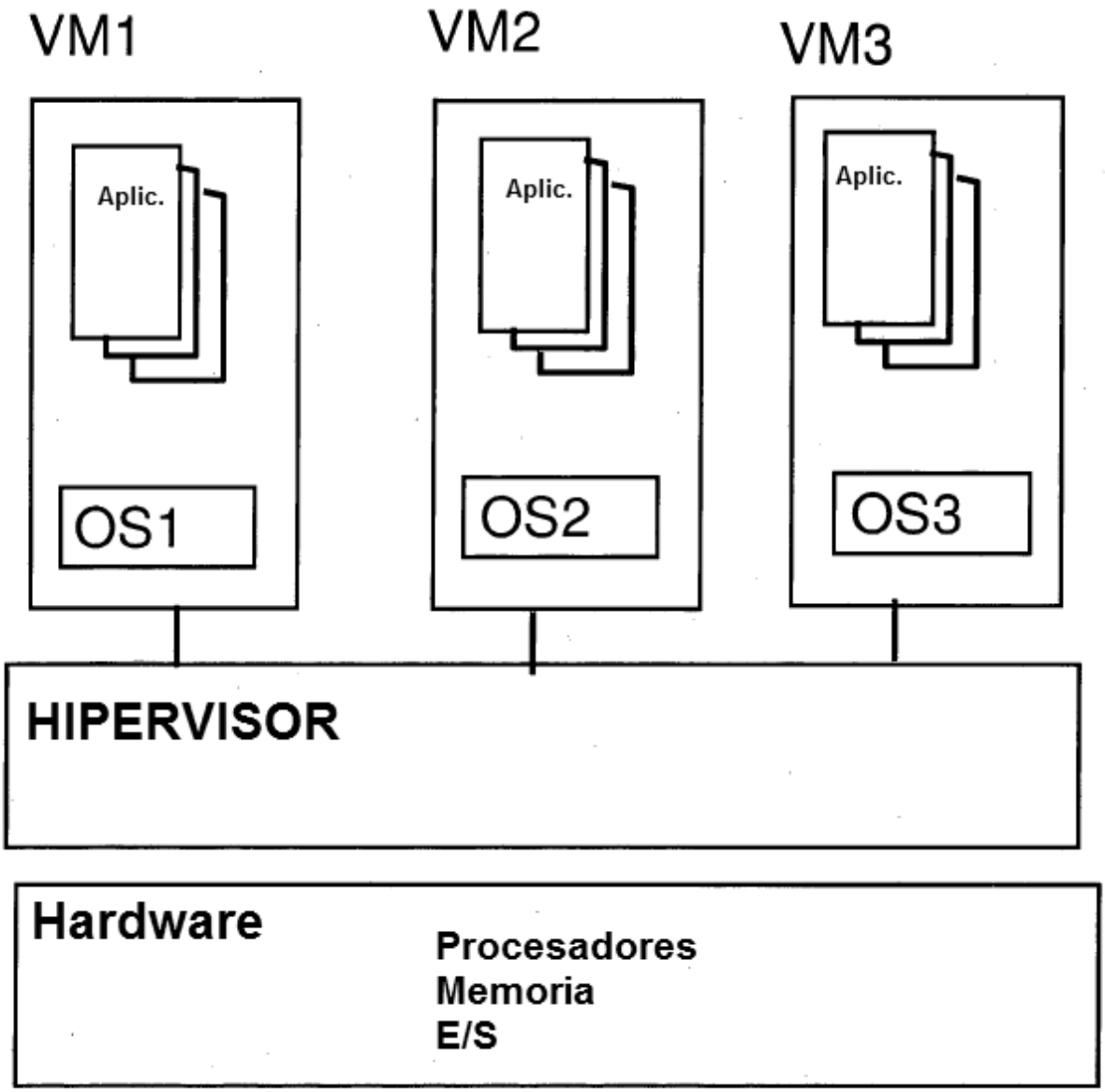

VM1
VM2
VM3
Aplic.
Aplic.
Aplic.
OS1
OS2
OS3
HIPERVISOR
Hardware
Procesadores
Memoria
E/S

FIG. 5

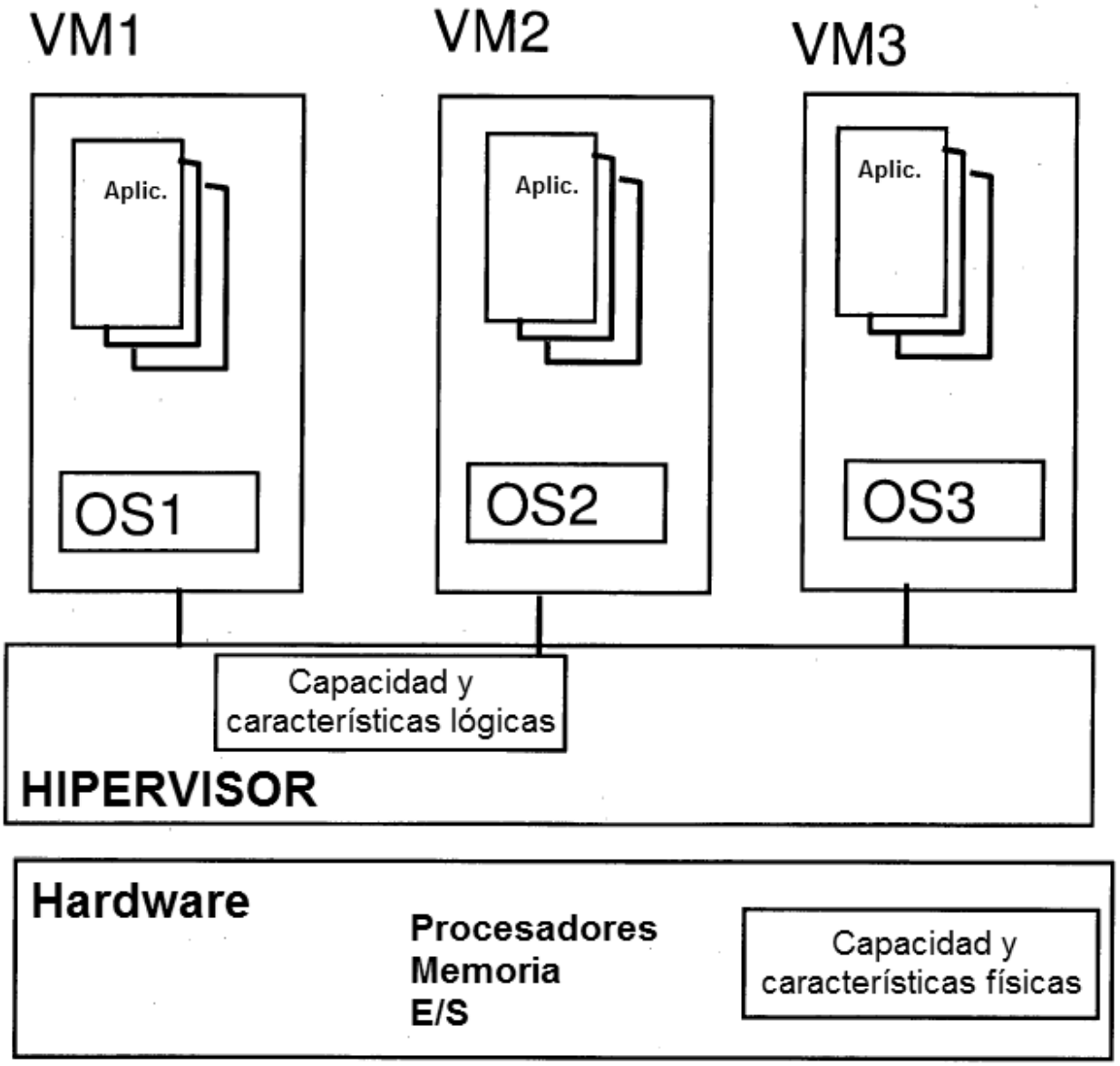

VM1
VM2
VM3
Aplic.
Aplic.
Aplic.
OS1
OS2
OS3
Capacidad y características lógicas
HIPERVISOR
Hardware
Procesadores
Memoria
E/S
Capacidad y características físicas

FIG. 6

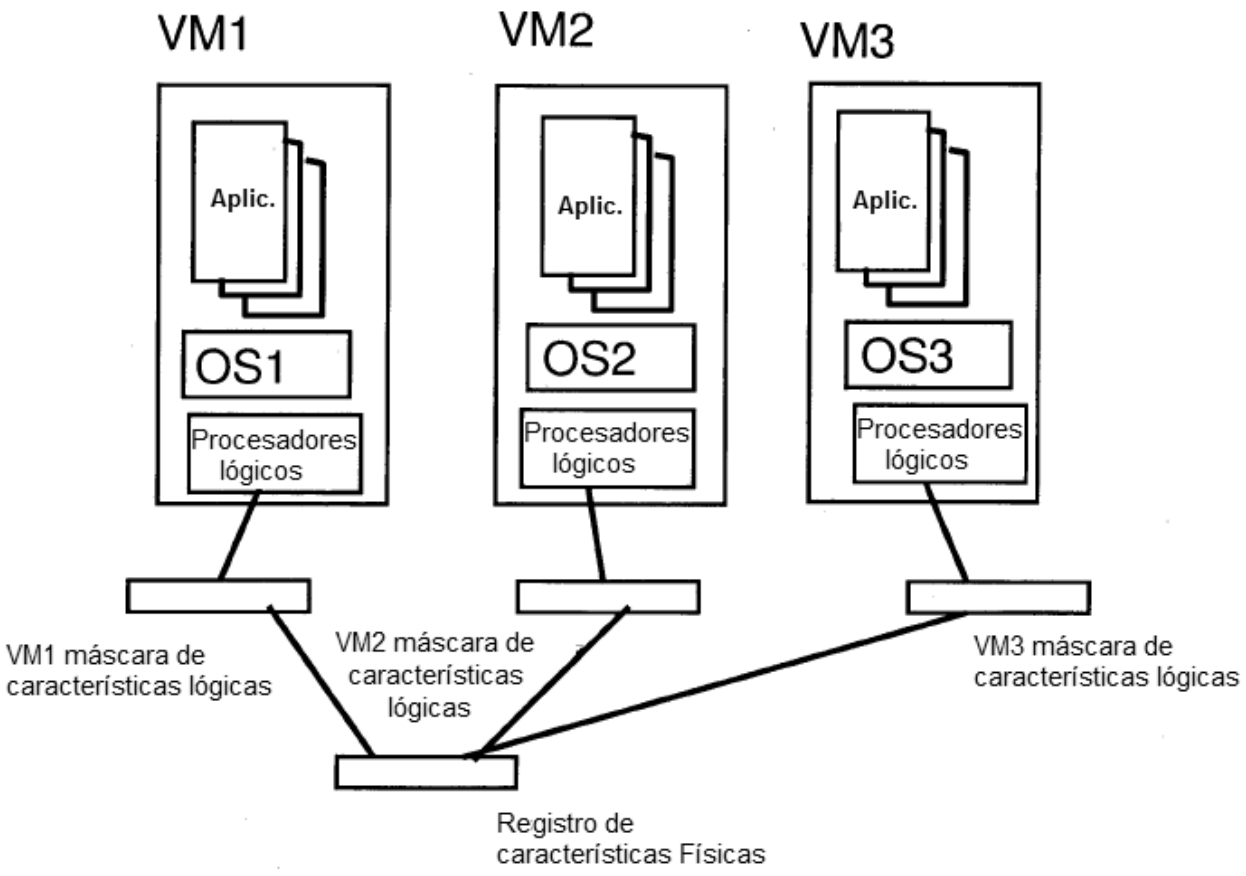

VM1
VM2
VM3
Aplic.
Aplic.
Aplic.
OS1
OS2
OS3
Procesadores lógicos
Procesadores lógicos
Procesadores lógicos
VM1 máscara de características lógicas
VM2 máscara de características lógicas
VM3 máscara de características lógicas
Registro de características Físicas

FIG. 7

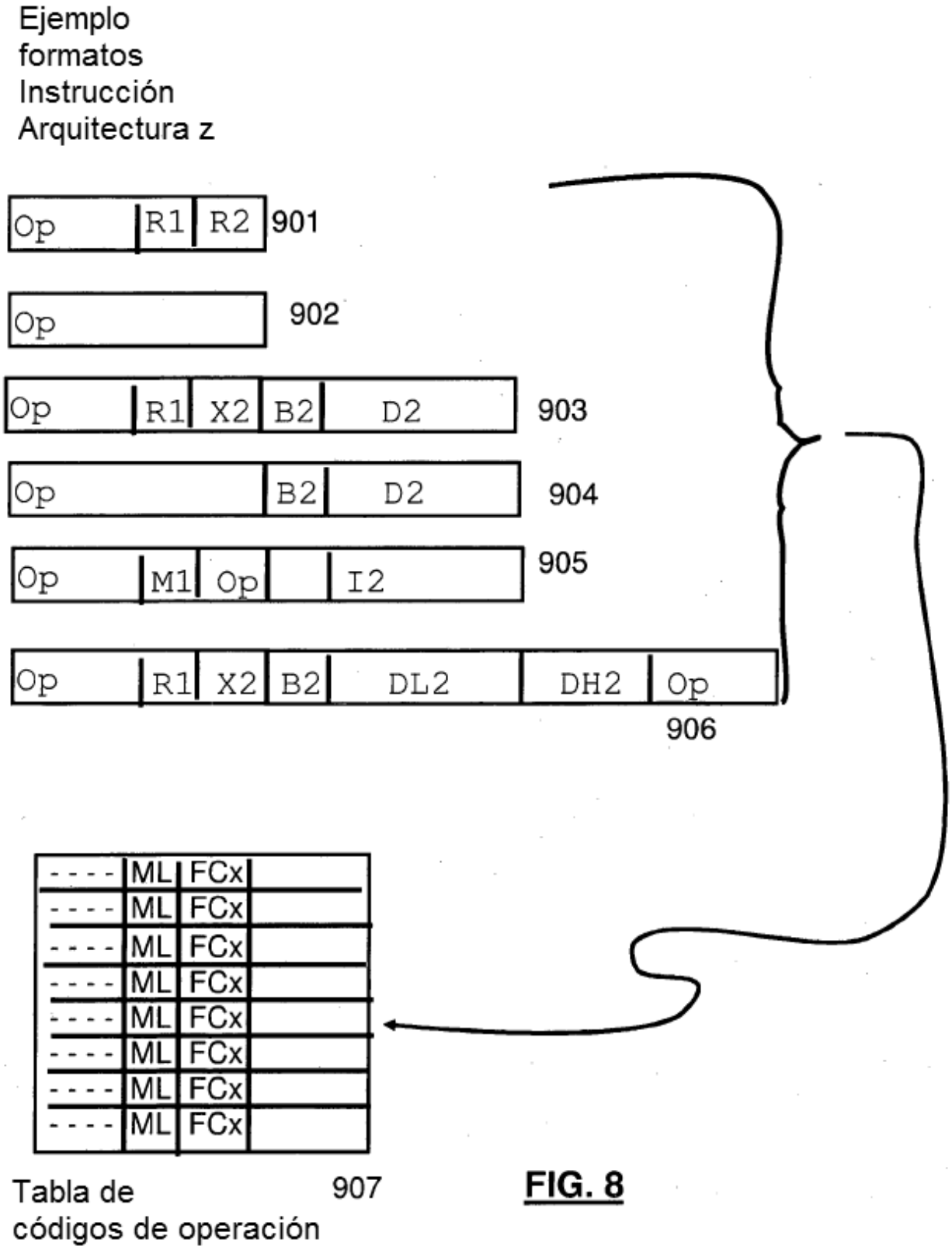
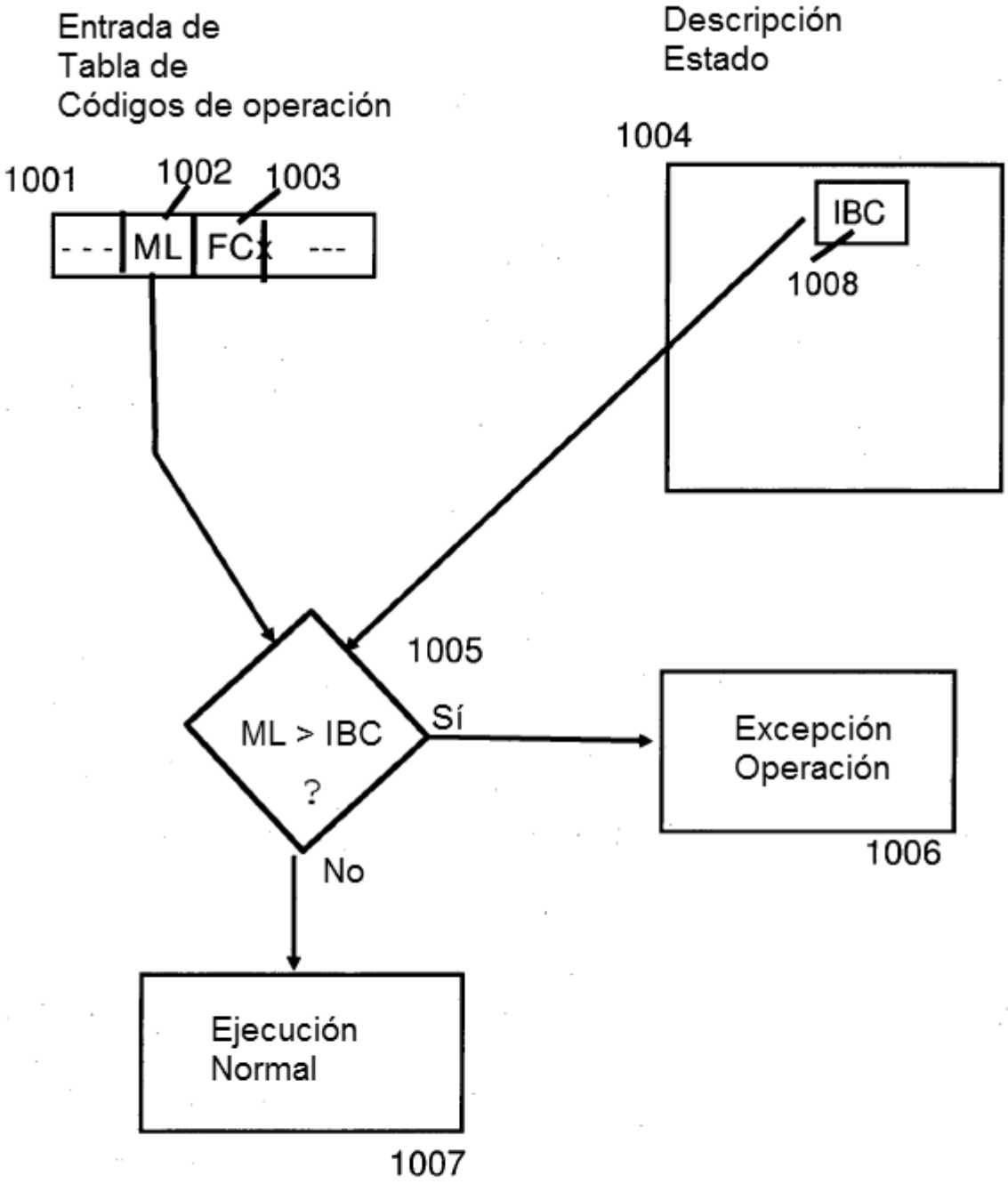
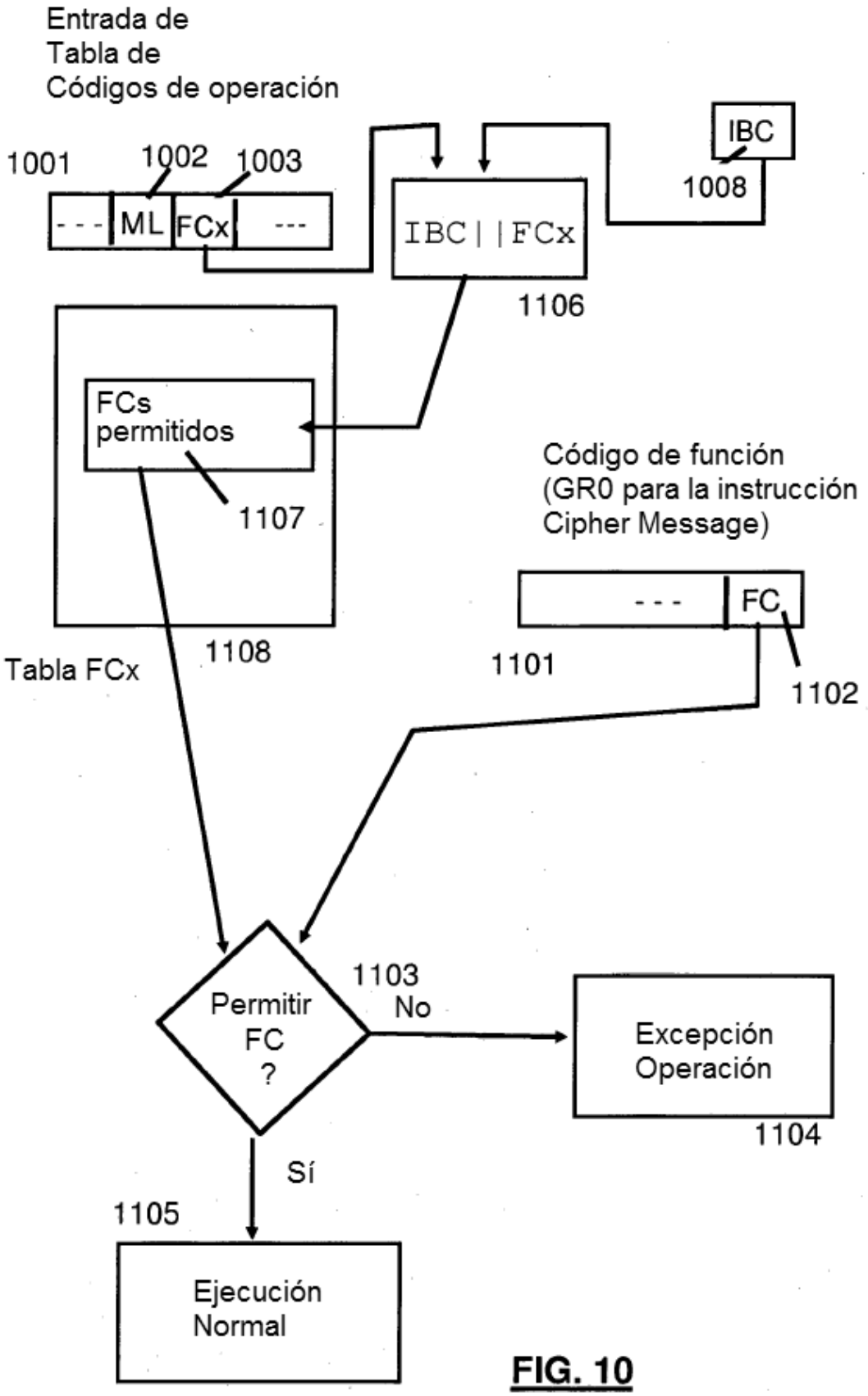

Ejemplo
formatos
Instrucción
Arquitectura z
Op
R1
R2
901
Op
902
Op
R1
X2
B2
D2
903
Op
B2
D2
904
Op
M1
Op
I2
905
Op
R1
X2
B2
DL2
DH2
Op
906
- - - -
ML
FCx
907
Tabla de
códigos de operación

FIG. 8


Entrada de
Tabla de
Códigos de operación
1001
1002
1003
- - -
ML
FCx

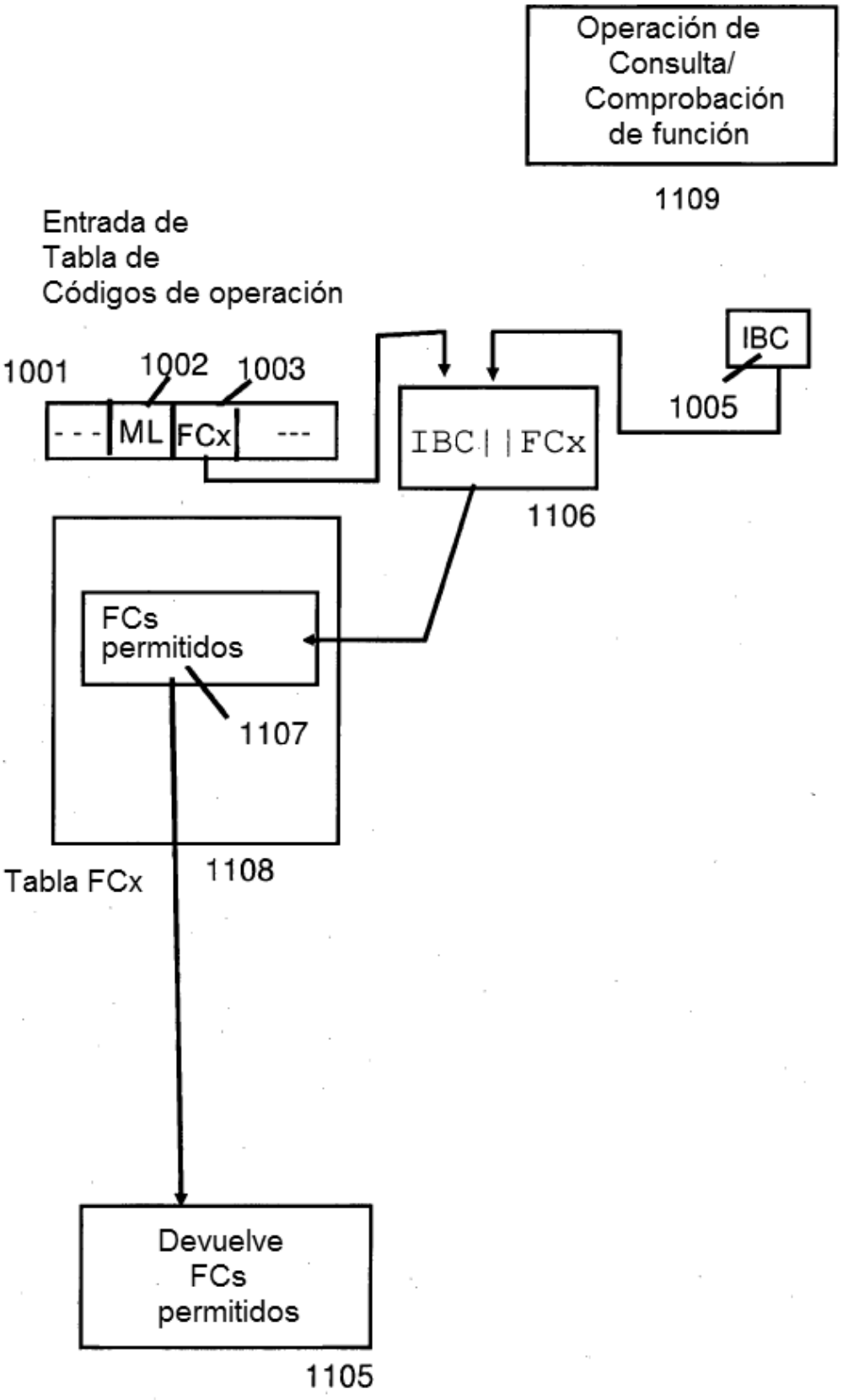
Descripción
Estado
1004
IBC
1008
1005
ML > IBC
?
Sí
Excepción
Operación
1006
No
Ejecución
Normal
1007

FIG. 9


Entrada de
Tabla de
Códigos de operación
1001
1002
1003
- - -
ML
FCx

IBC
1008
IBC||FCx
1106
FCs
permitidos
1107
1108
Tabla FCx
Código de función
(GR0 para la instrucción
Cipher Message)
- - -
FC
1101
1102
Permitir
FC
?
1103
No
Excepción
Operación
1104
Sí
1105
Ejecución
Normal

FIG. 10


Operación de Consulta/ Comprobación de función
1109
Entrada de Tabla de Códigos de operación
1001
1002
1003
- - -
ML
FCx

IBC
1005
IBC||FCx
1106
FCs permitidos
1107
1108
Tabla FCx
Devuelve FCs permitidos
1105

FIG. 11

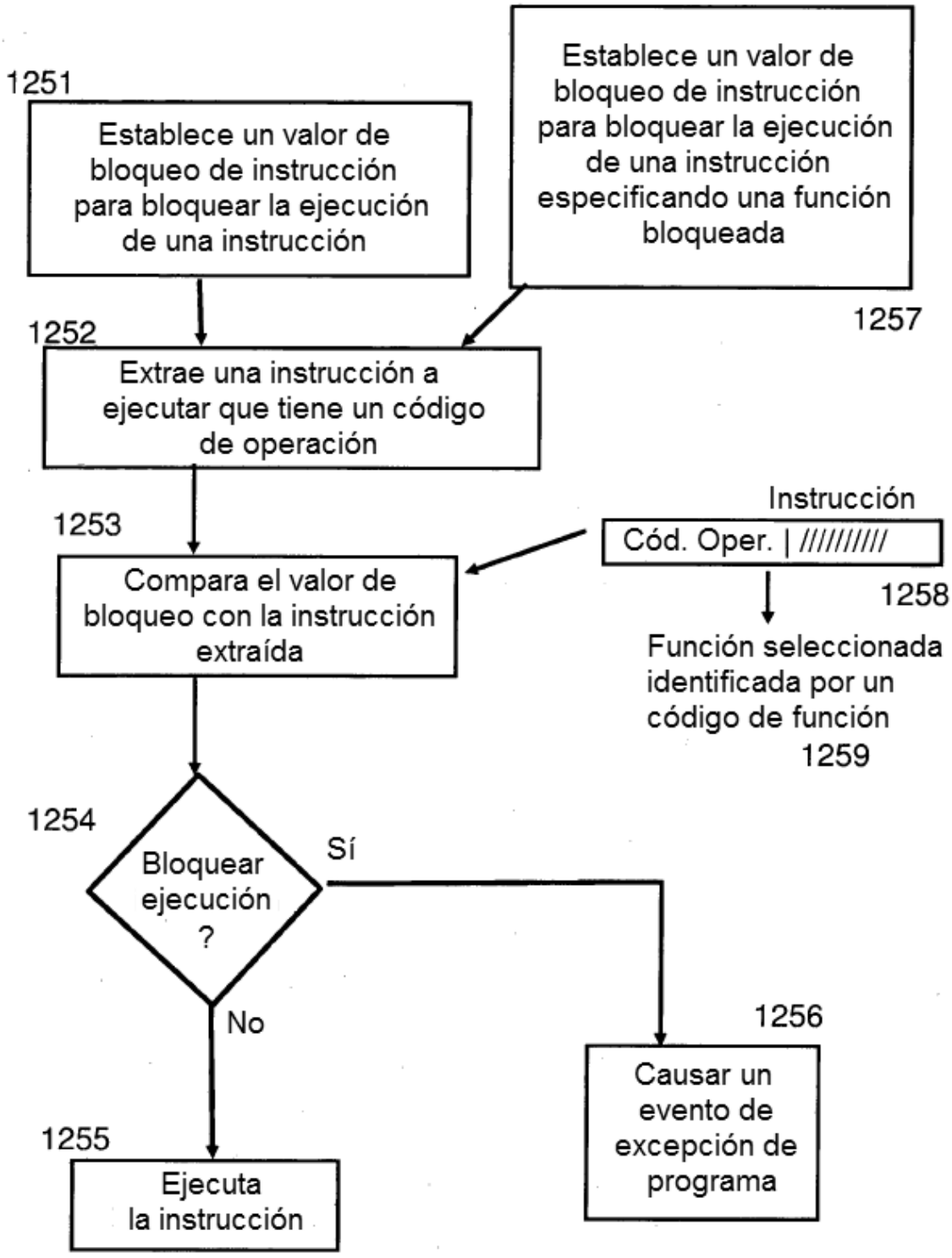

1251
Establece un valor de bloqueo de instrucción para bloquear la ejecución de una instrucción
Establece un valor de bloqueo de instrucción para bloquear la ejecución de una instrucción especificando una función bloqueada
1257
1252
Extrae una instrucción a ejecutar que tiene un código de operación
Instrucción
Cód. Oper. | //////////
1258
1253
Compara el valor de bloqueo con la instrucción extraída
Función seleccionada identificada por un código de función
1259
1254
Bloquear ejecución ?
Sí
No
1256
Causar un evento de excepción de programa
1255
Ejecuta la instrucción

FIG. 12

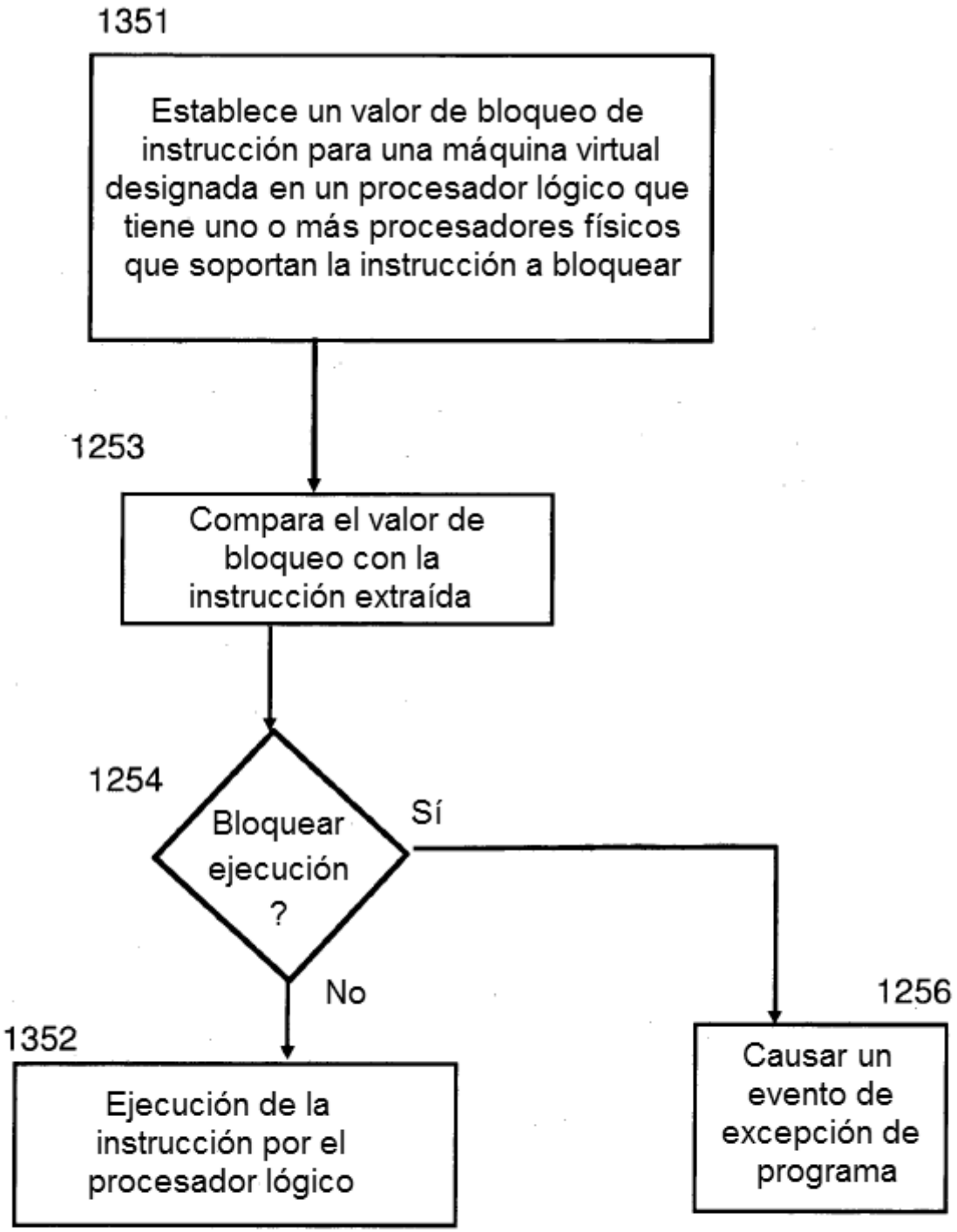

1351
Establece un valor de bloqueo de instrucción para una máquina virtual designada en un procesador lógico que tiene uno o más procesadores físicos que soportan la instrucción a bloquear
1253
Compara el valor de bloqueo con la instrucción extraída
1254
Bloquear ejecución ?
Sí
No
1256
Causar un evento de excepción de programa
1352
Ejecución de la instrucción por el procesador lógico

FIG. 13

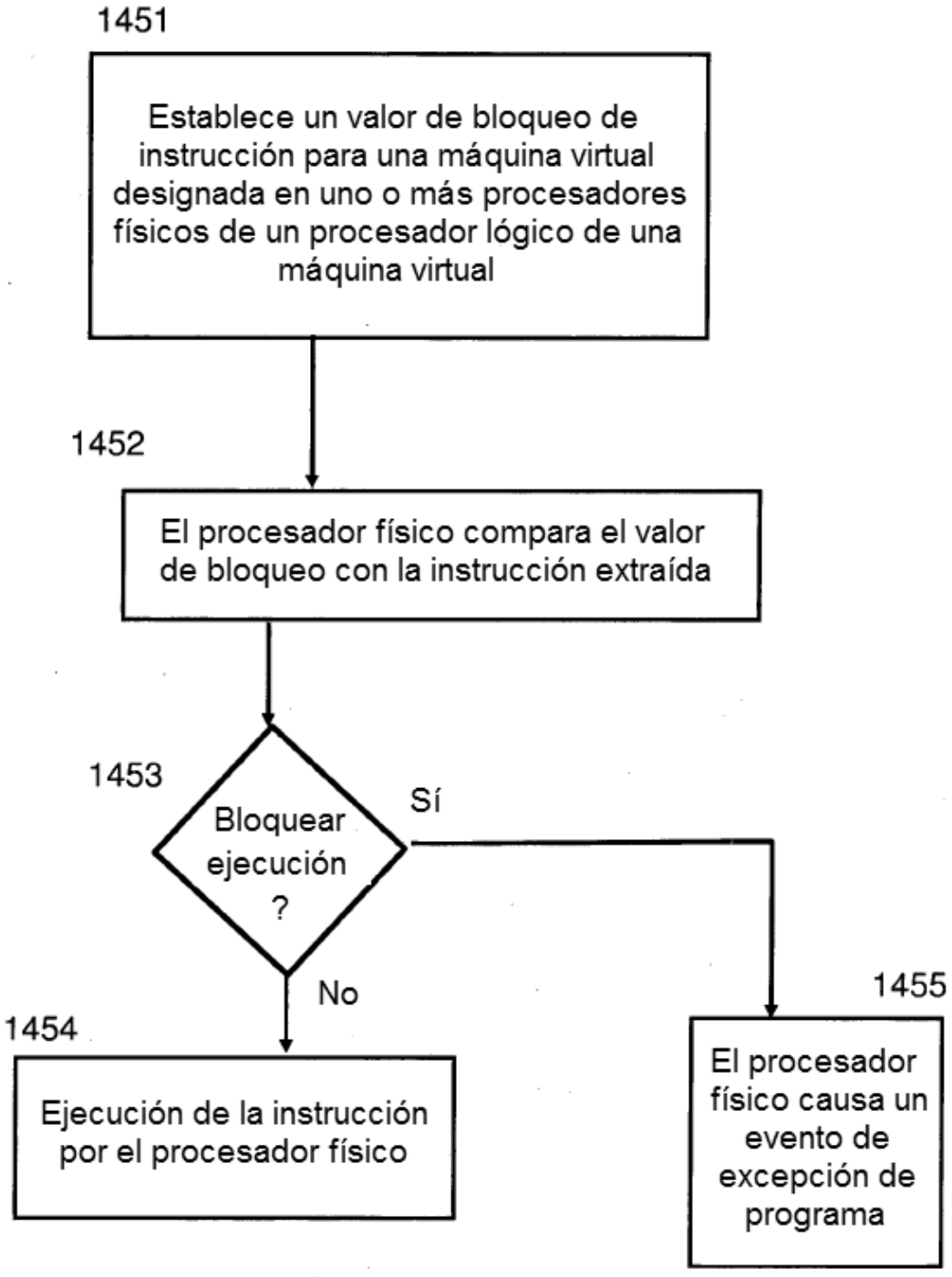

1451
Establece un valor de bloqueo de instrucción para una máquina virtual designada en uno o más procesadores físicos de un procesador lógico de una máquina virtual
1452
El procesador físico compara el valor de bloqueo con la instrucción extraída
1453
Bloquear ejecución ?
Sí
No
1454
Ejecución de la instrucción por el procesador físico
1455
El procesador físico causa un evento de excepción de programa

FIG. 14

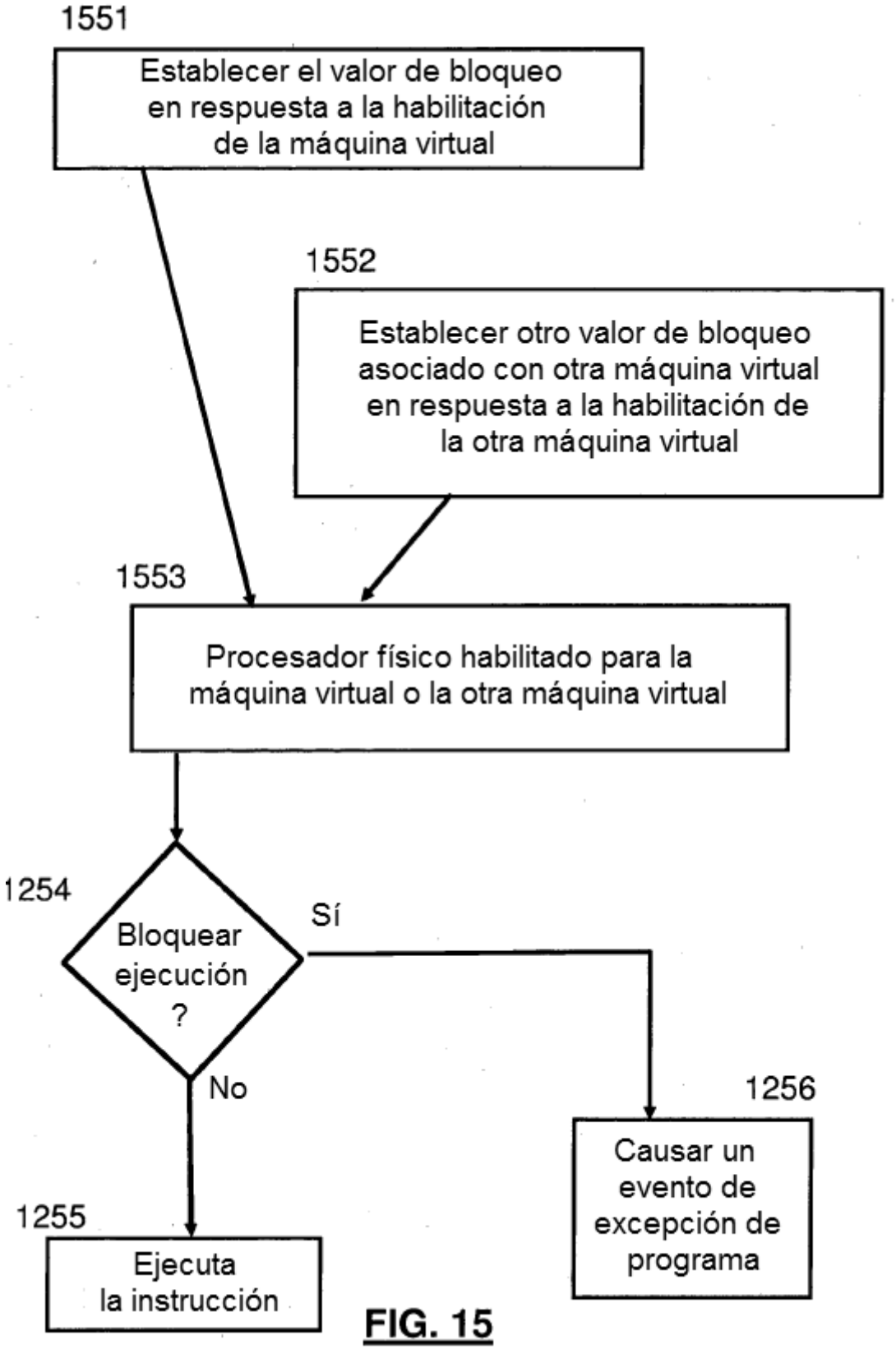

1551
Establecer el valor de bloqueo en respuesta a la habilitación de la máquina virtual
1552
Establecer otro valor de bloqueo asociado con otra máquina virtual en respuesta a la habilitación de la otra máquina virtual
1553
Procesador físico habilitado para la máquina virtual o la otra máquina virtual
1254
Bloquear ejecución ?
Sí
No
1256
Causar un evento de excepción de programa
1255
Ejecuta la instrucción

FIG. 15