(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 719 685**

(51) Int. CI.:

G06F 9/54 (2006.01)
G06F 9/30 (2008.01)
G06F 9/38 (2008.01)
G06F 9/46 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **26.11.2012 PCT/IB2012/056732**

(87) Fecha y número de publicación internacional: **19.12.2013 WO13186602**

(96) Fecha de presentación y número de la solicitud europea: **26.11.2012 E 12878862 (7)**

(97) Fecha y número de publicación de la concesión europea: **13.03.2019 EP 2862057**

(54) Título: **Filtrado de interrupción de programa en ejecución transaccional**

(30) Prioridad:

15.06.2012 US 201213524839

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
12.07.2019

(73) Titular/es:

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

(72) Inventor/es:

GREINER, DAN;
JACOBI, CHRISTIAN;
SLEGEL, TIMOTHY y
MITRAN, MARCEL

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Filtrado de interrupción de programa en ejecución transaccional

Antecedentes

Uno o más aspectos se refieren, en general, al multiprocesamiento de entornos informáticos y, en particular, al procesamiento transaccional dentro de dichos entornos informáticos.

Un desafío perdurable en la programación de un multiprocesador es el de las actualizaciones a la misma ubicación de almacenamiento por múltiples unidades centrales de procesamiento (CPU, por sus siglas en inglés). Muchas instrucciones que actualizan ubicaciones de almacenamiento, incluidas incluso simples operaciones lógicas como, por ejemplo, AND, lo hacen con múltiples accesos a la ubicación. Por ejemplo, primero, la ubicación de almacenamiento se captura y, luego, el resultado actualizado se almacena otra vez.

Con el fin de que múltiples CPU actualicen de manera segura la misma ubicación de almacenamiento, el acceso a la ubicación se serializa. Una instrucción, la instrucción TEST AND SET, introducida con la arquitectura S/360 anteriormente ofrecida por International Business Machines Corporation, brindaba una actualización de enclavamiento de una ubicación de almacenamiento. Actualización de enclavamiento significa que, según lo observado por otras CPU y el subsistema de entrada/salida (E/S) (p.ej., subsistema de canal), todo el acceso de almacenamiento de la instrucción parece ocurrir de forma atómica. Más tarde, la arquitectura S/370 ofrecida por International Business Machines Corporation introdujo las instrucciones COMPARE AND SWAP y COMPARE DOUBLE AND SWAP que proveen un medio más sofisticado para llevar a cabo la actualización de enclavamiento, y permiten la implementación de lo que comúnmente se conoce como una palabra de bloqueo (o semáforo). Instrucciones recientemente añadidas han proporcionado capacidades de actualización de enclavamiento adicionales, incluidas COMPARE AND SWAP AND PURGE, y COMPARE AND SWAP AND STORE. Sin embargo, todas dichas instrucciones proveen enclavamiento solamente para una única ubicación de almacenamiento.

Técnicas de programa más complejas pueden requerir la actualización de enclavamiento de múltiples ubicaciones de almacenamiento como, por ejemplo, cuando se añade un elemento a una lista doblemente enlazada. En dicha operación, tanto un puntero hacia adelante como uno hacia atrás parecen actualizarse de forma simultánea, según lo observado por otras CPU y el subsistema E/S. Con el fin de llevar a cabo dicha actualización de múltiples ubicaciones, el programa se ve forzado a usar un solo punto separado de serialización como, por ejemplo, una palabra de bloqueo. Sin embargo, las palabras de bloqueo pueden proveer un nivel de serialización mucho más grueso que el garantizado; por ejemplo, las palabras de bloqueo pueden serializar toda una cola de millones de elementos, incluso si solo dos elementos se están actualizando. El programa puede estructurar los datos para usar la serialización de grano más fino (p.ej., una jerarquía de puntos de bloqueo), pero que introduce problemas adicionales como, por ejemplo, potenciales situaciones de atasco si la jerarquía se viola, y cuestiones de recuperación si el programa encuentra un error mientras está sosteniendo uno o más bloqueos o si el bloqueo no puede adquirirse.

Además de lo descrito más arriba, hay numerosos escenarios donde un programa puede ejecutar una secuencia de instrucciones que pueden o pueden no resultar en una condición de excepción. Si no ocurre condición de excepción alguna, entonces el programa continúa; sin embargo, si se reconoce una excepción, entonces el programa puede tomar una acción correctiva para eliminar la condición de excepción. Java, como un ejemplo, puede explotar dicha ejecución en, por ejemplo, la ejecución especulativa, la expansión en línea parcial de una función y/o en la resecuencia de comprobación nula de puntero.

En entornos de sistemas operativos clásicos como, por ejemplo, z/OS y sus predecesores ofrecidos por International Business Machines Corporation, el programa establece un entorno de recuperación para interceptar cualquier condición de excepción de programa que pueda encontrar. Si el programa no intercepta la excepción, el sistema operativo normalmente termina, de forma anormal, el programa para las excepciones que el sistema operativo no está preparado para manejar. El establecimiento y la explotación de dicho entorno son costosos y complicados.

El documento US 2012/0084477 A1 describe un mecanismo de prioridad de memoria transaccional. El documento *Intel Architecture Instruction Set Extensions Programming Reference*, 1 febrero 2012 (01-02-2012), describe una arquitectura donde las excepciones en una transacción provocarán un aborto.

El documento US 2011/0296148 A1 describe un sistema de memoria transaccional que soporta una ejecución suspendida sin interrupciones.

Breve compendio

La invención provee un método reivindicado en la reivindicación 1 y un sistema y programa de ordenador correspondientes.

Breve descripción de las varias vistas de los dibujos

Uno o más aspectos se señalan en particular y se reivindican de manera distinta como ejemplos en las reivindicaciones en la conclusión de la memoria descriptiva. Lo anterior y otros objetos, características y ventajas son aparentes a partir de la siguiente descripción detallada tomada en conjunto con los dibujos anexos, en los cuales:

La Figura 1 representa una realización de un entorno informático;

la Figura 2A representa un ejemplo de una instrucción Transaction Begin (TBEGIN);

la Figura 2B representa una realización de detalles adicionales de un campo de la instrucción TBEGIN de la Figura 2A;

la Figura 3A representa un ejemplo de una instrucción Transaction Begin restringida (TBEGINC);

la Figura 3B representa una realización de detalles adicionales de un campo de la instrucción TBEGINC de la Figura 3A;

la Figura 4 representa un ejemplo de una instrucción Transaction End (TEND);

la Figura 5 representa un ejemplo de una instrucción Transaction Abort (TABORT);

la Figura 6 representa un ejemplo de transacciones anidadas;

la Figura 7 representa un ejemplo de una instrucción NONTRANSACTIONAL STORE (NTSTG);

la Figura 8 representa un ejemplo de una instrucción EXTRACT TRANSACTION NESTING DEPTH (ETND);

la Figura 9 representa un ejemplo de un bloque de diagnóstico de transacción;

la Figura 10 representa motivos a modo de ejemplo para el aborto, junto con códigos de aborto y códigos de condición asociados;

la Figura 11 representa una realización de la lógica asociada a la ejecución de una instrucción TBEGINC;

la Figura 12 representa una realización de la lógica asociada a la ejecución de una instrucción TBEGIN;

la Figura 13 representa una realización de la lógica asociada a la ejecución de una instrucción TEND;

la Figura 14 representa una realización de la lógica asociada al procesamiento de aborto de transacción;

la Figura 15 representa un ejemplo de niveles de filtrado de interrupción de programa;

las Figuras 16A-16B representan un ejemplo de condiciones de excepción de programa y clases de transacción y códigos de condición asociados;

las Figuras 17A-17B representan un ejemplo de la lógica asociada al filtrado de interrupciones;

las Figuras 18A-18B representan un ejemplo de inserción de un elemento de cola en una lista doblemente enlazada de elementos de cola;

la Figura 19 representa una realización de un producto de programa de ordenador;

la Figura 20 representa una realización de un sistema informático anfitrión;

la Figura 21 representa un ejemplo adicional de un sistema informático;

la Figura 22 representa otro ejemplo de un sistema informático que comprende una red de ordenadores;

la Figura 23 representa una realización de varios elementos de un sistema informático;

la Figura 24A representa una realización de la unidad de ejecución del sistema informático de la Figura 23;

la Figura 24B representa una realización de la unidad de derivación del sistema informático de la Figura 23;

la Figura 24C representa una realización de la unidad de carga/almacenamiento del sistema informático de la Figura 23; y

la Figura 25 representa una realización de un sistema informático anfitrión emulado.

Descripción detallada

Según un aspecto, se provee una facilidad de ejecución transaccional (TX). Dicha facilidad provee procesamiento transaccional para instrucciones y, en una o más realizaciones, ofrece diferentes modos de ejecución, según se describe más abajo, así como niveles anidados de procesamiento transaccional.

La facilidad de ejecución transaccional introduce un estado de CPU llamado el modo de ejecución transaccional (TX). Siguiendo una reiniciación de la CPU, la CPU no se encuentra en el modo TX. La CPU entra en el modo TX por una instrucción TRANSACTION BEGIN. La CPU abandona el modo TX por (a) una instrucción TRANSACTION END más externa (más detalles sobre interior y exterior se describirán), o (b) la transacción que se está abortando. Mientras que en el modo TX, los accesos al almacenamiento por la CPU parecen ser concurrentes con los bloques según lo observado por otras CPU y el subsistema E/S. Los accesos al almacenamiento (a) están comprometidos al almacenamiento cuando la transacción más externa finaliza sin abortar (a saber, p.ej., las actualizaciones llevadas a cabo en caché o memoria intermedia local para la CPU se propagan y almacenan en memoria real y son visibles para otras CPU, o (b) se descartan si la transacción se aborta.

Las transacciones pueden anidarse. Es decir, mientras la CPU está en el modo TX, puede ejecutar otra instrucción TRANSACTION BEGIN. La instrucción que hace que la CPU entre en el modo TX se llama TRANSACTION BEGIN más externa; de manera similar, se dice que el programa está en la transacción más externa. Las ejecuciones posteriores de TRANSACTION BEGIN se llaman instrucciones internas; y el programa está ejecutando una transacción interna. El modelo provee una profundidad de jerarquización mínima y una profundidad de jerarquización máxima dependiente del modelo. Una instrucción EXTRACT TRANSACTION NESTING DEPTH devuelve el valor de profundidad de jerarquización actual y, en una realización adicional, puede devolver un valor de profundidad de jerarquización máximo. La presente técnica usa un modelo llamado "jerarquización plana" en el cual una condición de aborto en cualquier profundidad de jerarquización hace que todos los niveles de la transacción se aborten, y el control se devuelve a la instrucción que sigue a TRANSACTION BEGIN más externa.

Durante el procesamiento de una transacción, se dice que un acceso transaccional llevado a cabo por una CPU está en conflicto con (a) un acceso transaccional o acceso no transaccional llevado a cabo por otra CPU, o (b) un acceso no transaccional llevado a cabo por el subsistema E/S, si ambos accesos son a cualquier ubicación dentro de la misma línea de caché, y uno o ambos accesos es/son un almacenamiento. En otras palabras, con el fin de que la ejecución transaccional sea productiva, la CPU no se observará realizando accesos transaccionales hasta que se comprometa. El presente modelo de programación puede ser altamente eficaz en ciertos entornos; por ejemplo, la actualización de dos puntos en una lista doblemente enlazada de un millón de elementos. Sin embargo, puede ser menos eficaz, si hay mucha contienda para las ubicaciones de almacenamiento a las que se está accediendo de manera transaccional.

En un modelo de ejecución transaccional (a la que, en la presente memoria, se hace referencia como una transacción no restringida), cuando una transacción se aborta, el programa puede intentar reconducir la transacción con la esperanza de que la condición de aborto ya no esté presente, o el programa puede "replegarse" a un trayecto no transaccional equivalente. En otro modelo de ejecución transaccional (al que, en la presente memoria, se hace referencia como una transacción restringida), una transacción abortada se reconduce automáticamente por la CPU; en ausencia de violaciones de restricción, se asegura la finalización eventual de la transacción restringida.

Cuando se inicia una transacción, el programa puede especificar varios controles como, por ejemplo, (a) qué registros generales se restablecen a sus contenidos originales si la transacción se aborta, (b) si se permite que la transacción modifique el contexto de registro de punto flotante, incluidos, por ejemplo, registros de punto flotante y el registro de control de punto flotante, (c) si se permite que la transacción modifique registros de acceso (AR), y (d) si se evitará que ciertas condiciones de excepción provoquen una interrupción. Si una transacción no restringida se aborta, puede proveerse información de diagnóstico. Por ejemplo, la instrucción TBEGIN más externa que inicia una transacción no restringida puede designar un bloque de diagnóstico de transacción (TDB, por sus siglas en inglés) especificado de programa. Además, el TDB en el área de prefijo de la CPU o designado por la descripción de estado del anfitrión puede también usarse si la transacción se aborta debido a una interrupción de programa o a una condición que hace que la ejecución interpretativa finalice, respectivamente.

Más arriba se indican varios tipos de registros. Estos se explican en mayor detalle en la presente memoria. Los registros generales pueden usarse como acumuladores en operaciones aritméticas y lógicas generales. En una realización, cada registro contiene 64 posiciones de bits, y hay 16 registros generales. Los registros generales se identifican por los números 0-15, y se designan por un campo R de cuatro bits en una instrucción. Algunas instrucciones proveen la dirección de múltiples registros generales al tener varios campos R. Para algunas instrucciones, el uso de un registro general específico es implícito antes que designado explícitamente por un campo R de la instrucción.

Además de su uso como acumuladores en operaciones aritméticas y lógicas generales, 15 de los 16 registros generales también se usan como registros de dirección e índice de base en la generación de direcciones. En dichos casos, los registros se designan por un campo B de cuatro bits o campo X en una instrucción. Un valor de cero en el campo B o X especifica que no se aplicará una base o índice y, por consiguiente, el registro general 0 no se designará como uno que contiene una dirección o índice de base.

Las instrucciones de punto flotante usan un conjunto de registros de punto flotante. La CPU tiene 16 registros de punto flotante, en una realización. Los registros de punto flotante se identifican por los números 0-15 y se designan por un campo R de cuatro bits en instrucciones de punto flotante. Cada registro de punto flotante es de 64 bits de largo y puede contener un operando de punto flotante corto (de 32 bits) o largo (de 64 bits).

Un registro de control de punto flotante (FPC, por sus siglas en inglés) es un registro de 32 bits que contiene bits de máscara, bits de bandera, un código de excepción de datos, y bits de modo de redondeo, y se usa durante el procesamiento de operaciones de punto flotante.

Además, en una realización, la CPU tiene 16 registros de control, cada uno con 64 posiciones de bits. Las posiciones de bits en los registros se asignan a facilidades particulares en el sistema como, por ejemplo, Registro de Episodio de Programa (PER, por sus siglas en inglés) (descrito más abajo), y se usan para especificar que una operación puede tener lugar o para proveer información especial requerida por la facilidad. En una realización, para la facilidad transaccional, se usan CR0 (bits 8 y 9) y CR2 (bits 61-63), según se describe más abajo.

La CPU tiene, por ejemplo, 16 registros de acceso numerados 0-15. Un registro de acceso consiste en 32 posiciones de bits que contienen una especificación indirecta de un elemento de control de espacio de dirección (ASCE, por sus siglas en inglés). Un elemento de control de espacio de dirección es un parámetro usado por el mecanismo de traducción dinámica de direcciones (DAT, por sus siglas en inglés) para traducir referencia en un espacio de dirección correspondiente. Cuando la CPU está en un modo llamado el modo de registro de acceso (controlado por bits en la palabra de estado de programa (PSW, por sus siglas en inglés)), un campo B de instrucción, usado para especificar una dirección lógica para una referencia de operando de almacenamiento, designa un registro de acceso, y el elemento de control de espacio de dirección especificado por el registro de acceso se usa por DAT para la referencia que se está llevando a cabo. Para algunas instrucciones, un campo R se usa en lugar de un campo B. Se proveen instrucciones para cargar y almacenar los contenidos de los registros de acceso y para mover los contenidos de un registro de acceso a otro.

Cada uno de los registros de acceso 1-15 puede designar cualquier espacio de dirección. El registro de acceso 0 designa el espacio de instrucción primario. Cuando uno de los registros de acceso 1-15 se usa para designar un espacio de dirección, la CPU determina qué espacio de dirección se designa por la traducción de los contenidos del registro de acceso. Cuando el registro de acceso 0 se usa para designar un espacio de dirección, la CPU trata el registro de acceso como uno que designa el espacio de instrucción primario, y no examina los contenidos reales del registro de acceso. Por lo tanto, los 16 registros de acceso pueden designar, en cualquier momento, el espacio de instrucción primario y un máximo de 15 espacios diferentes.

En una realización, hay múltiples tipos de espacios de dirección. Un espacio de dirección es una secuencia consecutiva de números enteros (direcciones virtuales), junto con los parámetros de transformación específicos que permiten que cada número se asocie a una ubicación de byte en el almacenamiento. La secuencia comienza en cero y procede de izquierda a derecha.

En, por ejemplo, z/Architecture, cuando una dirección virtual se usa por una CPU para acceder al almacenamiento principal (también conocido como memoria principal), primer se convierte, por medio de la traducción dinámica de direcciones (DAT), en una dirección real, y luego, por medio de la prefijación, en una dirección absoluta. DAT puede usar de uno a cinco niveles de tablas (página, segmento, región tercera, región segunda, y región primera) como parámetros de transformación. La designación (origen y longitud) de la tabla de nivel más alto para un espacio de dirección específico se llama elemento de control de espacio de dirección, y se descubre para el uso por DAT en un registro de control o según se especifica por un registro de acceso. De manera alternativa, el elemento de control de espacio de dirección para un espacio de dirección puede ser una designación de espacio real, que indica que DAT traducirá la dirección virtual simplemente tratándola como una dirección real y sin usar tablas.

DAT usa, en diferentes momentos, los elementos de control de espacio de dirección en diferentes registros de control o según lo especificado por los registros de acceso. La elección se determina por el modo de traducción especificado en la PSW actual. Cuatro modos de traducción están disponibles: modo de espacio primario, modo de espacio secundario, modo de registro de acceso y modo de espacio doméstico. Diferentes espacios de dirección son direccionables dependiendo del modo de traducción.

En cualquier instante cuando la CPU está en el modo de espacio primario o modo de espacio secundario, la CPU puede traducir direcciones virtuales que pertenecen a dos espacios de dirección - el espacio de dirección primario y el segundo espacio de dirección. En cualquier instante cuando la CPU está en el modo de registro de acceso, puede traducir direcciones virtuales de hasta 16 espacios de dirección - el espacio de dirección primario y hasta 15 espacios de dirección de AR especificados. En cualquier instante cuando la CPU está en el modo de espacio doméstico, puede traducir direcciones virtuales del espacio de dirección doméstico.

El espacio de dirección primario se identifica como tal porque consiste en direcciones virtuales primarias, que se traducen por medio del elemento de control de espacio de dirección (ASCE) primario. De manera similar, el espacio de dirección secundario consiste en direcciones virtuales secundarias traducidas por medio del ASCE secundario; los espacios de dirección de AR especificados consisten en direcciones virtuales de AR especificados traducidas por medio de ASCE de AR especificados; y el espacio de dirección doméstico de direcciones virtuales domésticas traducidas por medio del ASCE doméstico. Los ASCE primario y secundario están en los registros de control 1 y 7, respectivamente. Los ASCE de AR especificados están en entradas segunda tabla de ASN que se ubican a través de un proceso llamado traducción de registro de acceso (ART, por sus siglas en inglés) mediante el uso de los registros de control 2, 5 y 8. El ASCE doméstico está en el registro de control 13.

Una realización de un entorno informático para incorporar y usar uno o más aspectos de la facilidad transaccional descrita en la presente memoria se describe con referencia a la Figura 1.

Con referencia a la Figura 1, en un ejemplo, el entorno informático 100 se basa en z/Architecture, ofrecida por International Business Machines (IBM®) Corporation, Armonk, Nueva York. z/Architecture se describe en una Publicación de IBM titulada "*z/Architecture - Principies of Operation*", Publicación No. SA22-7932-08, 9ª. Edición, agosto 2010.

Z/ARCHITECTURE, IBM, y Z/OS y Z/VM (a los que se hace referencia más abajo) son marcas comerciales registradas de International Business Machines Corporation, Armonk, Nueva York. Otros nombres usados en la presente memoria pueden ser marcas comerciales registradas, marcas comerciales o nombres de productos de International Business Machines Corporation u otras compañías.

A modo de ejemplo, el entorno informático 100 incluye un complejo de procesador central (CPC, por sus siglas en inglés) 102 acoplado a uno o más dispositivos de entrada/salida (E/S) 106 mediante una o más unidades de control 108. El complejo de procesador central 102 incluye, por ejemplo, uno o más procesadores centrales 110, una o más particiones 112 (p.ej., particiones lógicas (LP)), un hipervisor de particiones lógicas 114, y un subsistema de entrada/salida 115, cada uno de los cuales se describe más abajo.

Los procesadores centrales 110 son recursos de procesadores físicos asignados a las particiones lógicas. En particular, cada partición lógica 112 tiene uno o más procesadores lógicos, cada uno de los cuales representa todos o una parte de un procesador físico 110 asignado a la partición. Los procesadores lógicos de una partición 112 particular pueden dedicarse a la partición, de modo que el recurso de procesador 110 subyacente se reserva para dicha partición; o se comparten con otra partición, de modo que el recurso de procesador subyacente está potencialmente disponible para otra partición.

Una partición lógica funciona como un sistema separado y tiene una o más aplicaciones y, de manera opcional, un sistema operativo residente allí, que puede diferir para cada partición lógica. En un ejemplo, el sistema operativo es el sistema operativo z/OS, el sistema operativo z/VM, el sistema operativo z/Linux, o el sistema operativo TPF, ofrecidos por International Business Machines Corporation, Armonk, Nueva York. Las particiones lógicas 112 se gestionan por un hipervisor de particiones lógicas 114, que se implementa por firmware que se ejecuta en los procesadores 110. Según su uso en la presente memoria, firmware incluye, p.ej., el microcódigo y/o milicódigo del procesador. Este incluye, por ejemplo, las instrucciones a nivel de hardware y/o estructuras de datos usadas en la implementación del código de máquina de nivel más alto. En un ejemplo, incluye, por ejemplo, código patentado que se entrega, normalmente, como microcódigo que incluye software fiable o microcódigo específico al hardware subyacente y controla el acceso del sistema operativo al hardware del sistema.

Las particiones lógicas y el hipervisor de particiones lógicas comprenden, cada uno, uno o más programas que residen en las respectivas particiones del almacenamiento central asociado a los procesadores centrales. Un ejemplo de hipervisor de particiones lógicas 114 es el Gestor de Sistemas/Recursos de Procesador (PR/SM, por sus siglas en inglés), ofrecido por International Business Machines Corporation, Armonk, Nueva York.

El subsistema de entrada/salida 115 dirige el flujo de información entre dispositivos de entrada/salida 106 y el almacenamiento principal (también conocido como memoria principal). Este se acopla al complejo de procesamiento central, en el sentido de que puede ser una parte del complejo de procesamiento central o separado de aquel. El subsistema E/S libra a los procesadores centrales de la tarea de comunicarse directamente con los dispositivos de entrada/salida y permite que el procesamiento de datos proceda de manera concurrente con el procesamiento de entrada/salida. Para proveer comunicaciones, el subsistema E/S emplea adaptadores de comunicaciones E/S. Existen varios tipos de adaptadores de comunicaciones que incluyen, por ejemplo, canales, adaptadores E/S, tarjetas PCI, tarjetas Ethernet, tarjetas de Interfaz de Almacenamiento de Ordenador Pequeño (SCSI, por sus siglas en inglés), etc. En el ejemplo particular descrito en la presente memoria, los adaptadores de comunicaciones E/S son canales y, por lo tanto, en la presente memoria se hace referencia al subsistema E/S como un subsistema de canales. Sin embargo, esto es solo un ejemplo. Otros tipos de subsistemas E/S pueden usarse.

El subsistema E/S usa uno o más trayectos de entrada/salida como enlaces de comunicación en el manejo del flujo de información a o de los dispositivos de entrada/salida 106. En el presente ejemplo particular, dichos trayectos se llaman trayectos de canales, dado que los adaptadores de comunicación son canales.

El entorno informático descrito más arriba es solo un ejemplo de un entorno informático que puede usarse. Otros entornos, incluidos, pero sin limitación a ello, entornos no particionados, otros entornos particionados, y/o entornos emulados, pueden usarse; las realizaciones no se encuentran limitadas a un entorno.

Según uno o más aspectos, la facilidad de ejecución transaccional es una mejora de CPU que provee los medios por los cuales la CPU puede ejecutar una secuencia de instrucciones - conocida como una transacción - que pueden acceder a múltiples ubicaciones de almacenamiento, incluida la actualización de dichas ubicaciones. Según se observa por otras CPU y el subsistema E/S, la transacción (a) se completa en su totalidad como una sola operación atómica, o (b) se aborta, y potencialmente no deja evidencia alguna de que se ha ejecutado alguna vez (a excepción de ciertas condiciones descritas en la presente memoria). Por consiguiente, una transacción completada con éxito

puede actualizar numerosas ubicaciones de almacenamiento sin bloqueo especial que se necesita en el modelo de multiprocesamiento clásico.

La facilidad de ejecución transaccional incluye, por ejemplo, uno o más controles; una o más instrucciones; procesamiento transaccional, incluida la ejecución restringida y no restringida; y procesamiento de aborto, cada uno de los cuales se describe en mayor detalle más abajo.

En una realización, tres controles de propósito especial, incluidos Palabra de Estado de Programa (PSW) de aborto de transacción, una dirección de bloque de diagnóstico de transacción (TDB), y una profundidad de jerarquización de transacción; cinco bits de registro de control; y seis instrucciones generales, incluidas TRANSACTION BEGIN (restringida y no restringida), TRANSACTION END, EXTRACT TRANSACTION NESTING DEPTH, TRANSACTION ABORT, y NONTRANSACTIONAL STORE, se usan para controlar la facilidad de ejecución transaccional. Cuando la facilidad se instala, se instala, por ejemplo, en todas las CPU en la configuración. Una indicación de facilidad, bit 73 en una implementación, cuando es uno, indica que la facilidad de ejecución transaccional está instalada.

Cuando la facilidad de ejecución transaccional se instala, la configuración provee una facilidad de ejecución transaccional no restringida y, de manera opcional, una facilidad de ejecución transaccional restringida, cada una de las cuales se describe más abajo. Cuando las indicaciones de facilidad 50 y 73, como ejemplos, son ambas uno, la facilidad de ejecución transaccional restringida se instala. Ambas indicaciones de facilidad se almacenan en la memoria en ubicaciones especificadas.

Según su uso en la presente memoria, el nombre de instrucción TRANSACTION BEGIN se refiere a las instrucciones que tienen la nemotecnia TBEGIN (Iniciar Transacción para una transacción no restringida) y TBEGINC (Iniciar Transacción para una transacción restringida). Las descripciones relativas a una instrucción específica se indican por el nombre de instrucción seguido de la nemotecnia en paréntesis o corchetes, o simplemente de la nemotecnia.

Una realización de un formato de una instrucción TRANSACTION BEGIN (TBEGIN) se representa en las Figuras 2A-2B. A modo de ejemplo, una instrucción TBEGIN 200 incluye un campo de código operacional 202 que incluye un código operacional que especifica una operación de iniciar transacción no restringida; un campo base (B_1) 204; un campo de desplazamiento (D_1) 206; y un campo inmediato (I_2) 208. Cuando el campo B_1 es diferente de cero, los contenidos del registro general especificado por B_1 204 se añaden a D_1 206 para obtener la primera dirección del operando.

Cuando el campo B_1 es diferente de cero, se aplica lo siguiente:

• Cuando la profundidad de jerarquización de transacción es inicialmente cero, la primera dirección del operando designa la ubicación del bloque de diagnóstico de transacción de 256 bytes, llamado el TDB de TBEGIN especificado (descrito en mayor detalle más abajo) en el cual la información de diagnóstico puede almacenarse si la transacción se aborta. Cuando la CPU está en el modo de espacio primario o modo de registro de acceso, la primera dirección de operando designa una ubicación en el espacio de dirección primario. Cuando la CPU está en el modo de espacio secundario o de espacio doméstico, la primera dirección de operando designa una ubicación en el espacio de dirección secundario o doméstico, respectivamente. Cuando DAT está apagada, la dirección del bloque de diagnóstico de transacción (TDB) (TDBA) designa una ubicación en el almacenamiento real.

La accesibilidad del almacenamiento al primer operando está determinada. Si es accesible, la dirección lógica del operando se coloca en la dirección de bloque de diagnóstico de transacción (TDBA), y la TDBA es válida.

• Cuando la CPU ya está en el modo de ejecución transaccional no restringido, la TDBA no se modifica, y es impredecible si el primer operando se verifica para la accesibilidad.

Cuando el campo B_1 es cero, no se detectan excepciones de acceso para el primer operando y, para la instrucción TBEGIN más externa, la TDBA es inválida.

Los bits del campo I_2 se definen de la siguiente manera, en un ejemplo:

Máscara de Guardado de Registro General (GRSM, por sus siglas en inglés) 210 (Figura 2B): Los bits 0-7 del campo I_2 contienen la máscara de guardado de registro general (GRSM). Cada bit de la GRSM representa un par par-impar de registros generales, donde el bit 0 representa los registros 0 y 1, el bit 1 representa los registros 2 y 3, y así sucesivamente. Cuando un bit en la GRSM de la instrucción TBEGIN más externa es cero, el par de registro correspondiente no se guarda. Cuando un bit en la GRSM de la instrucción TBEGIN más externa es uno, el par de registro correspondiente se guarda en una ubicación dependiente del modelo que no es directamente accesible por el programa.

Si la transacción se aborta, los pares de registros guardados se restablecen a sus contenidos cuando la instrucción TBEGIN más externa se ejecutó. Los contenidos de todos los otros registros generales (no guardados) no se restablecen cuando una transacción se aborta.

La máscara de guardado de registro general se ignora en todas las TBEGIN a excepción de la más externa.

Permitir Modificación AR (A) 212: El control A, el bit 12 del campo I_2, controla si se permite que la transacción modifique un registro de acceso. El control de permiso de modificación AR eficaz es la AND lógica del control A en la instrucción TBEGIN para el nivel de jerarquización actual y para todos los niveles externos.

Si el control A eficaz es cero, la transacción se abortará con el código de aborto 11 (instrucción restringida) si se realiza un intento de modificar cualquier registro de acceso. Si el control A eficaz es uno, la transacción no se abortará si un registro de acceso se modifica (en ausencia de cualquier otra condición de aborto).

Permitir la Operación de Punto Flotante (F) 214: El control F, el bit 13 del campo I_2, controla si se permite que la transacción ejecute instrucciones de punto flotante especificadas. El control de permiso de operación de punto flotante eficaz es la AND lógica del control F en la instrucción TBEGIN para el nivel de jerarquización actual y para todos los niveles externos.

Si el control F eficaz es cero, entonces (a) la transacción se abortará con el código de aborto 11 (instrucción restringida) si se realiza un intento de ejecutar una instrucción de punto flotante, y (b) el código de excepción de datos (DXC) en el byte 2 del registro de control de punto flotante (FPCR, por sus siglas en inglés) no se establecerá por cualquier condición de excepción de programa de excepción de datos. Si el control F eficaz es uno, entonces (a) la transacción no se abortará si se realiza un intento de ejecutar una instrucción de punto flotante (en ausencia de cualquier otra condición de aborto), y (b) el DXC en el FPCR puede establecerse por una condición de excepción de programa de excepción de datos.

Control de Filtrado de Interrupción de Programa (PIFC, por sus siglas en inglés) 216: Los bits 14-15 del campo I_2 son el control de filtrado de interrupción de programa (PIFC). El PIFC controla si ciertas clases de condiciones de excepción de programa (p.ej., excepción de dirección, excepción de datos, excepción de operación, excepción de protección, etc.) que ocurren mientras la CPU está en el modo de ejecución transaccional resultan en una interrupción.

El PIFC eficaz es el valor más alto del PIFC en la instrucción TBEGIN para el nivel de jerarquización actual y para todos los niveles externos. Cuando el PIFC eficaz es cero, todas las condiciones de excepción de programa resultan en una interrupción. Cuando el PIFC eficaz es uno, las condiciones de excepción de programa que tienen una clase de ejecución transaccional de 1 y 2 resultan en una interrupción. (A cada condición de excepción de programa se le asigna al menos una clase de ejecución transaccional, dependiendo de la gravedad de la excepción. La gravedad se basa en la probabilidad de recuperación durante una ejecución repetida de la ejecución transaccional, y si el sistema operativo necesita ver la interrupción). Cuando el PIFC eficaz es dos, las condiciones de excepción de programa que tienen una clase de ejecución transaccional de 1 resultan en una interrupción. Un PIFC de 3 se reserva.

Los bits 8-11 del campo I_2 (bits 40-43 de la instrucción) se reservan y deben contener ceros; de lo contrario, el programa puede no operar de manera compatible en el futuro.

Una realización de un formato de una instrucción Transaction Begin restringida (TBEGINC) se describe con referencia a las Figuras 3A-3B. En un ejemplo, TBEGINC 300 incluye un campo de código operacional 302 que incluye un código operacional que especifica una operación de iniciar transacción restringida; un campo base (B_1) 304; un campo de desplazamiento (D_1) 306; y un campo inmediato (I_2) 308. Los contenidos del registro general especificado por B_1 304 se añaden a D_1 306 para obtener la primera dirección del operando. Sin embargo, con la instrucción restringida de iniciar transacción, la primera dirección del operando no se usa para acceder al almacenamiento. En su lugar, el campo B_1 de la instrucción incluye ceros; de lo contrario, una excepción de especificación se reconoce.

En una realización, el campo I_2 incluye varios controles, un ejemplo de los cuales se representa en la Figura 3B.

Los bits del campo I_2 se definen de la siguiente manera, en un ejemplo:

Máscara de Guardado de Registro General (GRSM) 310: Los bits 0-7 del campo I_2 contienen la máscara de guardado de registro general (GRSM). Cada bit de la GRSM representa un par par-impar de registros generales, donde el bit 0 representa los registros 0 y 1, el bit 1 representa los registros 2 y 3, y así sucesivamente. Cuando un bit en la GRSM es cero, el par de registros correspondiente no se guarda. Cuando un bit en la GRSM es uno, el par de registros correspondiente se guarda en una ubicación dependiente del modelo que no es directamente accesible por el programa.

Si la transacción se aborta, los pares de registros guardados se restablecen a sus contenidos cuando la instrucción TRANSACTION BEGIN más externa se ejecutó. Los contenidos de todos los otros registros generales (no guardados) no se restablecen cuando una transacción restringida se aborta.

Cuando TBEGINC se usa para continuar la ejecución en el modo de ejecución de transacción no restringido, la máscara de guardado de registro general se ignora.

Permitir Modificación AR (A) 312: El control A, el bit 12 del campo I_2, controla si se permite que la transacción modifique un registro de acceso. El control de permiso de modificación AR eficaz es la AND lógica del control A en la instrucción TBEGINC para el nivel de jerarquización actual y para cualquier instrucción TBEGIN o TBEGINC externa.

Si el control A eficaz es cero, la transacción se abortará con el código de aborto 11 (instrucción restringida) si se realiza un intento de modificar cualquier registro de acceso. Si el control A eficaz es uno, la transacción no se abortará si un registro de acceso se modifica (en ausencia de cualquier otra condición de aborto).

Los bits 8-11 y 13-15 del campo I_2 (bits 40-43 y 45-47 de la instrucción) se reservan y deben contener ceros.

El fin de una instrucción Transaction Begin se especifica por una instrucción TRANSACTION END (TEND), cuyo formato se representa en la Figura 4. A modo de ejemplo, una instrucción TEND 400 incluye un campo de código operacional 402 que incluye un código operacional que especifica una operación de finalizar transacción.

Una cantidad de términos usados con respecto a la facilidad de ejecución transaccional y, por lo tanto, solo en aras de la conveniencia, una lista de términos se provee más abajo. En una realización, dichos términos tienen la siguiente definición:

Abortar: Una transacción se aborta cuando finaliza antes de una instrucción TRANSACTION END que resulta en una profundidad de jerarquización de transacción de cero. Cuando una transacción se aborta, ocurre lo siguiente, en una realización:

• Accesos al almacenamiento transaccional realizados por todos y cada uno de los niveles de la transacción se descartan (es decir, no se realizan).

• Accesos al almacenamiento no transaccional realizados por todos y cada uno de los niveles de la transacción se realizan.

• Los registros designados por la máscara de guardado de registro general (GRSM) de la instrucción TRANSACTION BEGIN más externa se restablecen a sus contenidos con anterioridad a la ejecución transaccional (es decir, a sus contenidos al momento de ejecución de la instrucción TRANSACTION BEGIN más externa). Los registros generales no designados por la máscara de guardado de registro general de la instrucción TRANSACTION BEGIN más externa no se restablecen.

• Los registros de acceso, registros de punto flotante y el registro de control de punto flotante no se restablecen. Cualquier cambio realizado en dichos registros durante la ejecución de transacción se retiene cuando la transacción se aborta.

Una transacción puede abortarse debido a una variedad de razones, incluidos la ejecución intentada de una instrucción restringida, la modificación intentada de un recurso restringido, conflicto transaccional, superación de varios recursos CPU, cualquier condición de interceptación de ejecución interpretativa, cualquier interrupción, una instrucción TRANSACTION ABORT, y otras razones. Un código de abortar transacción provee razones específicas por las cuales una transacción puede abortarse.

Un ejemplo de un formato de una instrucción TRANSACTION ABORT (TABORT) se describe con referencia a la Figura 5. A modo de ejemplo, una instrucción TABORT 500 incluye un campo de código operacional 502 que incluye un código operacional que especifica una operación de abortar transacción; un campo base (B_2) 504; y un campo de desplazamiento (D_2) 506. Cuando el campo B_2 es diferente de cero, los contenidos del registro general especificado por B_2 504 se añaden a D_2 506 para obtener una segunda dirección de operando; de lo contrario, la segunda dirección de operando se forma solamente a partir del campo D_2, y el campo B_2 se ignora. La segunda dirección de operando no se usa para datos de dirección; en su lugar, la dirección forma el código de abortar transacción que se coloca en un bloque de diagnóstico de transacción durante el procesamiento de aborto. El cálculo de dirección para la segunda dirección de operando sigue las reglas de aritmética de dirección: en el modo de dirección de 24 bits, los bits 0-29 se establecen en ceros; en el modo de dirección de 31 bits, los bits 0-32 se establecen en ceros.

Comprometer: Al completar una instrucción TRANSACTION END más externa, la CPU compromete los accesos al almacenamiento realizados por la transacción (a saber, la transacción más externa y cualquier nivel anidado) de modo que son visibles para otras CPU y subsistema E/S. Según lo observado por otras CPU y por el subsistema E/S, todos los accesos de captura y almacenamiento realizados por todos los niveles anidados de la transacción parecen ocurrir como una sola operación concurrente cuando el compromiso ocurre.

Los contenidos de los registros generales, registros de acceso, registros de punto flotante y el registro de control de punto flotante no se modifican por el proceso de comprometer. Cualquier cambio realizado en dichos registros durante la ejecución transaccional se retiene cuando los almacenamientos de la transacción se comprometen.

Conflicto: Un acceso transaccional llevado a cabo por una CPU está en conflicto con (a) un acceso transaccional o acceso no transaccional llevado a cabo por otra CPU, o (b) un acceso no transaccional llevado a cabo por el subsistema E/S, si ambos accesos son a cualquier ubicación dentro de la misma línea de caché, y uno o ambos

accesos es/son un almacenamiento.

Un conflicto puede detectarse por la ejecución especulativa de instrucciones de una CPU, aunque el conflicto puede no detectarse en la secuencia conceptual.

Transacción Restringida: Una transacción restringida es una transacción que se ejecuta en el modo de ejecución transaccional restringido y está sujeta a las siguientes limitaciones:

- Un subconjunto de las instrucciones generales está disponible.
- Un número limitado de instrucciones puede ejecutarse.
- Puede accederse a un número limitado de ubicaciones de operando de almacenamiento.
- La transacción se encuentra limitada a un solo nivel de jerarquización.

En ausencia de interrupciones repetidas o conflictos con otras CPU o con el subsistema E/S, una transacción restringida finalmente se completa y, por consiguiente, una rutina del manejador de abortos no se requiere. Las transacciones restringidas se describen en detalle más abajo.

Cuando una instrucción TRANSACTION BEGIN restringida (TBEGINC) se ejecuta mientras la CPU ya está en el modo de ejecución de transacción no restringida, la ejecución continúa como una transacción no restringida anidada.

Modo de Ejecución Transaccional Restringido: Cuando la profundidad de jerarquización de transacción es cero, y una transacción se inicia por una instrucción TBEGINC, la CPU entra en el modo de ejecución transaccional restringido. Mientras la CPU está en el modo de ejecución transaccional restringido, la profundidad de jerarquización de transacción es uno.

Transacción Anidada: Cuando la instrucción TRANSACTION BEGIN se emite mientras la CPU está en el modo de ejecución transaccional no restringido, la transacción se anida.

La facilidad de ejecución transaccional usa un modelo llamado jerarquización plana. En el modo de jerarquización plana, los almacenamientos realizados por una transacción interna no son observables por otras CPU y por el subsistema E/S hasta que la transacción más externa comprometa sus almacenamientos. De manera similar, si una transacción se aborta, todas las transacciones anidadas se abortan, y todos los almacenamientos transaccionales de todas las transacciones anidadas se descartan.

Un ejemplo de transacciones anidadas se representa en la Figura 6. Como se muestra, una primera TBEGIN 600 inicia una transacción más externa 601, TBEGIN 602 inicia una primera transacción anidada, y TBEGIN 604 inicia una segunda transacción anidada. En el presente ejemplo, TBEGIN 604 y TEND 606 definen una transacción más interna 608. Cuando TEND 610 se ejecuta, los almacenamientos transaccionales se comprometen 612 para la transacción más externa y todas las transacciones internas.

Transacción No Restringida: Una transacción no restringida es una transacción que se ejecuta en el modo de ejecución transaccional no restringido. Aunque una transacción no restringida no se encuentra limitada en la manera como una transacción restringida, puede aún abortarse debido a una variedad de motivos.

Modo de Ejecución Transaccional No Restringido: Cuando una transacción se inicia por la instrucción TBEGIN, la CPU entra en el modo de ejecución transaccional no restringido. Mientras la CPU está en el modo de ejecución transaccional no restringido, la profundidad de jerarquización de transacción puede variar de uno a la máxima profundidad de jerarquización de transacción.

Acceso No Transaccional: Los accesos no transaccionales son accesos de operando de almacenamiento realizados por la CPU cuando no está en el modo de ejecución transaccional (es decir, accesos de almacenamiento clásicos fuera de una transacción). Además, los accesos realizados por el subsistema E/S son accesos no transaccionales. Además, la instrucción NONTRANSACTIONAL STORE puede usarse para provocar un acceso de almacenamiento no transaccional mientras la CPU está en el modo de ejecución transaccional no restringido.

Un ejemplo de un formato de una instrucción NONTRANSACTIONAL STORE se describe con referencia a la Figura 7. Como un ejemplo, una instrucción NONTRANSACTIONAL STORE 700 incluye múltiples campos de código operacional 702a, 702b que especifican un código operacional que designa una operación de almacenamiento no transaccional; un campo de registro 704 que especifica un registro, cuyos contenidos se llaman el primer operando; un campo de índice (X_2) 706; un campo base (B_2) 708; un primer campo de desplazamiento (DL_2) 710; y un segundo campo de desplazamiento (DH_2) 712. Los contenidos de los registros generales designados por los campos X_2 y B_2 se añaden a los contenidos de una concatenación de contenidos de los campos DH_2 y DL_2 para formar la segunda dirección de operando. Cuando cualquiera de o ambos campos X_2 o B_2 son cero, el registro correspondiente no toma parte en la incorporación.

El primer operando de 64 bits se coloca, de manera no transaccional, sin cambios, en la segunda ubicación de

operando.

El desplazamiento, formado por la concatenación de los campos DH_2 y DL_2, se trata como un entero binario con signo de 20 bits.

El segundo operando se alineará en un límite de palabra doble; de lo contrario, la excepción de especificación se reconoce y la operación se suprime.

Transacción Externa/Más Externa: Una transacción con una profundidad de jerarquización de transacción de número más bajo es una transacción externa. Una transacción con un valor de profundidad de jerarquización de transacción de uno es la transacción más externa.

Una instrucción TRANSACTION BEGIN más externa es una que se ejecuta cuando la profundidad de jerarquización de transacción es inicialmente cero. Una instrucción TRANSACTION END más externa es una que hace que la profundidad de jerarquización de transacción pase de uno a cero. Una transacción restringida es la transacción más externa, en la presente realización.

Filtrado de Interrupción de Programa: Cuando una transacción se aborta debido a ciertas condiciones de excepción de programa, el programa puede, de manera opcional, evitar que la interrupción ocurra. La presente técnica se llama filtrado de interrupción de programa. El filtrado de interrupción de programa está sujeto a la clase transaccional de la interrupción, el control de filtrado de interrupción de programa eficaz de la instrucción TRANSACTION BEGIN, y el filtrado de interrupción de programa de ejecución transaccional se anulan en el registro de control 0.

Transacción: Una transacción incluye los accesos de operando de almacenamiento realizados, y los registros generales seleccionados alterados, mientras la CPU está en el modo de ejecución de transacción. Para una transacción no restringida, los accesos de operando de almacenamiento pueden incluir tanto accesos transaccionales como accesos no transaccionales. Para una transacción restringida, los accesos de operando de almacenamiento se encuentran limitados a accesos transaccionales. Según se observa por otras CPU y por el subsistema E/S, todos los accesos de operando de almacenamiento realizados por la CPU mientras está en el modo de ejecución de transacción parecen ocurrir como una sola operación concurrente. Si una transacción se aborta, los accesos de almacenamiento transaccional se descartan, y cualquier registro designado por la máscara de guardado de registro general de la instrucción TRANSACTION BEGIN más externa se restablece a su contenido con anterioridad a la ejecución transaccional.

Accesos Transaccionales: Los accesos transaccionales son accesos de operando de almacenamiento realizados mientras la CPU está en el modo de ejecución transaccional, con la excepción de accesos realizados por la instrucción NONTRANSACTIONAL STORE.

Modo de Ejecución Transaccional: La expresión modo de ejecución transaccional (también conocido como modo de ejecución de transacción) describe la operación común de los modos de ejecución transaccional no restringidos y restringidos. Por consiguiente, cuando se describe la operación, los términos no restringido y restringido se usan para calificar el modo de ejecución transaccional.

Cuando la profundidad de jerarquización de transacción es cero, la CPU no está en el modo de ejecución transaccional (también llamado el modo de ejecución no transaccional).

Según lo observado por la CPU, las capturas y almacenamiento realizados en el modo de ejecución transaccional no son diferentes de aquellos realizados mientras no se está en el modo de ejecución transaccional.

En un ejemplo de z/Architecture, la facilidad de ejecución transaccional está bajo el control de los bits 8-9 del registro de control 0, los bits 61-63 del registro de control 2, la profundidad de jerarquización de transacción, la dirección de bloque de diagnóstico de transacción, y la palabra de estado de programa (PSW) de aborto de transacción.

Siguiendo una reiniciación de CPU inicial, los contenidos de las posiciones de bit 8-9 del registro de control 0, las posiciones de bit 62-63 del registro de control 2, y la profundidad de jerarquización de transacción se establecen en cero.

Cuando el control de ejecución transaccional, bit 8 del registro de control 0, es cero, la CPU no puede colocarse en el modo de ejecución transaccional.

Detalles adicionales sobre los varios controles se describen más abajo.

Según se indica, la facilidad de ejecución transaccional se controla por dos bits en el registro de control cero y tres bits en el registro de control dos. Por ejemplo:

Bits del Registro de Control 0: Las asignaciones de bits son de la siguiente manera, en un ejemplo:

Control de Ejecución Transaccional (TXC): El bit 8 del registro de control cero es el control de ejecución transaccional. Dicho bit provee un mecanismo por medio del cual el programa de control (p.ej., sistema operativo)

puede indicar si la facilidad de ejecución transaccional es utilizable o no por el programa. El bit 8 será uno que entre con éxito en el modo de ejecución transaccional.

Cuando el bit 8 del registro de control 0 es cero, la ejecución intentada de las instrucciones EXTRACT TRANSACTION NESTING DEPTH, TRANSACTION BEGIN y TRANSACTION END resulta en una ejecución de operación especial.

Un ejemplo de un formato de una instrucción EXTRACT TRANSACTION NESTING DEPTH se describe con referencia a la Figura 8. A modo de ejemplo, una instrucción EXTRACT TRANSACTION NESTING DEPTH 800 incluye un campo de código operacional 802 que especifica un código operacional que indica la operación de extraer profundidad de jerarquización de transacción; y un campo de registro R_1 804 que designa un registro general.

La profundidad de jerarquización de transacción actual se coloca en los bits 48-63 del registro general R_1. Los bits 0-31 del registro permanecen sin cambios, y los bits 32-47 del registro se establecen en cero.

En un ejemplo adicional, la profundidad de jerarquización de transacción máxima se coloca también en el registro general R_1 como, por ejemplo, en los bits 16-31.

Anulación de Filtrado de Interrupción de Programa (PIFO, por sus siglas en inglés) de Ejecución de Transacción: El bit 9 del registro de control cero es la anulación de filtrado de interrupción de programa de ejecución transaccional. Dicho bit provee un mecanismo por medio del cual el programa de control puede asegurar que cualquier condición de excepción de programa que ocurra mientras la CPU está en el modo de ejecución transaccional resulta en una interrupción, independientemente del control de filtrado de interrupción de programa eficaz especificado o implícito por la instrucción TRANSACTION BEGIN.

Bits del Registro de Control 2: Las asignaciones son de la siguiente manera, en una realización:

Alcance de Diagnóstico de Transacción (TDS, por sus siglas en inglés): El bit 61 del registro de control 2 controla la aplicabilidad del control de diagnóstico de transacción (TDC, por sus siglas en inglés) en los bits 62-63 del registro, de la siguiente manera:

TDS

Valor Significado

0 El TDC se aplica independientemente de si la CPU está en el estado de problema o supervisor.

1 El TDC se aplica solamente cuando la CPU está en el estado de problema. Cuando la CPU está en el estado de supervisor, el procesamiento es como si el TDC contuviera cero.

Control de Diagnóstico de Transacción (TDC): Los bits 62-63 del registro de control 2 son un entero sin signo de 2 bits que puede usarse para hacer que las transacciones se aborten de forma aleatoria con fines de diagnóstico. La codificación del TDC es de la siguiente manera, en un ejemplo:

TDC

Valor Significado

0 Operación normal; las transacciones no se abortan como resultado del TDC.

1 Abortar cada transacción en una instrucción aleatoria, pero antes de la ejecución de la instrucción TRANSACTION END más externa.

2 Abortar transacciones aleatorias en una instrucción aleatoria.

3 Reservado

Cuando una transacción se aborta debido a un TDC diferente de cero, entonces cualquiera de las siguientes puede ocurrir:

• El código de aborto se establece en cualquiera de los códigos 7-11, 13-16, o 255, con el valor del código elegido de forma aleatoria por la CPU; el código de condición se establece correspondiendo al código de aborto. Los códigos de aborto se describen en mayor detalle más abajo.

• Para una transacción no restringida, el código de condición se establece en uno. En el presente caso, el código de aborto no es aplicable.

Depende del modelo si el valor TDC 1 se implementa. Si no se implementa, un valor de 1 actúa como si 2 se especificara.

Para una transacción restringida, un valor TDC de 1 se trata como si un valor TDC de 2 se especificara.

Si un valor TDC de 3 se especifica, los resultados son impredecibles.

Dirección de Bloque de Diagnóstico de Transacción (TDBA, por sus siglas en inglés)

Una dirección de bloque de diagnóstico de transacción (TDBA) válida se establece a partir de la primera dirección de operando de la instrucción TRANSACTION BEGIN (TBEGIN) más externa cuando el campo B_1 de la instrucción es diferente de cero. Cuando la CPU está en el modo de espacio primario o modo de registro de acceso, la TDBA designa una ubicación en el espacio de dirección primario. Cuando la CPU está en el modo de espacio secundario, o modo de espacio doméstico, la TDBA designa una ubicación en el espacio de dirección secundario o doméstico, respectivamente. Cuando DAT (Traducción Dinámica de Direcciones) está apagada, la TDBA designa una ubicación en el almacenamiento real.

La TDBA se usa por la CPU para ubicar el bloque de diagnóstico de transacción - llamado TDB de TBEGIN especificada - si la transacción se aborta posteriormente. Los tres bits más a la derecha de la TDBA son cero, lo cual significa que la TDB de TBEGIN especificada está en un límite de palabra doble.

Cuando el campo B_1 de una instrucción TRANSACTION BEGIN (TBEGIN) más externa es cero, la dirección de bloque de diagnóstico transaccional es inválida, y ningún TDB de TBEGIN especificada se almacena si la transacción se aborta posteriormente.

PSW de Aborto de Transacción (TAPSW, por sus siglas en inglés)

Durante la ejecución de la instrucción TRANSACTION BEGIN (TBEGIN) cuando la profundidad de jerarquización es inicialmente cero, la PSW de aborto de transacción se establece en los contenidos de la PSW actual; y la dirección de instrucción de la PSW de aborto de transacción designa la siguiente instrucción secuencial (es decir, la instrucción que sigue a la TBEGIN más externa). Durante la ejecución de la instrucción TRANSACTION BEGIN restringida (TBEGINC) cuando la profundidad de jerarquización es inicialmente cero, la PSW de aborto de transacción se establece en los contenidos de la PSW actual, excepto que la dirección de instrucción de la PSW de aborto de transacción designa la instrucción TBEGINC (antes que la siguiente instrucción secuencial que sigue a la TBEGINC).

Cuando una transacción se aborta, el código de condición en la PSW de aborto de transacción se reemplaza por un código que indica la gravedad de la condición de aborto. Posteriormente, si la transacción se aborta debido a causas que no resultan en una interrupción, la PSW se carga desde la PSW de aborto de transacción; si la transacción se aborta debido a causas que resultan en una interrupción, la PSW de aborto de transacción se almacena como la PSW antigua de interrupción.

La PSW de aborto de transacción no se altera durante la ejecución de cualquier instrucción TRANSACTION BEGIN interna.

Profundidad de Jerarquización de Transacción (TND, por sus siglas en inglés)

La profundidad de jerarquización de transacción es, por ejemplo, un valor sin signo de 16 bits que aumenta cada vez que una instrucción TRANSACTION BEGIN se completa con el código de condición 0 y disminuye cada que vez que una instrucción TRANSACTION END se completa. La profundidad de jerarquización de transacción se restablece en cero cuando una transacción se aborta o por una reiniciación de la CPU.

En una realización, una TND máxima de 15 se implementa.

En una implementación, cuando la CPU está en el modo de ejecución transaccional restringido, la profundidad de jerarquización de transacción es uno. De manera adicional, aunque la TND máxima puede representarse como un valor de 4 bits, la TND se define para que sea un valor de 16 bits para facilitar su inspección en el bloque de diagnóstico de transacción.

Bloque de Diagnóstico de Transacción (TDB)

Cuando una transacción se aborta, información de estado puede guardarse en un bloque de diagnóstico de transacción (TDB), de la siguiente manera:

1. TDB de TBEGIN especificada: Para una transacción no restringida, cuando el campo B_1 de la instrucción TBEGIN más externa es diferente de cero, la primera dirección de operando de la instrucción designa el TDB de TBEGIN especificada. Esta es una ubicación especificada por el programa de aplicación que puede examinarse por el manejador de abortos de la aplicación.

2. TDB de Interrupción de Programa (PI, por sus siglas en inglés): Si una transacción no restringida se aborta debido a una condición de excepción de programa no filtrada, o si una transacción restringida se aborta debido a cualquier condición de excepción de programa (es decir, cualquier condición que resulta en que una interrupción de programa

se reconoce), la PI - TDB se almacena en ubicaciones en el área de prefijo. Esto está disponible para el sistema operativo para inspeccionar y cerrar sesión en cualquier informe de diagnóstico que pueda proveer.

3. TDB de Interceptación: Si la transacción se aborta debido a cualquier condición de excepción de programa que resulta en la interceptación (es decir, la condición hace que la ejecución interpretativa finalice y controle el regreso al programa anfitrión), un TDB se almacena en una ubicación especificada en el bloque de descripción de estado para el sistema operativo anfitrión.

El TDB de TBEGIN especificada solo se almacena, en una realización, cuando la dirección TDB es válida (es decir, cuando el campo B_1 de la instrucción TBEGIN más externa es diferente de cero).

Para abortos debidos a condiciones de excepción de programa no filtradas, solo una de la PI -TDB o TDB de Interceptación se almacenará. Por consiguiente, puede haber cero, uno o dos TDB almacenados para un aborto.

Detalles adicionales con respecto a un ejemplo de cada uno de los TDB se describen más abajo:

TDB de TBEGIN especificado: La ubicación de 256 bytes especificada por una dirección de bloque de diagnóstico de transacción válida. Cuando la dirección de bloque de diagnóstico de transacción es válida, el TDB de TBEGIN especificada se almacena en un aborto de transacción. El TDB de TBEGIN especificada está sujeto a todos los mecanismos de protección de almacenamiento que están vigentes al momento de ejecución de la instrucción TRANSACTION BEGIN más externa. Un episodio de alteración de almacenamiento PER (Registro de Episodios de Programas) para cualquier porción del TDB de TBEGIN especificada se detecta durante la ejecución de la TBEGIN más externa, no durante el procesamiento de aborto de transacción.

Un propósito de PER es asistir en la depuración de programas. Ello permite alertar al programa sobre los siguientes tipos de episodios, como ejemplos:

• Ejecución de una instrucción de derivación exitosa. La opción se provee con un evento que ocurre solamente cuando la ubicación objetivo de derivación se encuentra dentro del área de almacenamiento designada.

• Captura de una instrucción del área de almacenamiento designada.

• Alteración de los contenidos del área de almacenamiento designada. La opción se provee con un evento que ocurre solamente cuando el área de almacenamiento se encuentra dentro de espacios de dirección designados.

• Ejecución de una instrucción STORE USING REAL ADDRESS.

• Ejecución de la instrucción TRANSACTION END.

El programa puede, de manera selectiva, especificar que uno o más de los tipos de episodios de más arriba se reconozcan, excepto que el episodio para STORE USING REAL ADDRESS puede especificarse solamente junto con el episodio de alteración de almacenamiento. La información relativa a un episodio PER se provee al programa por medio de una interrupción de programa, con la causa de la interrupción identificándose en el código de interrupción.

Cuando la dirección de bloque de diagnóstico de transacción no es válida, un TDB de TBEGIN especificada no se almacena.

TDB de Interrupción de Programa: Ubicaciones reales 6,144-6,399 (1800-18FF hexadecimal). El TDB de interrupción de programa se almacena cuando una transacción se aborta debido a una interrupción de programa. Cuando una transacción se aborta debido a otras causas, los contenidos del TDB de interrupción de programa son impredecibles.

El TDB de interrupción de programa no está sujeto a un mecanismo de protección. Los episodios de alteración de almacenamiento PER no se detectan para el TDB de interrupción de programa cuando se almacena durante una interrupción de programa.

TDB de Interceptación: La ubicación real de anfitrión de 256 bytes especificada por las ubicaciones 488-495 de la descripción de estado. El TDB de interceptación se almacena cuando una transacción abortada resulta en una interceptación de interrupción de programa invitado (es decir, código de interceptación 8). Cuando una transacción se aborta debido a otras causas, los contenidos del TDB de interceptación son impredecibles. El TDB de interceptación no está sujeto a un mecanismo de protección.

Como se representa en la Figura 9, los campos de un bloque de diagnóstico de transacción 900 son los siguientes, en una realización:

Formato 902: El byte 0 contiene una indicación de validez y formato, de la siguiente manera:

Valor Significado

0 Los restantes campos del TDB son impredecibles.

1 Un TDB de formato 1, cuyos restantes campos se describen más abajo.

2-255 Reservado

Se hace referencia a un TDB en el cual el campo de formato es cero como un TDB nulo.

Banderas 904: El byte 1 contiene varias indicaciones, de la siguiente manera:

Validez de Testigo de Conflicto (CTV, por sus siglas en inglés): Cuando una transacción se aborta debido a un conflicto de captura o almacenamiento (es decir, códigos de aborto 9 o 10, respectivamente), el bit 0 del byte 1 es la indicación de validez de testigo de conflicto. Cuando la indicación CTV es uno, el testigo de conflicto 910 en los bytes 16-23 del TDB contienen la dirección lógica en la cual el conflicto se ha detectado. Cuando la indicación CTV es cero, los bytes 16-23 del TDB son impredecibles.

Cuando una transacción se aborta debido a cualquier motivo diferente de un conflicto de captura o almacenamiento, el bit 0 del byte 1 se almacena como cero.

Indicación de Transacción Restringida (CTI, por sus siglas en inglés): Cuando la CPU está en el modo de ejecución transaccional restringido, el bit 1 del byte 1 se establece en uno. Cuando la CPU está en el modo de ejecución transaccional no restringido, el bit 1 del byte 1 se establece en cero.

Reservado: Los bits 2-7 del byte 1 están reservados, y se almacenan como ceros.

Profundidad de Jerarquización de Transacción (TND) 906: Los bytes 6-7 contienen la profundidad de jerarquización de transacción cuando la transacción se abortó.

Código de Aborto de Transacción (TAC, por sus siglas en inglés) 908: Los bytes 8-15 contienen un código de aborto de transacción sin signo de 64 bits. Cada punto de código indica una razón por la cual una transacción se aborta.

Depende del modelo si el código de aborto de transacción se almacena en el TDB de interrupción de programa cuando una transacción se aborta debido a condiciones diferentes de una interrupción de programa.

Testigo de Conflicto 910: Para las transacciones que se abortan debido al conflicto de captura o almacenamiento (es decir, códigos de aborto 9 y 10, respectivamente), los bytes 16-23 contienen la dirección lógica de la ubicación de almacenamiento en la cual el conflicto se ha detectado. El testigo de conflicto es significativo cuando el bit CTV, bit 0 del byte 1, es uno.

Cuando el bit CTV es cero, los bytes 16-23 son impredecibles.

Debido a la ejecución especulativa por la CPU, el testigo de conflicto puede designar una ubicación de almacenamiento a la que no se accederá necesariamente por la secuencia de ejecución conceptual de la transacción.

Dirección de Instrucción de Transacción Abortada (ATIA, por sus siglas en inglés) 912: Los bytes 24-31 contienen una dirección de instrucción que identifica la instrucción que se estaba ejecutando cuando un aborto se detectó. Cuando una transacción se aborta debido a los códigos de aborto 2, 5, 6, 11, 13 o 256 o superiores, o cuando una transacción se aborta debido a los códigos de aborto 4 o 13 y la condición de excepción de programa es de anulación, la ATIA señala directamente la instrucción que se estaba ejecutando. Cuando una transacción se aborta debido a los códigos de aborto 4 o 12, y la condición de excepción de programa no es de anulación, la ATIA señala más allá de la instrucción que se estaba ejecutando.

Cuando una transacción se aborta debido a los códigos de aborto 7-10, 14-16, o 255, la ATIA no indica necesariamente la instrucción exacta que provoca el aborto, pero puede señalar una instrucción anterior o posterior dentro de la transacción.

Si una transacción se aborta debido a una instrucción que es el objetivo de una instrucción tipo ejecutar, la ATIA identifica la instrucción tipo ejecutar, ya sea señalando la instrucción o más allá de esta, dependiendo del código de aborto según se describe más arriba. La ATIA no indica el objetivo de la instrucción tipo ejecutar.

La ATIA está sujeta al modo de direccionamiento cuando la transacción se aborta. En el modo de direccionamiento de 24 bits, los bits 0-40 del campo contienen ceros. En el modo de direccionamiento de 31 bits, los bits 0-32 del campo contienen ceros.

Depende del modelo si la dirección de instrucción de transacción abortada se almacena en el TDB de interrupción de programa cuando una transacción se aborta debido a condiciones diferentes de una interrupción de programa.

Cuando una transacción se aborta debido al código de aborto 4 o 12, y la condición de excepción de programa no es de anulación, la ATIA no señala la instrucción que provoca el aborto. Mediante la resta del número de medias

palabras indicadas por el código de longitud de interrupción (ILC, por sus siglas en inglés) de la ATIA, la instrucción que provoca el aborto puede identificarse en condiciones que son de supresión o terminación, o para episodios diferentes de PER que se están completando. Cuando una transacción se aborta debido a un episodio PER, y ninguna otra condición de excepción de programa está presente, la ATIA es impredecible.

Cuando la dirección de bloque de diagnóstico de transacción es válida, el ILC puede examinarse en la identificación de interrupción de programa (PIID, por sus siglas en inglés) en los bytes 36-39 del TDB de TBEGIN especificada. Cuando el filtrado no se aplica, el ILC puede examinarse en la PIID en la ubicación 140-143 en el almacenamiento real.

Identificación de Acceso de Excepción (EAID, por sus siglas en inglés) 914: Para transacciones que se abortan debido a ciertas condiciones de excepción de programa filtradas, el byte 32 del TDB de TBEGIN especificada contiene la identificación de acceso de excepción. En un ejemplo de z/Architecture, el formato de la EAID, y los casos para los cuales se almacena, son iguales a aquellos descritos en la ubicación real 160 cuando la condición de excepción resulta en una interrupción, según se describe en los Principios de Operación a los que se hace referencia más arriba.

Para las transacciones que se abortan por otros motivos, incluida cualquier condición de excepción que resulta en una interrupción de programa, el byte 32 es impredecible. El byte 32 es impredecible en el TDB de interrupción de programa.

El presente campo se almacena solamente en el TDB designado por la dirección de bloque de diagnóstico de transacción; de lo contrario, el campo se reserva. La EAID se almacena solamente para la lista de acceso controlada o protección DAT, tipo ASCE, traducción de páginas, primera traducción de región, segunda traducción de región, tercera traducción de región, y condiciones de excepción de programa de traducción de segmentos.

Código de Excepción de Datos (DXC) 916: Para transacciones que se abortan debido a condiciones de excepción de programa de excepción de datos filtradas, el byte 33 del TDB de TBEGIN especificada contiene el código de excepción de datos. En un ejemplo de z/Architecture, el formato de la DXC, y los casos para los cuales se almacena, son iguales a aquellos descritos en la ubicación real 147 cuando la condición de excepción resulta en una interrupción, según se describe en los Principios de Operación a los que se hace referencia más arriba. En un ejemplo, la ubicación 147 incluye el DXC.

Para las transacciones que se abortan por otros motivos, incluida cualquier condición de excepción que resulta en una interrupción de programa, el byte 33 es impredecible. El byte 33 es impredecible en el TDB de interrupción de programa.

El presente campo se almacena solamente en el TDB designado por la dirección de bloque de diagnóstico de transacción; de lo contrario, el campo se reserva. El DXC se almacena solamente para condiciones de excepción de programa de datos.

Identificación de Interrupción de Programa (PIID) 918: Para transacciones que se abortan debido a condiciones de excepción de programa filtradas, los bytes 36-39 del TDB de TBEGIN especificada contienen la identificación de interrupción de programa. En un ejemplo de z/Architecture, el formato de la PIID es igual a aquel descrito en las ubicaciones reales 140-143 cuando la condición resulta en una interrupción (según se describe en los Principios de Operación a los que se hace referencia más arriba), excepto que el código de longitud de instrucción en los bits 13-14 de la PIID es con respecto a la instrucción en la cual la condición de excepción se ha detectado.

Para las transacciones que se abortan por otros motivos, incluidas las condiciones de excepción que resultan en una interrupción de programa, los bytes 36-39 son impredecibles. Los bytes 36-39 son impredecibles en el TDB de interrupción de programa.

El presente campo se almacena solamente en el TDB designado por la dirección de bloque de diagnóstico de transacción; de lo contrario, el campo se reserva. La identificación de interrupción de programa solo se almacena para las condiciones de excepción de programa.

Identificación de Excepción de Traducción (TEID, por sus siglas en inglés) 920: Para transacciones que se abortan debido a cualquiera de las siguientes condiciones de excepción de programa filtradas, los bytes 40-47 del TDB de TBEGIN especificada contienen la identificación de excepción de traducción.

- Lista de acceso controlada o protección DAT
- Tipo ASCE
- Traducción de páginas
- Primera traducción de región
- Segunda traducción de región

• Tercera traducción de región

• Excepción de traducción de segmentos

En un ejemplo de z/Architecture, el formato de la TEID es igual a aquel descrito en las ubicaciones reales 168-175 cuando la condición resulta en una interrupción, según se describe en los Principios de Operación a los que se hace referencia más arriba.

Para las transacciones que se abortan por otros motivos, incluidas las condiciones de excepción que resultan en una interrupción de programa, los bytes 40-47 son impredecibles. Los bytes 40-47 son impredecibles en el TDB de interrupción de programa.

El presente campo se almacena solamente en el TDB designado por la dirección de bloque de diagnóstico de transacción; de lo contrario, el campo se reserva.

Dirección de Episodios de Interrupción 922: Para transacciones que se abortan debido a condiciones de excepción de programa filtradas, los bytes 48-55 del TDB de TBEGIN especificada contienen la dirección de episodio de interrupción. En un ejemplo de z/Architecture, el formato de la dirección de episodio de interrupción es igual a aquel descrito en las ubicaciones reales 272-279 cuando la condición resulta en una interrupción, según se describe en los Principios de Operación a los que se hace referencia más arriba.

Para las transacciones que se abortan por otros motivos, incluidas las condiciones de excepción que resultan en una interrupción de programa, los bytes 48-55 son impredecibles. Los bytes 48-55 son impredecibles en el TDB de interrupción de programa.

El presente campo se almacena solamente en el TDB designado por la dirección de bloque de diagnóstico de transacción; de lo contrario, el campo se reserva.

Detalles adicionales con respecto a los episodios de interrupción se describen más abajo.

En un ejemplo de z/Architecture, cuando la facilidad PER-3 se instala, provee al programa la dirección de la última instrucción para provocar una interrupción en la ejecución secuencial de la CPU. El registro de dirección de episodio de interrupción puede usarse como una ayuda de depuración para la detección de derivación salvaje. La presente facilidad provee, por ejemplo, un registro de 64 bits en la CPU, llamado el registro de dirección de episodio de interrupción. Cada vez que una instrucción diferente de TRANSACTION ABORT provoca una interrupción en la ejecución de instrucción secuencial (es decir, la dirección de instrucción en la PSW se reemplaza, antes que aumentarse por la longitud de la instrucción), la dirección de dicha instrucción se coloca en el registro de direcciones de episodios de interrupción. Cuando ocurre una interrupción de programa, independientemente de si el PER se indica o no, los contenidos actuales del registro de dirección de episodio de interrupción se colocan en ubicaciones de almacenamiento reales 272-279.

Si la instrucción que provoca el episodio de interrupción es el objetivo de una instrucción tipo ejecutar (EXECUTE o EXECUTE RELATIVE LONG), entonces la dirección de instrucción usada para capturar la instrucción tipo ejecutar se coloca en el registro de dirección de episodio de interrupción.

En un ejemplo de z/Architecture, se considera que un episodio de interrupción ocurre cuando una de las siguientes instrucciones provoca la derivación: BRANCH AND LINK (BAL, BALR); BRANCH AND SAVE (BAS, BASR); BRANCH AND SAVE AND SET MODE (BASSM); BRANCH AND SET MODE (BSM); BRANCH AND STACK (BAKR); BRANCH ON CONDITION (BC, BCR); BRANCH ON COUNT (BCT, BCTR, BCTG, BCTGR); BRANCH ON INDEX HIGH (BXH, BXHG); BRANCH ON INDEX LOW OR EQUAL (BXLE, BXLEG); BRANCH RELATIVE ON CONDITION (BRC); BRANCH RELATIVE ON CONDITION LONG (BRCL); BRANCH RELATIVE ON COUNT (BRCT, BRCTG); BRANCH RELATIVE ON INDEX HIGH (BRXH, BRXHG); BRANCH RELATIVE ON INDEX LOW OR EQUAL (BRXLE, BRXLG); COMPARE AND BRANCH (CRB, CGRB); COMPARE AND BRANCH RELATIVE (CRJ, CGRJ); COMPARE IMMEDIATE AND BRANCH (CIB, CGIB); COMPARE IMMEDIATE AND BRANCH RELATIVE (CIJ, CGIJ); COMPARE LOGICAL AND BRANCH (CLRB, CLGRB); COMPARE LOGICAL AND BRANCH RELATIVE (CLRJ, CLGRJ); COMPARE LOGICAL IMMEDIATE AND BRANCH (CLIB, CLGIB); y COMPARE LOGICAL IMMEDIATE AND BRANCH RELATIVE (CLIJ, CLGIJ).

También se considera que un episodio de interrupción ocurre cuando una de las siguientes instrucciones se completa: BRANCH AND SET AUTHORITY (BSA); BRANCH IN SUBSPACE GROUP (BSG); BRANCH RELATIVE AND SAVE (BRAS); BRANCH RELATIVE AND SAVE LONG (BRASL); LOAD PSW (LPSW); LOAD PSW EXTENDED (LPSWE); PROGRAM CALL (PC); PROGRAM RETURN (PR); PROGRAM TRANSFER (PT); PROGRAM TRANSFER WITH INSTANCE (PTI); RESUME PROGRAM (RP); y TRAP (TRAP2, TRAP4).

No se considera que un episodio de interrupción ocurre como resultado de una transacción que se está abortando (ya sea de manera implícita o como resultado de la instrucción TRANSACTION ABORT).

Información de Diagnóstico Dependiente del Modelo 924: Los bytes 112-127 contienen información de diagnóstico

dependiente del modelo.

Para todos los códigos de aborto excepto 12 (interrupción de programa filtrada), la información de diagnóstico dependiente del modelo se guarda en cada TDB que se almacena.

En un ejemplo, la información de diagnóstico dependiente del modelo incluye lo siguiente:

• Los bytes 112-119 contienen un vector de 64 bits llamado las indicaciones de derivación de ejecución transaccional (TXBI, por sus siglas en inglés). Cada uno de los primeros 63 bits del vector indica los resultados de la ejecución de una instrucción de derivación mientras la CPU estaba en el modo de ejecución transaccional, de la siguiente manera:

Valor Significado

0 La instrucción se ha completado sin derivación.

1 La instrucción se ha completado con derivación.

El bit 0 representa el resultado de la primera instrucción de derivación, el bit 1 representa el resultado de la segunda instrucción, etc.

Si menos de 63 instrucciones de derivación se han ejecutado mientras la CPU estaba en el modo de ejecución transaccional, los bits más a la derecha que no corresponden a instrucciones de derivación se establecen en ceros (incluido el bit 63). Cuando más de 63 instrucciones de derivación se han ejecutado, el bit 63 de las TXBI se establece en uno.

Los bits en las TXBI se establecen por instrucciones que pueden provocar un episodio de interrupción, según se enumera más arriba, a excepción de lo siguiente:

- Una instrucción restringida no hace que un bit se establezca en las TXBI.

- Para las instrucciones de, por ejemplo, z/Architecture, cuando el campo M_1 de la instrucción BRANCH ON CONDITION, BRANCH RELATIVE ON CONDITION o BRANCH RELATIVE ON CONDITION LONG es cero, o cuando el campo R_2 de las siguientes instrucciones es cero, depende del modelo si la ejecución de la instrucción hace que un bit se establezca en las TXBI.

• BRANCH AND LINK (BALR);BRANCH AND SAVE (BASR); BRANCH AND SAVE AND SET MODE (BASSM); BRANCH AND SET MODE (BSM); BRANCH ON CONDITION (BCR); y BRANCH ON COUNT (BCTR, BCTGR)

• Para condiciones de aborto que se han provocado por una excepción de acceso de anfitrión, la posición de bit 0 del byte 127 se establece en uno. Para todas las otras condiciones de aborto, la posición de bit 0 del byte 127 se establece en cero.

• Para condiciones de aborto que se han detectado por la unidad de carga/almacenamiento (LSU, por sus siglas en inglés), los cinco bits más a la derecha del byte 127 contienen una indicación de la causa. Para condiciones de aborto que no se han detectado por la LSU, el byte 127 se reserva.

Registros Generales 930: Los bytes 128-255 contienen los contenidos de los registros generales 0-15 al momento en el que la transacción se ha abortado. Los registros se almacenan en orden ascendente, comenzando con el registro general 0 en los bytes 128-135, el registro general 1 en los bytes 136-143, etc.

Reservado: Todos los otros campos están reservados. Salvo que se indique lo contrario, los contenidos de campos reservados son impredecibles.

Según lo observado por otras CPU y el subsistema E/S, el almacenamiento de los TDB durante un aborto de transacción es una referencia de acceso múltiple que ocurre después de almacenamientos no transaccionales.

Una transacción puede abortarse debido a motivos que se encuentran fuera del alcance de la configuración inmediata en la cual se ejecuta. Por ejemplo, episodios transitorios reconocidos por un hipervisor (como, por ejemplo, LPAR o z/VM) pueden hacer que una transacción se aborte.

La información provista en el bloque de diagnóstico de transacción se pretende con fines de diagnóstico y es sustancialmente correcta. Sin embargo, dado que un aborto puede haber sido provocado por un episodio fuera del alcance de la configuración inmediata, información como, por ejemplo, el código de aborto o la identificación de interrupción de programa, puede no reflejar, de manera exacta, condiciones dentro de la configuración y, por consiguiente, no debe usarse al determinar la acción de programa.

Además de la información de diagnóstico guardada en TDB, cuando una transacción se aborta debido a cualquier condición de excepción de programa de excepción de datos y tanto el control de registro AFP, bit 45 del registro de control 0, como el control de operación de permiso de punto flotante eficaz (F) son uno, el código de excepción de

datos (DXC) se coloca en el byte 2 del registro de control de punto flotante (FPCR), independientemente de si el filtrado se aplica a la condición de excepción de programa. Cuando una transacción se aborta, y cualquiera de o ambos del control de registro AFP o control de operación de permiso de punto flotante eficaz son cero, DXC no se coloca en el FPCR.

En una realización, según se indica en la presente memoria, cuando la facilidad de ejecución transaccional se instala, se proveen las siguientes instrucciones generales.

• EXTRACT TRANSACTION NESTING DEPTH

• NONTRANSACTIONAL STORE

• TRANSACTION ABORT

• TRANSACTION BEGIN

• TRANSACTION END

Cuando la CPU está en el modo de ejecución transaccional, la ejecución intentada de ciertas instrucciones es restringida y hace que la transacción se aborte.

Cuando se emite en el modo de ejecución transaccional restringido, la ejecución intentada de instrucciones restringidas puede también resultar en una interrupción de programa de restricción de transacción, o puede resultar en el procedimiento de ejecución como si la transacción no estuviera restringida.

En un ejemplo de z/Architecture, las instrucciones restringidas incluyen, como ejemplos, las siguientes instrucciones no privilegiadas: COMPARE AND SWAP AND STORE; MODIFY RUNTIME INSTRUMENTATION CONTROLS; PERFORM LOCKED OPERATION; PREFETCH DATA (RELATIVE LONG), cuando el código en el campo M_1 es 6 o 7; STORE CHARACTERS UNDER MASK HIGH, cuando el campo M_3 es cero y el código en el campo R_1 es 6 o 7; STORE FACILITY LIST EXTENDED; STORE RUNTIME INSTRUMENTATION CONTROLS; SUPERVISOR CALL; y TEST RUNTIME INSTRUMENTATION CONTROLS.

En la lista de más arriba, COMPARE AND SWAP AND STORE y PERFORM LOCKED OPERATION son instrucciones complejas que pueden implementarse de manera más eficaz por el uso de instrucciones básicas en el modo TX. Los casos para PREFETCH DATA y PREFETCH DATA RELATIVE LONG son restringidos dado que los códigos de 6 y 7 liberan una línea de caché y, por consiguiente, necesitan el compromiso de los datos potencialmente antes de la finalización de una transacción. SUPERVISOR CALL está restringida dado que provoca una interrupción (que hace que una transacción se aborte).

Bajo las condiciones enumeradas más abajo, las siguientes instrucciones están restringidas:

• BRANCH AND LINK (BALR);BRANCH AND SAVE (BASR), y BRANCH AND SAVE AND SET MODE, cuando el campo R_2 de la instrucción es diferente de cero y el seguimiento de derivación se permite.

• BRANCH AND SAVE AND SET MODE y BRANCH AND SET MODE, cuando el campo R_2 es diferente de cero y el seguimiento de modo se permite; SET ADDRESSING MODE, cuando el seguimiento de modo se permite.

• MONITOR CALL, cuando una condición de episodio de monitoreo se reconoce.

La lista de más arriba incluye instrucciones que pueden formar entradas de seguimiento. Si se permite que dichas instrucciones se ejecuten de manera transaccional y entradas de seguimiento formadas, y la transacción se aborta posteriormente, el puntero de tabla de seguimiento en el registro de control 12 avanzará, pero los almacenamientos a la tabla de seguimiento se descartarán. Ello dejará a un espacio vacío incoherente en la tabla de seguimiento. Por consiguiente, las instrucciones se restringen en los casos donde formarán entradas de seguimiento.

Cuando la CPU está en el modo de ejecución transaccional, depende del modelo si las siguientes instrucciones son restringidas: CIPHER MESSAGE; CIPHER MESSAGE WITH CFB; CIPHER MESSAGE WITH CHAINING; CIPHER MESSAGE WITH COUNTER; CIPHER MESSAGE WITH OFB; COMPRESSION CALL; COMPUTE INTERMEDIATE MESSAGE DIGEST; COMPUTE LAST MESSAGE DIGEST; COMPUTE MESSAGE AUTHENTICATION CODE; CONVERT UNICODE-16 TO UNICODE-32; CONVERT UNICODE-16 TO UNICODE-8; CONVERT UNICODE-32 TO UNICODE-16; CONVERT UNICODE-32 TO UNICODE-8; CONVERT UNICODE-8 TO UNICODE-16; CONVERT UNICODE-8 TO UNICODE-32; PERFORM CRYPTOGRAPHIC COMPUTATION; RUNTIME INSTRUMENTATION OFF; y RUNTIME INSTRUMENTATION ON.

Cada una de las instrucciones de más arriba se implementa actualmente por el coprocesador de hardware, o ha estado en máquinas pasadas y, por lo tanto, se consideran restringidas.

Cuando el control de permiso de modificación AR (A) eficaz es cero, las siguientes instrucciones están restringidas: COPY ACCESS; LOAD ACCESS MULTIPLE; LOAD ADDRESS EXTENDED; y SET ACCESS.

Cada una de las instrucciones de más arriba hace que los contenidos de un registro de acceso se modifiquen. Si el control A en la instrucción TRANSACTION BEGIN es cero, entonces el programa ha indicado de manera explícita que la modificación del registro de acceso no se permitirá.

Cuando el control de operación de permiso de punto flotante (F) eficaz es cero, las instrucciones de punto flotante están restringidas.

En ciertas circunstancias, las siguientes instrucciones pueden estar restringidas: EXTRACT CPU TIME; EXTRACT PSW; STORE CLOCK; STORE CLOCK EXTENDED; y STORE CLOCK FAST.

Cada una de las instrucciones de más arriba está sujeta a un control de interceptación en la descripción de estado de ejecución interpretativa. Si el hipervisor ha establecido el control de interceptación para dichas instrucciones, entonces su ejecución puede prolongarse debido a la implementación del hipervisor; por consiguiente, se consideran restringidas si ocurre una interceptación.

Cuando una transacción no restringida se aborta debido a la ejecución intentada de una instrucción restringida, el código de aborto de transacción en el bloque de diagnóstico de transacción se establece en 11 (instrucción restringida), y el código de condición se establece en 3, excepto por lo siguiente: cuando una transacción no restringida se aborta debido a la ejecución intentada de una instrucción que, de lo contrario, resultará en una excepción de operación privilegiada, es impredecible si el código de aborto se establece en 11 (instrucción restringida) o 4 (interrupción de programa no filtrada que resulta del reconocimiento de la interrupción de programa de operación privilegiada). Cuando una transacción no restringida se aborta debido a la ejecución intentada de PREFETCH DATA (RELATIVE LONG) cuando el código en el campo M_1 es 6 o 7 o STORE CHARACTERS UNDER MASK HIGH cuando el campo M_3 es cero y el código en el campo R_1 es 6 o 7, es impredecible si el código de aborto se establece en 11 (instrucción restringida) o 16 (otra caché). Cuando una transacción no restringida se aborta debido a la ejecución intentada de MONITOR CALL, y tanto una condición de episodio de monitoreo como una condición de excepción de especificación están presentes, es impredecible si el código de aborto se establece en 11 o 4, o, si la interrupción de programa se filtra, 12.

Instrucciones adicionales pueden restringirse en una transacción restringida. Aunque dichas instrucciones no se definen actualmente para estar restringidas en una transacción no restringida, pueden restringirse en ciertas circunstancias en una transacción no restringida en procesadores futuros.

Ciertas instrucciones restringidas pueden permitirse en el modo de ejecución transaccional en futuros procesadores. Por lo tanto, el programa no debe depender de la transacción que se está abortando debido a la ejecución intentada de una instrucción restringida. La instrucción TRANSACTION ABORT debe usarse para hacer, de manera fiable, que una transacción se aborte.

En una transacción no restringida, el programa debe proveer un trayecto de código no transaccional alternativo para alojar una transacción que se aborta debido a una instrucción restringida.

Durante la operación, cuando la profundidad de jerarquización de transacción es cero, la ejecución de la instrucción TRANSACTION BEGIN (TBEGIN) que resulta en el código de condición cero hace que la CPU entre en el modo de ejecución transaccional no restringido. Cuando la profundidad de jerarquización de transacción es cero, la ejecución de la instrucción TRANSACTION BEGIN restringida (TBEGINC) que resulta en el código de condición cero hace que la CPU entre en el modo de ejecución transaccional restringido.

Excepto donde se indique explícitamente lo contrario, todas las reglas que son aplicables a la ejecución no transaccional también son aplicables a la ejecución transaccional. Más abajo se describen características adicionales del procesamiento mientras la CPU está en el modo de ejecución transaccional.

Cuando la CPU está en el modo de ejecución transaccional no restringido, la ejecución de la instrucción TRANSACTION BEGIN que resulta en el código de condición cero hace que la CPU permanezca en el modo de ejecución transaccional no restringido.

Según lo observado por la CPU, las capturas y almacenamientos realizados en el modo de ejecución de transacción no son diferentes de aquellos realizados mientras no se está en el modo de ejecución transaccional. Según lo observado por otras CPU y por el subsistema E/S, todos los accesos de operando de almacenamiento realizados mientras una CPU está en el modo de ejecución transaccional parecen ser un solo acceso concurrente con el bloque. Es decir, los accesos a todos los bytes dentro de media palabra, palabra, doble palabra, o cuádruple palabra se especifican para que parezcan ser concurrentes con el bloque según lo observado por otras CPU y programas E/S (p.ej., canal). Se hace referencia, en la presente sección, a la media palabra, palabra, doble palabra o cuádruple palabra como un bloque. Cuando se especifica que una referencia tipo captura parece ser concurrente dentro de un bloque, no se permite un acceso de almacenamiento al bloque por otra CPU o programa E/S durante el tiempo en el que los bytes contenidos en el bloque se están capturando. Cuando se especifica que una referencia tipo almacenamiento parece ser concurrente dentro de un bloque, no se permite un acceso al bloque, ya sea de captura o almacenamiento, por otra CPU o programa E/S durante el tiempo en el que los bytes dentro del bloque se están almacenando.

Los accesos del almacenamiento para la instrucción y capturas de tabla DAT y ART (Tabla de Registro de Acceso) siguen las reglas no transaccionales.

La CPU abandona el modo de ejecución transaccional normalmente por medio de una instrucción TRANSACTION END que hace que la profundidad de jerarquización de transacción pase a cero, en cuyo caso, la transacción se completa.

Cuando la CPU abandona el modo de ejecución transaccional por medio de la finalización de una instrucción TRANSACTION END, todos los almacenamientos realizados mientras se está en el modo de ejecución transaccional se comprometen; es decir, los almacenamientos parecen ocurrir como una sola operación concurrente con el bloque según lo observado por otras CPU y por el subsistema E/S.

Una transacción puede abortarse, de manera implícita, por una variedad de motivos, o puede abortarse, de manera explícita, por la instrucción TRANSACTION ABORT. Posibles causas a modo de ejemplo de un aborto de transacción, el código de aborto correspondiente, y el código de condición que se coloca en la PSW de aborto de transacción se describen más abajo.

Interrupción Externa: El código de aborto de transacción se establece en 2, y el código de condición en la PSW de aborto de transacción se establece en 2. La PSW de aborto de transacción se almacena como la PSW antigua externa como parte del procesamiento de interrupción externo.

Interrupción de Programa (No Filtrada): Una condición de excepción de programa que resulta en una interrupción (es decir, una condición no filtrada) hace que la transacción se aborte con el código 4. El código de condición en la PSW de aborto de transacción se establece específicamente para el código de interrupción de programa. La PSW de aborto de transacción se almacena como la PSW antigua de programa como parte del procesamiento de interrupción de programa.

Una instrucción que de lo contrario resultaría en una transacción abortada debido a una excepción de operación puede producir resultados alternos: para una transacción no restringida, la transacción puede, en su lugar, abortarse con el código de aborto 11 (instrucción restringida); para una transacción restringida, una interrupción de programa de restricción de transacción puede reconocerse en lugar de la excepción de operación.

Cuando un episodio PER (Registro de Episodios de Programas) se reconoce en conjunto con cualquier otra condición de excepción de programa no filtrada, el código de condición se establece en 3.

Interrupción de Comprobación de Máquina: El código de aborto de transacción se establece en 5, y el código de condición en la PSW de aborto de transacción se establece en 2. La PSW de aborto de transacción se almacena como la PSW antigua de comprobación de máquina como parte del procesamiento de interrupción de comprobación de máquina.

Interrupción E/S: El código de aborto de transacción se establece en 6, y el código de condición en la PSW de aborto de transacción se establece en 2. La PSW de aborto de transacción se almacena como la PSW antigua E/S como parte del procesamiento de interrupción E/S.

Captura de Sobreflujo: Una condición de captura de sobreflujo se detecta cuando la transacción intenta capturar desde más ubicaciones que las que la CPU soporta. El código de aborto de transacción se establece en 7, y el código de condición se establece en 2 o en 3.

Almacenamiento de Sobreflujo: Una condición de almacenamiento de sobreflujo se detecta cuando la transacción intenta almacenar para más ubicaciones que las que la CPU soporta. El código de aborto de transacción se establece en 8, y el código de condición se establece en 2 o en 3.

Permitir que el código de condición sea 2 o 3 en respuesta al aborto de captura o almacenamiento de sobreflujo permite a la CPU indicar situaciones potencialmente reintentables (p.ej., el código de condición 2 indica que la reejecución de la transacción puede ser productiva; mientras que el código de condición 3 no recomienda la reejecución).

Captura de Conflicto: Una condición de captura de conflicto se detecta cuando otra CPU o el subsistema E/S intenta almacenar en una ubicación que se ha capturado de forma transaccional por dicha CPU. El código de aborto de transacción se establece en 9, y el código de condición se establece en 2.

Almacenamiento de Conflicto: Una condición de almacenamiento de conflicto se detecta cuando otra CPU o el subsistema E/S intenta acceder a una ubicación que se ha almacenado durante la ejecución transaccional por dicha CPU. El código de aborto de transacción se establece en 10, y el código de condición se establece en 2.

Instrucción Restringida: Cuando la CPU está en el modo de ejecución transaccional, la ejecución intentada de una instrucción restringida hace que la transacción se aborte. El código de aborto de transacción se establece en 11, y el código de condición se establece en 3.

Cuando la CPU está en el modo de ejecución transaccional restringido, es impredecible si la ejecución intentada de una transacción restringida resulta en una interrupción de programa de restricción de transacción o en un aborto debido a una instrucción restringida. La transacción aún se aborta pero el código de aborto puede indicar cualquier causa.

Condición de Excepción de Programa (Filtrada): Una condición de excepción de programa que no resulta en una interrupción (es decir, una condición filtrada) hace que la transacción se aborte con un código de aborto de transacción de 12. El código de condición se establece en 3.

Profundidad de Jerarquización Superada: La condición de profundidad de jerarquización superada se detecta cuando la profundidad de jerarquización de transacción se encuentra en el valor máximo permisible para la configuración, y una instrucción TRANSACTION BEGIN se ejecuta. La transacción se aborta con un código de aborto de transacción de 13, y el código de condición se establece en 3.

Condición Relacionada con Captura de Caché: Una condición relacionada con ubicaciones de almacenamiento capturadas por la transacción se detecta por los circuitos de caché de la CPU. La transacción se aborta con un código de aborto de transacción de 14, y el código de condición se establece en 2 o en 3.

Condición Relacionada con el Almacenamiento de Caché: Una condición relacionada con ubicaciones de almacenamiento almacenadas por la transacción se detecta por los circuitos de caché de la CPU. La transacción se aborta con un código de aborto de transacción de 15, y el código de condición se establece en 2 o en 3.

Otra Condición de Caché: Otra condición de caché se detecta por los circuitos de caché de la CPU. La transacción se aborta con un código de aborto de transacción de 16, y el código de condición se establece en 2 o en 3.

Durante la ejecución transaccional, si la CPU accede a instrucciones u operandos de almacenamiento mediante el uso de diferentes direcciones lógicas que se mapean a la misma dirección absoluta, depende del modelo si la transacción se aborta. Si la transacción se aborta debido a accesos mediante el uso de diferentes direcciones lógicas mapeadas a la misma dirección absoluta, el código de aborto 14, 15 o 16 se establece, dependiendo de la condición.

Condiciones Diversas: Una condición diversa es cualquier otra condición reconocida por la CPU que hace que la transacción se aborte. El código de aborto de transacción se establece en 255, y el código de condición se establece en 2 o en 3.

Cuando múltiples configuraciones se están ejecutando en la misma máquina (por ejemplo, particiones lógicas o máquinas virtuales), una transacción puede abortarse debido a una comprobación de máquina externa o interrupción E/S que ha ocurrido en una configuración diferente.

Aunque más arriba se proveen ejemplos, otras causas de un aborto de transacción con códigos de aborto y códigos de condición correspondientes pueden proveerse. Por ejemplo, una causa puede ser Reiniciar Interrupción, donde el código de aborto de transacción se establece en 1, y el código de condición en la PSW de aborto de transacción se establece en 2. La PSW de aborto de transacción se almacena como la PSW antigua de reinicio como parte del procesamiento de reinicio. Como un ejemplo adicional, una causa puede ser una condición Supervisor Call, en la cual el código de aborto se establece en 3, y el código de condición en la PSW de aborto de transacción se establece en 3. Otros ejemplos o ejemplos diferentes también son posibles.

Notas:

1. La condición diversa puede resultar de cualquiera de las siguientes:

• Instrucciones como, por ejemplo, en z/Architecture, COMPARE AND REPLACE DAT TABLE ENTRY, COMPARE AND SWAP AND PURGE, INVALIDATE DAT TABLE ENTRY, INVALIDATE PAGE TABLE ENTRY, PERFORM FRAME MANAGEMENT FUNCTION en las cuales el control NQ es cero y el control SK es uno, SET STORAGE KEY EXTENDED en la cual el control NQ es cero, llevadas a cabo por otra CPU en la configuración; el código de condición se establece en 2.

• Una función de operador como, por ejemplo, restablecer, reiniciar o detener, o la orden SIGNAL PROCESSOR equivalente se llevan a cabo en la CPU.

• Cualquier otra condición no enumerada más arriba; el código de condición se establece en 2 o 3.

2. La ubicación en la cual la captura y almacenamiento de conflictos se detectan puede ser en cualquier lugar dentro de la misma línea de caché.

3. Bajo ciertas condiciones, la CPU puede no distinguir entre condiciones de aborto similares. Por ejemplo, la captura o almacenamiento de un sobreflujo puede no ser distinguible de una captura o almacenamiento de conflicto respectivo.

4. La ejecución especulativa de múltiples trayectos de instrucción por la CPU puede resultar en una transacción que se está abortando debido a condiciones de conflicto o sobreflujo, incluso si dichas condiciones no ocurren en la secuencia conceptual. Mientras está en el modo de ejecución transaccional restringido, la CPU puede, de manera temporal, exhibir una ejecución especulativa y, por consiguiente, permitir que la transacción intente completarse sin detectar dichos conflictos o sobreflujos de forma especulativa.

La ejecución de una instrucción TRANSACTION ABORT hace que la transacción se aborte. El código de aborto de transacción se establece a partir de la segunda dirección de operando. El código de condición se establece en 2 o 3, dependiendo de si el bit 63 de la segunda dirección de operando es cero o uno, respectivamente.

La Figura 10 resume códigos de aborto a modo de ejemplo almacenados en un bloque de diagnóstico de transacción, y el código de condición (CC) correspondiente. La descripción en la Figura 10 ilustra una implementación particular. Otras implementaciones y codificaciones de valores son posibles.

En una realización, y según se menciona más arriba, la facilidad transaccional provee tanto transacciones restringidas como transacciones no restringidas, así como el procesamiento asociado a aquellas. Inicialmente, las transacciones restringidas se describen y luego las transacciones no restringidas.

Una transacción restringida se ejecuta en el modo transaccional sin un trayecto de repliegue. Es un modo de procesamiento útil para funciones compactas. En ausencia de interrupciones repetidas o conflictos con otras CPU o el subsistema E/S (a saber, provocados por condiciones que no permitirán que la transacción se complete con éxito), una transacción restringida se completará finalmente; por consiguiente, una rutina de manejador de abortos no se requiere y no se especifica. Por ejemplo, en ausencia de violación de una condición que no puede dirigirse (p.ej., dividir por 0); una condición que no permite que la transacción se complete (p.ej., una interrupción de temporizador que no permite que una instrucción se ejecute; una E/S caliente; etc.); o una violación de una restricción asociada a una transacción restringida, la transacción se completará finalmente.

Una transacción restringida se inicia por una instrucción TRANSACTION BEGIN restringida (TBEGINC) cuando la profundidad de jerarquización de transacción es inicialmente cero. Una transacción restringida está sujeta a las siguientes restricciones, en una realización.

1. La transacción ejecuta no más de 32 instrucciones, no incluidas las transacciones TRANSACTION BEGIN restringida (TBEGINC) y TRANSACTION END.

2. Todas las instrucciones en la transacción se encontrarán dentro de 256 bytes contiguos de almacenamiento, incluidas las instrucciones TRANSACTION BEING restringida (TBEGINC) y TRANSACTION END.

3. Además de las instrucciones restringidas, las siguientes restricciones se aplican a una transacción restringida.

a. Las instrucciones están limitadas a aquellas a las que se hace referencia como Instrucciones Generales, incluidas, por ejemplo, sumar, restar, multiplicar, dividir, desplazar, rotar, etc.

b. Las instrucciones de derivación se limitan a las siguientes (las instrucciones enumeradas son de z/Architecture en un ejemplo):

• BRANCH RELATIVE ON CONDITION en la cual M_1 es diferente de cero y el campo RI_2 contiene un valor positivo.

• BRANCH RELATIVE ON CONDITION LONG en la cual el campo M_1 es diferente de cero, y el campo RI_2 contiene un valor positivo que no provoca desbordamiento de dirección.

• COMPARE AND BRANCH RELATIVE, COMPARE IMMEDIATE AND BRANCH RELATIVE, COMPARE LOGICAL AND BRANCH RELATIVE, y COMPARE LOGICAL IMMEDIATE AND BRANCH RELATIVE en las cuales el campo M_3 es diferente de cero y el campo RI_4 contiene un valor positivo. (Es decir, solo derivaciones hacia adelante con máscaras de derivación diferentes de cero).

c. A excepción de TRANSACTION END e instrucciones que provocan una serialización de operando especificada, las instrucciones que provocan una función de serialización son restringidas.

d. Operaciones de almacenamiento-y-almacenamiento (SS-, por sus siglas en inglés), y operaciones de almacenamiento-y-almacenamiento con una instrucción de código operacional extendida (SSE-, por sus siglas en inglés) son restringidas.

e. Todas las siguientes instrucciones generales (que son de z/Architecture en el presente ejemplo) son restringidas: CHECKSUM; CIPHER MESSAGE; CIPHER MESSAGE WITH CFB; CIPHER MESSAGE WITH CHAINING; CIPHER MESSAGE WITH COUNTER; CIPHER MESSAGE WITH OFB; COMPARE AND FORM CODEWORD; COMPARE LOGICAL LONG; COMPARE LOGICAL LONG EXTENDED; COMPARE LOGICAL LONG UNICODE; COMPARE LOGICAL STRING; COMPARE UNTIL SUBSTRING EQUAL; COMPRESSION CALL; COMPUTE INTERMEDIATE MESSAGE DIGEST; COMPUTE LAST MESSAGE DIGEST; COMPUTE MESSAGE AUTHENTICATION CODE; CONVERT TO BINARY; CONVERT TO DECIMAL; CONVERT UNICODE-16 TO UNICODE-32; CONVERT

UNICODE-16 TO UNICODE-8; CONVERT UNICODE-32 TO UNICODE-16; CONVERT UNICODE-32 TO UNICODE-8; CONVERT UNICODE-8 TO UNICODE-16; CONVERT UNICODE-8 TO UNICODE-32; DIVIDE; DIVIDE LOGICAL; DIVIDE SINGLE; EXECUTE; EXECUTE RELATIVE LONG; EXTRACT CACHE ATTRIBUTE; EXTRACT CPU TIME; EXTRACT PSW; EXTRACT TRANSACTION NESTING DEPTH; LOAD AND ADD; LOAD AND ADD LOGICAL; LOAD AND AND; LOAD AND EXCLUSIVE OR; LOAD AND OR; LOAD PAIR DISJOINT; LOAD PAIR PROM QUADWORD; MONITOR CALL; MOVE LONG; MOVE LONG EXTENDED; MOVE LONG UNICODE; MOVE STRING; NON-TRANSACTIONAL STORE; PERFORM CRYPTOGRAPHIC COMPUTATION; PREFETCH DATA; PREFETCH DATA RELATIVE LONG; RUNTIME INSTRUMENTATION EMIT; RUNTIME INSTRUMENTATION NEXT; RUNTIME INSTRUMENTATION OFF; RUNTIME INSTRUMENTATION ON; SEARCH STRING; SEARCH; STRING UNICODE; SET ADDRESSING MODE; STORE CHARACTERS UNDER MASK HIGH, cuando el campo M_3 es cero, y el código en el campo R_1 es 6 o 7; STORE CLOCK; STORE CLOCK EXTENDED; STORE CLOCK FAST; STORE FACILITY LIST EXTENDED; STORE PAIR TO QUADWORD; TEST ADDRESSING MODE; TRANSACTION ABORT; TRANSACTION BEGIN (tanto TBEGIN como TBEGINC); TRANSLATE AND TEST EXTENDED; TRANSLATE AND TEST REVERSE EXTENDED; TRANSLATE EXTENDED; TRANSLATE ONE TO ONE; TRANSLATE ONE TO TWO TRANSLATE TWO TO ONE; y TRANSLATE TWO TO TWO.

4. Los operandos de almacenamiento de la transacción acceden a no más de cuatro octopalabras. Nota: Se considera que LOAD ON CONDITION y STORE ON CONDITION hacen referencia al almacenamiento independientemente del código de condición. Una octopalabra es, por ejemplo, un grupo de 32 bytes consecutivos en un límite de 32 bytes.

5. La transacción que se ejecuta en la presente CPU, o los almacenamientos por otras CPU o el subsistema E/S, no acceden a operandos de almacenamiento en bloques de 4 K-bytes que contienen los 256 bytes de almacenamiento comenzando con la instrucción TRANSACTION BEGIN restringida (TBEGINC).

6. La transacción no accede a instrucciones u operandos de almacenamiento mediante el uso de diferentes direcciones lógicas que se mapean a la misma dirección absoluta.

7. Las referencias de operandos realizadas por la transacción se encontrarán dentro de una sola palabra doble, excepto que para LOAD ACCESS MULTIPLE, LOAD MULTIPLE, LOAD MULTIPLE HIGH, STORE ACCESS MULTIPLE, STORE MULTIPLE, y STORE MULTIPLE HIGH, las referencias de operandos se encontrarán dentro de una sola octopalabra.

Si una transacción restringida viola cualquiera de las restricciones 1-7, enumeradas más arriba, entonces (a) se reconoce una interrupción de programa de restricción de transacción, o (b) la ejecución procede como si la transacción no estuviera restringida, excepto que violaciones de restricción adicionales pueden incluso resultar en una interrupción de programa restringida de transacción. Es impredecible qué acción se tomará, y la acción tomada puede diferir según qué restricción se viola.

En ausencia de violaciones de restricción, interrupciones repetidas o conflictos con otras CPU o con el subsistema E/S, una transacción restringida finalmente se completará, según se describe más arriba.

1. La posibilidad de completar con éxito una transacción restringida mejora si la transacción cumple con los siguientes criterios:

a. Las instrucciones emitidas son menos que el máximo de 32.

b. Las referencias de operandos de almacenamiento son menos que el máximo de 4 octopalabras.

c. Las referencias de operandos de almacenamiento se encuentran en la misma línea de caché.

d. Las referencias de operandos de almacenamiento a las mismas ubicaciones ocurren en el mismo orden por todas las transacciones.

2. No se asegura necesariamente que una transacción restringida se completará con éxito en su primera ejecución. Sin embargo, si una transacción restringida que no viola ninguna de las restricciones enumeradas se aborta, la CPU emplea circuitos para asegurar que una ejecución repetida de la transacción es exitosa posteriormente.

3. Dentro de una transacción restringida, TRANSACTION BEGIN es una instrucción restringida, por consiguiente, una transacción restringida no puede anidarse.

4. La violación de cualquiera de las restricciones 1-7 de más arriba por una transacción restringida puede resultar en un bucle de programa.

5. Las limitaciones de una transacción restringida son similares a aquellas de un bucle comparar-y-cambiar. Debido a la potencial interferencia de otras CPU y/o del subsistema E/S, no hay garantía arquitectónica de que una instrucción COMPARE AND SWAP se completará alguna vez con el código de condición 0. Una transacción restringida puede sufrir una interferencia similar en la forma de abortos de captura- o almacenamiento- de conflictos o interrupciones calientes.

La CPU emplea algoritmos de equidad para garantizar que, en ausencia de cualquier violación de restricción, una transacción restringida finalmente se completa.

6. Con el fin de determinar el número de iteraciones repetidas requeridas para completar una transacción restringida, el programa puede emplear un contador en un registro general que no está sujeto a la máscara de guardado de registro general. Un ejemplo se muestra más abajo.

	LH1	15,0	Contador reintento cero.
Bucle	TBEGINC	0(0),X 'FE00'	Preservar GRs 0-13
	AHI	15,1	Incrementar contador
	…		
	…	Código de ejecución transaccional restringida	
	…		
	TEND		Fin de transacción.

* R15 ahora contiene el conteo de intentos transaccionales repetidos.

Es preciso notar que ambos registros 14 y 15 no se restablecen en el presente ejemplo. También es preciso notar que, en algunos modelos, el conteo en el registro general 15 puede ser bajo si la CPU detecta la condición de aborto que sigue a la finalización de la instrucción TBEGINC, pero antes de la finalización de la instrucción AHI.

Según lo observado por la CPU, las capturas y los almacenamientos realizados en el modo de ejecución transaccional no son diferentes de aquellos realizados mientras no se está en el modo de ejecución de transacción.

En una realización, el usuario (a saber, el que crea la transacción) selecciona si una transacción se restringirá o no. Una realización de la lógica asociada al procesamiento de transacciones restringidas y, en particular, el procesamiento asociado a una instrucción TBEGINC, se describe con referencia a la Figura 11. La ejecución de la instrucción TBEGINC hace que la CPU entre en el modo de ejecución transaccional restringido o permanezca en el modo de ejecución no restringido. La CPU (a saber, el procesador) que ejecuta TBEGINC lleva a cabo la lógica de la Figura 11.

Con referencia a la Figura 11, según la ejecución de una instrucción TBEGINC, una función de serialización se lleva a cabo, ETAPA 1100. Una función u operación de serialización incluye completar todos los accesos de almacenamiento conceptualmente previos (y, para z/Architecture, a modo de ejemplo, establecimientos relacionados de bits de referencia y bits de cambio) por la CPU, según lo observado por otras CPU y por el subsistema E/S, antes de que ocurran los accesos de almacenamiento conceptualmente subsiguientes (y establecimientos relacionados de bits de referencia y bits de cambio). La serialización afecta la secuencia de todos los accesos CPU al almacenamiento y a las claves de almacenamiento, a excepción de aquellos asociados a la captura de entrada de tabla ART y entrada de tabla DAT.

Según lo observado por una CPU en el modo de ejecución transaccional, la serialización opera normalmente (según se describe más arriba). Según lo observado por otras CPU y por el subsistema E/S, una operación de serialización llevada a cabo mientras una CPU está en el modo de ejecución transaccional ocurre cuando la CPU abandona el modo de ejecución transaccional, ya sea como resultado de una instrucción TRANSACTION END que reduce la profundidad de jerarquización de transacción a cero (finalización normal), o como resultado de que la transacción se está abortando.

Después de llevar a cabo la serialización, se realiza una determinación en cuanto a si una excepción se reconoce, CONSULTA 1102. Si es así, la excepción se maneja, ETAPA 1104. Por ejemplo, una excepción de operación especial se reconoce y la operación se suprime si el control de ejecución transaccional, bit 8 del registro de control 0, es 0. A modo de ejemplos adicionales, una excepción de especificación se reconoce y la operación se suprime, si el campo B_1, bits 16-19 de la instrucción, es diferente de cero; una excepción de ejecución se reconoce y la operación se suprime, si TBEGINC es el objetivo de una instrucción tipo ejecutar; y una excepción de operación se reconoce y la operación se suprime, si la facilidad de ejecución transaccional no se instala en la configuración. Si la CPU ya está en el modo de ejecución de transacción restringido, entonces una excepción de programa de excepción restringida de transacción se reconoce y la operación se suprime. Además, si la profundidad de jerarquización de transacción, cuando se incrementa en 1, supera una profundidad de jerarquización de transacción máxima dependiente del modelo, la transacción se aborta con el código de aborto 13. Otras excepciones o excepciones diferentes pueden reconocerse y manejarse.

Sin embargo, si no hay una excepción, entonces se lleva a cabo una determinación con respecto a si la profundidad de jerarquización de transacción es cero, CONSULTA 1106. Si la profundidad de jerarquización de transacción es cero, entonces se considera que la dirección de bloque de diagnóstico de transacción será inválida, ETAPA 1108; la

PSW de aborto de transacción se establece a partir de los contenidos de la PSW actual, excepto que la dirección de instrucción de la PSW de aborto de transacción designa la instrucción TBEGINC, antes que la siguiente instrucción secuencial, ETAPA 1110; y los contenidos de los pares de registros generales según lo designado por la máscara de guardado de registro general se guardan en una ubicación dependiente del modelo que no es directamente accesible por el programa, ETAPA 1112. Además, la profundidad de jerarquización se establece en 1, ETAPA 1114. Además, el valor eficaz de la operación de permiso de punto flotante (F) y controles de filtrado de interrupción de programa (PIFC) se establecen en cero, ETAPA 1316. Además, el valor eficaz del control de permiso de modificación AR (A), el campo de bit 12 del campo I_2 de la instrucción, se determina, ETAPA 1118. Por ejemplo, el control A eficaz es la AND lógica del control A en la instrucción TBEGINC para el nivel actual y para cualquier instrucción TBEGIN externa.

Volviendo a la CONSULTA 1106, si la profundidad de jerarquización de transacción es mayor que cero, entonces la profundidad de jerarquización se incrementa en 1, ETAPA 1120. Además, el valor eficaz de la operación de permiso de punto flotante (F) se establece en cero, y el valor eficaz del control de filtrado de interrupción de programa (PIFC) no se cambia, ETAPA 1122. El procesamiento entonces continúa con la ETAPA 1118. En una realización, una iniciación exitosa de la transacción resulta en el código de condición 0. Ello concluye una realización de la lógica asociada a la ejecución de una instrucción TBEGINC.

En una realización, la comprobación de excepción provista más arriba puede ocurrir en orden variable. Un orden particular para la comprobación de excepción es de la siguiente manera:

Excepciones con la misma prioridad que la prioridad de condiciones de interrupción de programa para el caso general.

Excepción de especificación debido a que el campo B_1 contiene un valor diferente de cero.

Aborto debido a superación de profundidad de jerarquización de transacción.

Código de condición 0 debido a finalización normal.

De manera adicional, lo siguiente se aplica en una o más realizaciones:

1. Los registros designados para guardarse por la máscara de guardado de registro general solo se restablecen si la transacción se aborta, no cuando la transacción finaliza normalmente por medio de TRANSACTION END. Solo los registros designados por GRSM de la instrucción TRANSACTION BEGIN más externa se restablecen al momento del aborto.

El campo I_2 debe designar todos los pares de registros que proveen valores de entrada que se cambian por una transacción restringida. Por consiguiente, si la transacción se aborta, los valores de registro de entrada se restablecerán a sus contenidos originales cuando la transacción restringida se reejecuta.

2. En la mayoría de los modelos, el rendimiento mejorado puede llevarse a cabo, tanto en TRANSACTION BEGIN como cuando una transacción se aborta, mediante la especificación del número mínimo de registros que necesitan guardarse y restablecerse en la máscara de guardado de registro general.

3. A continuación, se ilustran los resultados de la instrucción TRANSACTION BEGIN (tanto TBEGIN como TBEGINC) según la profundidad de jerarquización de transacción (TND) actual y, cuando la TND es diferente de cero, si la CPU está en el modo de ejecución transaccional no restringido o restringido:

Instrucción	TND=0
TBEGIN	Entrar en el modo de ejecución transaccional no restringido
TBEGINC	Entrar en el modo de ejecución transaccional restringido

Instrucción	TND>0	
TBEGIN	Modo NTX	Modo CTX
	Continuar en el modo de ejecución transaccional no restringido	Excepción de transacción restringida
TBEGINC	Continuar en el modo de ejecución transaccional no restringido	Excepción de transacción restringida

Explicación:

CTX	La CPU está en el modo de ejecución transaccional restringido
NTX	La CPU está en el modo de ejecución transaccional no restringido
TND	Profundidad de jerarquización de transacción al comienzo de la instrucción.

Según se describe en la presente memoria, en un aspecto, se segura la finalización de una transacción restringida, suponiendo que no contiene una condición que la hace incapaz de completarse. Para asegurar que se completa, el procesador (p.ej., CPU) que ejecuta la transacción puede tomar ciertas acciones. Por ejemplo, si una transacción restringida tiene una condición de aborto, la CPU puede temporalmente:

(a) inhibir la ejecución fuera de orden;

(b) evitar que otras CPU accedan a ubicaciones de almacenamiento en conflicto;

(c) inducir retardos aleatorios en el procesamiento de abortos; y/o

(d) invocar otras medidas para facilitar la finalización exitosa.

Para resumir, el procesamiento de una transacción restringida es de la siguiente manera:

• Si ya está en el modo TX restringido, una excepción de transacción restringida se reconoce.

• Si TND (Profundidad de Jerarquización de Transacción) actual > 0, la ejecución procede como si la transacción no restringida

- o Control F eficaz establecido en cero
- o PIFC eficaz no se cambia
- o Permite que TX no restringida externa llame la función de servicio que puede o puede no usar la TX restringida.

• Si TND actual = 0:

o La dirección de bloque de diagnóstico de transacción es inválida

- No hay TDB de instrucción especificada almacenado al momento del aborto

o PSW de aborto de transacción establecida en la dirección de TBEGINC

- No la siguiente instrucción secuencial

o Pares de registros generales designados por GRSM guardados en una ubicación dependiente del modelo no accesible por el programa

o Testigo de transacción opcionalmente formado (a partir del operando D_2). El testigo de transacción es un identificador de la transacción. Puede ser igual a la dirección de operando de almacenamiento u otro valor.

• A eficaz = TBEGINC A & cualquier A externa

• TND aumentada

o Si TND pasa de 0 a 1, la CPU entra en el modo TX restringido

o De lo contrario, la CPU permanece en el modo TX no restringido

• La instrucción se completa con CC0

• Excepciones:

o Excepción de especificación (PIC (Código de Interrupción de Programa) 0006) si el campo B_1 es diferente de cero

o Excepción de operación especial (PIC 0013 hexadecimal) si el control de ejecución de transacción (CR0.8) es cero

o Excepción de restricción de transacción (PIC 0018 hexadecimal) se emite en el modo TX restringido

o Excepción de operación (PIC 0001) si la facilidad de ejecución transaccional restringida no se instala

o Excepción de ejecutar (PIC 0003) si la instrucción es el objetivo de una instrucción de tipo ejecutar

o Código de aborto 13 si la profundidad de jerarquización se ha superado

• Condiciones de aborto en transacción restringida:

o PSW de aborto señala la instrucción TBEGINC

- No la instrucción que la sigue

- La condición de aborto hace que toda la TX se redirija

* No hay trayecto de fallo

o La CPU toma medidas especiales para asegurar la finalización exitosa en la redirección

o Suponiendo que no hay conflicto persistente, interrupción, o violación restringida, se asegura la finalización eventual de la transacción.

• Violación de restricción:

o PIC 0018 hexadecimal - indica la violación de restricción de transacción

o O, la transacción se ejecuta como si no estuviera restringida

Según se describe más arriba, además del procesamiento de transacción restringida, que es opcional, en una realización, la facilidad transaccional también provee procesamiento de transacción no restringida. Detalles adicionales con respecto al procesamiento de transacciones no restringidas y, en particular, el procesamiento asociado a una instrucción TBEGIN se describen con referencia a la Figura 12. La ejecución de la instrucción TBEGIN hace que la CPU entre o permanezca en el modo de ejecución transaccional no restringido. La CPU (a saber, el procesador) que ejecuta TBEGIN lleva a cabo la lógica de la Figura 12.

Con referencia a la Figura 12, según la ejecución de la instrucción TBEGIN, una función de serialización (descrita más arriba) se lleva a cabo, ETAPA 1200. Después de llevar a cabo la serialización, se realiza una determinación en cuanto a si una excepción se reconoce, CONSULTA 1202. Si es así, entonces la excepción se maneja, ETAPA 1204. Por ejemplo, una excepción de operación especial se reconoce y la operación se suprime si el control de ejecución transaccional, bit 8 de registro de control 0, es cero. Además, una excepción de especificación se reconoce y la operación se suprime si el control de filtrado de interrupción de programa, bits 14-15 del campo I_2 de la instrucción, contiene el valor 3; o la primera dirección de operando no designa un límite de palabra doble. Una excepción de operación se reconoce y la operación se suprime, si la facilidad de ejecución transaccional no se instala en la configuración; y una excepción de ejecución se reconoce y la operación se suprime si TBEGIN es el objetivo de una instrucción tipo ejecutar. Además, si la CPU está en el modo de ejecución transaccional restringido, entonces una excepción de programa de excepción restringida de transacción se reconoce y la operación se suprime. Además, si la profundidad de jerarquización de transacción, cuando se incrementa en 1, supera una profundidad de jerarquización de transacción máxima dependiente del modelo, la transacción se aborta con el código de aborto 13.

De manera adicional, cuando el campo B_1 de la instrucción es diferente de cero y la CPU no está en el modo de ejecución transaccional, a saber, la profundidad de jerarquización de transacción es cero, entonces la accesibilidad de almacenamiento al primer operando se determina. Si no puede accederse al primer operando para almacenamientos, entonces una excepción de acceso se reconoce y la operación se anula, suprime o termina, dependiendo de la condición de excepción de acceso específica. Además, cualquier episodio de alternación de almacenamiento PER para el primer operando se reconoce. Cuando el campo B_1 es diferente de cero y la CPU ya está en el modo de ejecución transaccional, es impredecible si la accesibilidad del almacenamiento al primer operando se determina, y los episodios de alteración de almacenamiento PER se detectan para el primer operando. Si el campo B_1 es cero, entonces no se accede al primer operando.

Además de la comprobación de excepción, se lleva a cabo una determinación con respecto a si la CPU está en el modo de ejecución transaccional (a saber, la profundidad de jerarquización de transacción es cero), CONSULTA 1206. Si la CPU no está en el modo de ejecución transaccional, entonces los contenidos de pares de registros generales seleccionados se guardan, ETAPA 1208. En particular, los contenidos de los pares de registros generales designados por la máscara de guardado de registro general se guardan en una ubicación dependiente del modelo que no es directamente accesible por el programa.

Además, se lleva a cabo una determinación sobre si el campo B_1 de la instrucción es cero, CONSULTA 1210. Si el campo B_1 no es igual a cero, la primera dirección de operando se coloca en la dirección de bloque de diagnóstico de transacción, ETAPA 1214, y la dirección de bloque de diagnóstico de transacción es válida. Además, la PSW de aborto de transacción se establece a partir de los contenidos de la PSW actual, ETAPA 1216. La dirección de instrucción de la PSW de aborto de transacción designa la siguiente instrucción secuencial (es decir, la instrucción que sigue a la TBEGIN más externa).

Además, se lleva a cabo una determinación del valor eficaz del control de permiso de modificación AR (A), bit 12 del campo I_2 de la instrucción, ETAPA 1218. El control A eficaz es la AND lógica del control A en la instrucción TBEGIN para el nivel actual y para todos los niveles externos. Además, un valor eficaz del control de permiso de operación de punto flotante (F), bit 13 del campo I_2 de la instrucción, se determina, ETAPA 1220. El control F eficaz es la AND lógica del control F en la instrucción TBEGIN para el nivel actual y para todos los niveles externos. Además, un valor eficaz del control de filtrado de interrupción de programa (PIFC), bits 14-15 del campo I_2 de la instrucción, se determina, ETAPA 1222. El valor PIFC eficaz es el valor más alto en la instrucción TBEGIN para el nivel actual y para todos los niveles externos.

Además, un valor de uno se añade a la profundidad de jerarquización de transacción, ETAPA 1224, y la instrucción se completa con el establecimiento del código de condición 0, ETAPA 1226. Si la profundidad de jerarquización de transacción pasa de cero a uno, la CPU entra en el modo de ejecución transaccional no restringido; de lo contrario, la CPU permanece en el modo de ejecución transaccional no restringido.

Volviendo a la CONSULTA 1210, si B_1 es igual a cero, entonces la dirección de bloque de diagnóstico de transacción es inválida, ETAPA 1211, y el procesamiento continúa con la ETAPA 1218. De manera similar, si la CPU está en el modo de ejecución transaccional, CONSULTA 1206, el procesamiento continúa con la ETAPA 1218.

Código de Condición Resultante de ejecución de TBEGIN incluye, por ejemplo:

0 Iniciación de transacción exitosa

1 --

2 --

3 --

Las Excepciones de Programa incluyen, por ejemplo:

• Acceso (almacenamiento, primer operando)

• Operación (facilidad de ejecución transaccional no instalada)

• Operación especial

• Especificación

• Restricción de transacción (debido a instrucción restringida)

En una realización, la comprobación de excepción provista más arriba puede ocurrir en orden variable. Un orden particular para la comprobación de excepción es de la siguiente manera:

• Excepciones con la misma prioridad que la prioridad de condiciones de interrupción de programa para el caso general.

• Excepción de especificación debido a valor PIFC reservado.

• Excepción de especificación debido a que primera dirección de operando no está en un límite de doble palabra.

• Excepción de acceso (cuando el campo B_1 es diferente de cero).

• Aborto debido a superación de profundidad de jerarquización de transacción máxima.

• Código de condición 0 debido a finalización normal.

Notas:

1. Cuando el campo B_1 es diferente de cero, se aplica lo siguiente:

• Un bloque de diagnóstico de transacción (TDB) accesible se proveerá cuando una transacción más externa se inicie - incluso si la transacción nunca se aborta.

• Dado que es impredecible si la accesibilidad del TDB se prueba para transacciones anidadas, un TDB accesible debe proveerse para cualquier instrucción TBEGIN anidada.

• El rendimiento de cualquier TBEGIN en la cual el campo B_1 es diferente de cero, y el rendimiento de cualquier procesamiento de aborto que ocurre para una transacción que se ha iniciado por una TBEGIN más externa en la cual el campo B_1 es diferente de cero, pueden ser más bajos que cuando el campo B_1 es cero.

2. Los registros designados para guardarse por la máscara de guardado de registro general solo se restablecen, en

una realización, si la transacción se aborta, no cuando la transacción finaliza normalmente por medio de TRANSACTION END. Solo los registros designados por GRSM de la instrucción TRANSACTION BEGIN más externa se restablecen al momento del aborto.

El campo I_2 debe designar todos los pares de registros que proveen valores de entrada que se cambian por la transacción. Por consiguiente, si la transacción se aborta, los valores de registros de entrada se restablecerán a sus contenidos originales cuando el manejador de abortos se ingrese.

3. Se espera que la instrucción TRANSACTION BEGIN (TBEGIN) sea seguida de una instrucción de derivación condicional que determinará si la transacción se ha iniciado de manera exitosa.

4. Si una transacción se aborta debido a condiciones que no resultan en una interrupción, la instrucción designada por la PSW de aborto de transacción recibe control (es decir, la instrucción que sigue a la TRANSACTION BEGIN (TBEGIN) más externa). Además del código de condición establecido por la instrucción TRANSACTION BEGIN (TBEGIN), los códigos de condición 1-3 también se establecen cuando una transacción se aborta.

Por lo tanto, la secuencia de instrucciones que sigue a la instrucción TRANSACTION BEGIN (TBEGIN) más externa debe poder alojar todos los cuatro códigos de condición, aunque la instrucción TBEGIN solo establece el código 0, en el presente ejemplo.

5. En la mayoría de los modelos, el rendimiento mejorado puede llevarse a cabo, tanto en TRANSACTION BEGIN como cuando una transacción se aborta, mediante la especificación del número mínimo de registros que necesitan guardarse y restablecerse en la máscara de guardado de registro general.

6. Mientras está en el modo de ejecución transaccional no restringido, un programa puede llamar una función de servicio que puede alterar los registros de acceso o registros de punto flotante (incluido el registro de control de punto flotante). Aunque dicha rutina de servicio puede guardar los registros alterados en la entrada y restablecerlos en la salida, la transacción puede abortarse antes de la salida normal de la rutina. Si el programa de llamada no provee la preservación de dichos registros mientras la CPU está en el modo de ejecución transaccional no restringido, puede no tener capacidad para tolerar la alteración de la función de servicio de los registros.

Con el fin de prevenir la alteración involuntaria de registros de acceso mientras está en el modo de ejecución transaccional no restringido, el programa puede establecer el control de permiso de modificación AR, bit 12 del campo I_2 de la instrucción TRANSACTION BEGIN, en cero. De manera similar, para evitar la alteración involuntaria de los registros de punto flotante, el programa puede establecer el control de operación de permiso de punto flotante, bit 13 del campo I_2 de la instrucción TBEGIN, en cero.

7. Las condiciones de excepción de programa reconocidas durante la ejecución de la instrucción TRANSACTION BEGIN (TBEGIN) están sujetas al control de filtrado de interrupción de programa eficaz establecido por cualquier instrucción TBEGIN externa. Las condiciones de excepción de programa reconocidas durante la ejecución de la instrucción TBEGIN más externa no están sujetas a filtrado.

8. Con el fin de actualizar múltiples ubicaciones de almacenamiento en una manera serializada, las secuencias de códigos convencionales pueden emplear una palabra de bloqueo (semáforo). Si (a) la ejecución transaccional se usa para implementar actualizaciones de múltiples ubicaciones de almacenamientos, (b) el programa también provee un trayecto de "repliegue" que se invocará si la transacción se aborta, y (c) el trayecto de repliegue emplea una palabra de bloqueo, entonces el trayecto de ejecución transaccional debe también probar la disponibilidad del bloqueo y, si el bloqueo no está disponible, finalizar la transacción por medio de la instrucción TRANSACTION END y derivar al trayecto de repliegue. Ello asegura el acceso coherente a los recursos serializados, independientemente de si están actualizados de forma transaccional.

De manera alternativa, el programa puede abortarse si el bloqueo no está disponible, sin embargo, el procesamiento de aborto puede ser significativamente más lento que simplemente finalizar la transacción mediante TEND.

9. Si el control de filtrado de interrupción de programa (PIFC) eficaz es mayor que cero, la CPU filtra la mayoría de las interrupciones de programa de excepción de datos. Si el control de la operación de permiso de punto flotante (F) eficaz es cero, el código de excepción de datos (DXC) no se establecerá en el registro de control de punto flotante como resultado de un aborto debido a una condición de excepción de programa de excepción de datos. En el presente escenario (el filtrado se aplica y el control F eficaz es cero), la única ubicación en la cual DXC se inspecciona es en el TDB de TBEGIN especificada. Si el manejador de abortos del programa inspecciona el DXC en dicha situación, el registro general B_1 de ser diferente de cero, de modo que una dirección de bloque de diagnóstico de transacción (TDBA) válida se establece.

10. Si una alteración de almacenamiento PER o condición de detección de dirección cero existe para el TDB de TBEGIN especificada de la instrucción TBEGIN más externa, y la supresión de episodio PER no es aplicable, el episodio PER se reconoce durante la ejecución de la instrucción y, por consiguiente, provoca que la transacción se aborte inmediatamente, independientemente de si existe otra condición de aborto.

En una realización, la instrucción TBEGIN establece, de manera implícita, la dirección de aborto de transacción para que sea la siguiente instrucción secuencial que sigue a TBEGIN. Dicha dirección pretende ser una instrucción de derivación condicional que determina si derivar o no dependiendo del código de condición (CC). Una TBEGIN exitosa establece CC0, mientas que una transacción abortada establece CC1, CC2 o CC3.

En una realización, la instrucción TBEGIN provee un operando de almacenamiento opcional que designa la dirección de un bloque de diagnóstico de transacción (TDB) en el cual se almacena información si la transacción se aborta.

Además, provee un operando inmediato que incluye lo siguiente:

Una máscara de guardado de registro general (GRSM) que indica qué pares de registros generales se guardarán al inicio de la ejecución transaccional y se restablecerán si la transacción se aborta;

un bit (A) para permitir el aborto de la transacción si la transacción modifica registros de acceso;

un bit (F) para permitir el aborto de la transacción si la transacción intenta ejecutar instrucciones de punto flotante; y

un control de filtrado de interrupción de programa (PIFC) que permite que los niveles de transacción individuales eviten la presentación real de una interrupción de programa si una transacción se aborta.

Los controles A, F y PIFC pueden ser diferentes en varios niveles de jerarquización y restablecerse al nivel previo cuando los niveles de transacciones internos finalizan.

Además, TBEGIN (o en otra realización, TBEGINC) se usa para formar un testigo de transacción. De manera opcional, el testigo puede combinarse con un testigo formado por la instrucción TEND. Para cada instrucción TBEGIN (o TBEGINC), como un ejemplo, un testigo se forma a partir de la primera dirección de operando. Dicho testigo puede formarse independientemente de si el registro base es cero (a diferencia del establecimiento de dirección TDB que solo ocurre cuando el registro base es diferente de cero). Para cada instrucción TRANSACTION END ejecutada con un registro base diferente de cero, un testigo similar se forma a partir de su operando de almacenamiento. Si los testigos no concuerdan, una excepción de programa puede reconocerse para alertar al programa sobre una instrucción no emparejada.

La concordancia de testigos provee un mecanismo previsto para mejorar la fiabilidad de software asegurando que una declaración TEND se empareja de manera adecuada con una TBEGIN (o TBEGINC). Cuando una instrucción TBEGIN se ejecuta en un nivel de jerarquización particular, un testigo se forma a partir de la dirección de operando de almacenamiento que identifica dicha instancia de una transacción. Cuando una instrucción TEND correspondiente se ejecuta, un testigo se forma a partir de la dirección de operando de almacenamiento de la instrucción, y la CPU compara el testigo inicial para el nivel de jerarquización con el testigo final. Si los testigos no concuerdan, una condición de excepción se reconoce. Un modelo puede implementar la concordancia de testigos para solamente cierto número de niveles de jerarquización (o para niveles de no jerarquización). El testigo puede no implicar todos los bits de la dirección de operando de almacenamiento, o los bits pueden combinarse mediante troceado u otros métodos. Un testigo puede formarse por la instrucción TBEGIN incluso si no se accede a su operando de almacenamiento.

Para resumir, el procesamiento de una transacción no restringida es de la siguiente manera:

• Si TND = 0:

o Si $B_1 \neq 0$, la dirección de bloque de diagnóstico de transacción se establece a partir de la primera dirección de operando.

o PSW de aborto de transacción se establece en la siguiente dirección de instrucción secuencial.

o Pares de registros generales designados por el campo I_2 se guardan en la ubicación dependiente del modelo.

- No es accesible directamente por el programa

• Controles PIFC, A & F eficaces computados

o A eficaz = TBEGIN A & cualquier A externa

o F eficaz = TBEGIN F & cualquier F externa

o PIFC eficaz = máx(TBEGIN PIFC, cualquier PIFC externo)

• Profundidad de jerarquización de transacción (TND) incrementada

• Si TND pasa de 0 a 1, la CPU entra en el modo de ejecución transaccional

• Código de condición establecido en cero

o Cuando la instrucción que sigue a TBEGIN recibe el control:

- el éxito de TBEGIN se indica por CC0

- la transacción abortada se indica por CC diferente de cero

• Excepciones:

o Código de aborto 13 si la profundidad de jerarquización se ha superado

o Excepción de acceso (uno de varios PIC) si el campo B_1 es diferente de cero, y no puede accederse al operando de almacenamiento para una operación de almacenamiento

o Excepción de ejecutar (PIC 0003) si la instrucción TBEGIN es el objetivo de una instrucción tipo ejecutar

o Excepción de operación (PIC 0001) si la facilidad de ejecución transaccional no se instala

o PIC 0006 si

- PIFC es inválido (valor de 3)

- Dirección de segundo operando no está alineada con doble palabra

o PIC 0013 hexadecimal si el control de ejecución transaccional (CR0.8) es cero

o PIC 0018 hexadecimal si se emite en el modo TX restringido

Según se indica más arriba, una transacción, ya sea restringida o no restringida, puede finalizar por una instrucción TRANSACTION END (TEND). Detalles adicionales con respecto al procesamiento de una instrucción de finalizar transacción (TEND) se describen con referencia a la Figura 13. La CPU (a saber, el procesador) que ejecuta TEND lleva a cabo la lógica de la Figura 13.

Con referencia a la Figura 13, inicialmente, según el procesador que obtiene (p.ej., captura, recibe, etc.) la instrucción TEND, se lleva a cabo la comprobación de excepción y si hay una excepción, CONSULTA 1300, entonces la excepción se maneja, ETAPA 1302. Por ejemplo, si la TRANSACTION END es el objetivo de una instrucción tipo ejecutar, la operación se suprime y una excepción de ejecución se reconoce; y una excepción de operación especial se reconoce y la operación se suprime si el control de ejecución transaccional, bit 8 de CR0, es cero. Además, una excepción de operación se reconoce y la operación se suprime, si la facilidad de ejecución transaccional no se instala en la configuración.

Volviendo a CONSULTA 1300, si una excepción no se reconoce, entonces la profundidad de jerarquización de transacción se reduce (p.ej., en uno), ETAPA 1304. Una determinación se lleva a cabo sobre si la profundidad de jerarquización transaccional es cero después de la reducción, CONSULTA 1306. Si la profundidad de jerarquización de transacción es cero, entonces todos los accesos de almacenamiento realizados por la transacción (y otras transacciones dentro del nido de transacciones, si lo hubiera, del cual dicha transacción es parte) se comprometen, ETAPA 1308. Además, la CPU abandona el modo de ejecución transaccional, ETAPA 1310, y la instrucción se completa, ETAPA 1312.

Volviendo a la CONSULTA 1306, si la profundidad de jerarquización de transacción no es igual a cero, entonces la instrucción TRANSACTION END simplemente se completa.

Si la CPU está en el modo de ejecución de transacción al inicio de la operación, el código de condición se establece en 0; de lo contrario, el código de condición se establece en 2.

Se advierte que el control de operación de permiso de punto flotante (F) eficaz, el control de permiso de modificación AR (A) y el control de filtrado de interrupción de programa (PIFC) se restablecen a sus respectivos valores anteriores a la instrucción TRANSACTION BEGIN que ha iniciado el nivel que se está finalizando. Además, una función de serialización se lleva a cabo al final de la operación.

Los episodios de captura de instrucción PER y finalización de transacción que se reconocen al finalizar la instrucción TRANSACTION END más externa no resultan en que se aborte la transacción.

En un ejemplo, la instrucción TEND también incluye un campo base B_2 y un campo de desplazamiento D_2, que se combinan (p.ej., añaden) para crear una segunda dirección de operando. En el presente ejemplo, la concordancia de testigos puede llevarse a cabo. Por ejemplo, cuando B_2 es diferente de cero, los bits seleccionados de la segunda dirección de operando se combinan contra un testigo de transacción formado por la TBEGIN correspondiente. Si hay una discordancia, hay una excepción (p.ej., PIC 0006).

Además, una transacción puede abortarse de manera implícita o explícita por una instrucción TRANSACTION ABORT. El aborto de una transacción por TABORT o de otra manera incluye llevar a cabo un número de etapas. Un

ejemplo de las etapas para el procesamiento de aborto, en general, se describe con referencia a la Figura 14. Si hay una diferencia en el procesamiento según si el aborto se inicia por TABORT o de otra manera, esta se indica en la descripción más abajo. En un ejemplo, un procesador (p.ej., CPU) está llevando a cabo la lógica de la Figura 14.

Con referencia a la Figura 14, inicialmente, según la ejecución de la instrucción TABORT o un aborto implícito, los accesos de almacenamientos no transaccionales realizados mientras la CPU estaba en el modo de ejecución transaccional se comprometen, ETAPA 1400. Otros almacenamientos (p.ej., almacenamientos transaccionales) realizados mientras la CPU estaba en el modo de ejecución transaccional se descartan, ETAPA 1402.

La CPU abandona el modo de ejecución transaccional, ETAPA 1404, y los almacenamientos subsiguientes ocurren de manera no transaccional. La PSW actual se reemplaza por los contenidos de la PSW de aborto de transacción, excepto que el código de condición se establece según se describe más arriba (diferente de la situación de más abajo, en la cual, si TDBA es válida, pero el bloque es inaccesible, entonces CC=1), ETAPA 1406. Como parte de o subsiguiente al procesamiento de aborto, el procesamiento deriva a la ubicación especificada por PSW de aborto de transacción para llevar a cabo una acción. En un ejemplo en el cual la transacción es una transacción restringida, la ubicación es la instrucción TBEGINC y la acción es la reejecución de dicha instrucción; y en un ejemplo adicional en el cual la transacción es una transacción no restringida, la ubicación es la instrucción después de TBEGIN, y la acción es la ejecución de dicha instrucción, que puede ser, por ejemplo, una derivación a un manejador de abortos.

A continuación, se lleva a cabo una determinación sobre si la dirección de bloque de diagnóstico de transacción es válida, CONSULTA 1408. Cuando la dirección de bloque de diagnóstico de transacción es válida, la información de diagnóstico que identifica la razón para el aborto y los contenidos de los registros generales se almacenan en el bloque de diagnóstico de transacción de TBEGIN especificada, ETAPA 1410. Los campos TDB almacenados y las condiciones bajo las cuales se almacenan se describen más arriba con referencia al bloque de diagnóstico de transacción.

Si la dirección de bloque de diagnóstico de transacción es válida, pero el bloque se ha convertido en inaccesible, con posterioridad a la ejecución de la instrucción TBEGIN más externa, no se accede al bloque, y el código de condición 1 se aplica.

Para transacciones que se abortan debido a las condiciones de excepción de programa que resultan en una interrupción, el TDB de interrupción de programa se almacena.

Volviendo a la CONSULTA 1408, si la dirección de bloque de diagnóstico de transacción no es válida, no se almacena el TDB de TBEGIN especificada, y el código de condición 2 o 3 se aplica, dependiendo de la razón para abortar.

Además de lo descrito más arriba, la profundidad de jerarquización de transacción se establece en igual a cero, ETAPA 1412. Además, cualquier par de registros generales designado para guardarse por la instrucción TBEGIN más externa se restablece, ETAPA 1414. Los pares de registros generales que no se han designado para guardarse por la instrucción TBEGIN más externa no se restablecen cuando una transacción se aborta.

Además, una función de serialización se lleva a cabo, ETAPA 1416. Una función de serialización u operación incluye completar todos los accesos de almacenamiento conceptualmente previos (y, para z/Architecture, a modo de ejemplo, establecimientos relacionadas de bits de referencia y bits de cambio) por la CPU, según lo observado por otras CPU y por el subsistema E/S, antes de que ocurran los accesos de almacenamiento conceptualmente subsiguientes (y establecimientos relacionadas de bits de referencia y bits de cambio). La serialización efectúa la secuencia de todos los accesos CPU al almacenamiento y a las claves de almacenamiento, a excepción de aquellos asociados a la captura de entrada de tabla ART y entrada de tabla DAT.

Según lo observado por una CPU en el modo de ejecución transaccional, la serialización opera normalmente (según se describe más arriba). Según lo observado por otras CPU y por el subsistema E/S, una operación de serialización llevada a cabo mientras una CPU está en el modo de ejecución transaccional ocurre cuando la CPU abandona el modo de ejecución transaccional, ya sea como resultado de una instrucción TRANSACTION END que reduce la profundidad de jerarquización de transacción a cero (finalización normal), o como resultado de que la transacción se está abortando.

Para el procesamiento de aborto iniciado de manera diferente que por TABORT, si la transacción se aborta debido a una condición de excepción que resulta en una interrupción, CONSULTA 1418, los códigos de interrupción o parámetros asociados a la interrupción se almacenan en las ubicaciones de almacenamiento asignadas correspondientes al tipo de interrupción, ETAPA 1420. Además, la PSW actual, según se establece más arriba, se almacena en la PSW antigua de interrupción, ETAPA 1422. De allí en adelante, o si la transacción no se ha abortado debido a una condición de excepción que ha resultado en una interrupción, la instrucción finaliza con el código de condición cero.

Además de lo descrito más arriba, en un ejemplo para la ejecución interpretativa de z/Architecture, cuando la CPU está en el modo de ejecución transaccional, y ocurre una condición de invitado que normalmente resultaría en los códigos de interceptación 4, 12, 44, 56, 64, 68 o 72, la interceptación no ocurre. Más bien, la CPU permanece en el

modo de ejecución interpretativo, y las condiciones de aborto se indican al invitado de la siguiente manera:

• Para una transacción no restringida, la transacción se aborta debido a una instrucción restringida (código de aborto 11). Si un episodio PER concurrente se ha detectado y la CPU está habilitada para PER, ocurre una interrupción de programa con el código de interrupción 0280 hexadecimal.

• Para una transacción restringida, una excepción de restricción de transacción se reconoce. Si un episodio PER concurrente se ha detectado y la CPU está habilitada para PER, ocurre una interrupción de programa con el código de interrupción 0298 hexadecimal.

Cuando una transacción se aborta debido a una condición de excepción de programa, el filtrado de interrupción de programa puede inhibir la presentación de una interrupción. Para interrupciones de programa que pueden resultar en interceptación, el filtrado también inhibe la interceptación.

En una realización, durante la ejecución transaccional, las condiciones de excepción, incluidas las excepciones de programa, hacen que una transacción se aborte y que una interrupción se presente. Sin embargo, según un aspecto, una transacción como, por ejemplo, una transacción no restringida, puede filtrar interrupciones de programa; es decir, transacciones que se abortan debido a ciertas condiciones de excepción de programa no tienen necesariamente que resultar en una interrupción.

En su lugar, el programa puede, de manera opcional, especificar que ciertas condiciones de excepción de programa no resultan en una interrupción. Ello facilita la ejecución especulativa y, por consiguiente, permite que el programa se recupere de ciertas excepciones de programa sin la sobrecarga costosa de establecer rutinas de recuperación.

En un ejemplo, el filtrado de interrupción de programa está sujeto a los siguientes controles:

• Anulación del filtrado de interrupción de programa de ejecución transaccional, bit 9 de registro de control 0. Dicho control se establece por el sistema operativo, ya sea permitiendo o bloqueando el filtrado por transacciones.

• El control de filtrado de interrupción de programa (PIFC) eficaz en la instrucción TRANSACTION BEGIN.

• La condición de excepción que ha ocurrido.

Cuando la anulación del filtrado de interrupción de programa de ejecución transaccional es cero, el programa especifica qué clases de condiciones de excepción se filtrarán por medio del control de filtrado de interrupción de programa (PIFC), bits 14-15 del campo I_2 de la instrucción TRANSACTION BEGIN (TBEGIN). El PIFC eficaz es el valor más alto del PIFC en la instrucción TBEGIN para el nivel de jerarquización actual y para todos los niveles externos.

Para la mayoría de las condiciones de excepción de programa, hay una clase de ejecución transaccional correspondiente definida para la condición de excepción (es preciso ver las Figuras 16A-16B) y guardada dentro del procesador. El PIFC eficaz y las clases de ejecución transaccional interactúan de la siguiente manera:

PIFC eficaz	Resultados basados en la Clase de Ejecución Transaccional
0	No ocurre ningún filtrado de interrupción de programa. Las condiciones de excepción que tienen las clases 1, 2 o 3 siempre resultan en una interrupción.
1	Ocurre un filtrado de interrupción de programa limitado. Las condiciones de excepción que tienen las clases 1 o 2 resultan en una interrupción; las condiciones que tienen la clase 3 no resultan en una interrupción.
2	Ocurre un filtrado de interrupción de programa moderado. Solo las condiciones de excepción que tienen la clase 1 resultan en una interrupción; las condiciones que tienen las clases 2 o 3 no resultan en una interrupción.

La instrucción TRANSACTION BEGIN (TBEGINC) no provee un control de filtrado de interrupción de programa explícito; un PIFC implícito de cero se supone para TBEGINC. Por consiguiente, la CPU entra en el modo de ejecución transaccional restringido como resultado de TBEGINC, el PIFC eficaz es cero; cuando la CPU permanece en el modo de ejecución transaccional no restringido como resultado de TBEGINC, el PIFC eficaz no se cambia.

Un ejemplo de un resumen de las relaciones del PIFC eficaz, el tipo de filtrado de interrupción de programa, la clase de ejecución transaccional, y si la condición de excepción resulta en una interrupción se proveen en la Figura 15. En una realización, la instrucción TRANSACTION BEGIN (TBEGIN) se define para reconocer una excepción de especificación si PIFC=3. Sin embargo, en una realización adicional, ello puede modificarse para permitir el filtrado de interrupción de programa total cuando PIFC=3.

Un ejemplo de una lista de condiciones de excepción de programa, la clase de ejecución transaccional correspondiente, y el código de condición que se establece cuando la transacción se aborta debido a una condición

de excepción de programa se representan en las Figuras 16A-16B.

Detalles adicionales relacionados con el filtrado de interrupción de programa se describen con referencia a las Figuras 17A-17B, que representan una realización de lógica de procesamiento asociada al filtrado de interrupción de programa. En un ejemplo, un procesador lleva a cabo dicha lógica.

Con referencia, inicialmente, a la Figura 17A, en una realización, una o más transacciones iniciadas por una o más instrucciones TRANSACTION BEGIN se están ejecutando. Durante el procesamiento transaccional, una condición de excepción de programa se reconoce dentro de una de las transacciones, ETAPA 1700. De allí en adelante, se lleva a cabo una determinación sobre si una interrupción causada por la condición de excepción de programa se presentará, CONSULTA 1702, según se describe en mayor detalle más abajo con referencia a la Figura 17B.

Cuando una condición de excepción de programa no resulta en una interrupción (es decir, la condición se filtra), CONSULTA 1702, la PSW nueva de programa no se carga, ETAPA 1704. Además, ninguna de las ubicaciones de almacenamiento asignadas asociadas a una interrupción de programa se almacena, ETAPA 1706. Dichas ubicaciones incluyen, por ejemplo, la identificación de interrupción de programa, la dirección de episodio de interrupción, la PSW antigua de programa y, cuando corresponda, el código de excepción de datos, código PER, dirección PER, identificación de acceso de excepción, identificación de acceso PER, identificación de acceso de operando, e identificación de excepción de traducción.

Cuando una condición de excepción de programa resulta en una interrupción, CONSULTA 1702, o cuando una condición de excepción de programa resulta en el abandono del modo de ejecución interpretativo debido a la interceptación (es decir, la condición no se filtra), la mayoría de las ubicaciones de almacenamiento asignadas asociadas a una interrupción de programa se almacenan como es normal, ETAPA 1708; sin embargo, el código de longitud de instrucción en los bits 13-14 de la identificación de interrupción de programa es con respecto a la instrucción en la cual la condición de excepción se ha detectado, ETAPA 1710. Además, la PSW de aborto de transacción se almacena como la PSW antigua de programa, ETAPA 1712.

En un ejemplo, con referencia a la Figura 17B, para determinar si la interrupción se presentará, se realiza una determinación sobre si la anulación de filtrado de interrupción de programa, bit 9 de registro de control 0, se establece para permitir el filtrado, CONSULTA 1750. Dicho control se establece por el sistema operativo para permitir o bloquear el filtrado por transacciones. Si el filtrado se bloquea, entonces la interrupción se presenta, ETAPA 1752. De lo contrario, si el sistema operativo permite el filtrado, se realiza una determinación con respecto al valor del PIFC eficaz, CONSULTA 1754. Es decir, cuál es el valor más alto del PIFC del nivel de jerarquización actual y cualquier nivel externo, si lo hubiera. Si el valor del PIFC eficaz es cero, CONSULTA 1752, entonces la interrupción se presenta, ETAPA 1752.

Si el valor del PIFC eficaz es diferente de cero, entonces se lleva a cabo una comprobación sobre la clase transaccional de la condición de excepción de programa con respecto al PIFC, ETAPA 1756. Por ejemplo, para un PIFC eficaz=1, si la clase de la condición de excepción de programa es 1 o 2, se presenta una interrupción, pero no para la clase 3; y para un PIFC eficaz=2, se presenta una interrupción para la clase 1, y no para las clases 2 o 3. (En otra realización, los bloques de decisión 1754 y 1756 pueden condensarse en una sola decisión mediante comparación del PIFC con respecto al valor inverso de la clase de excepción de transacción (TXC). Por ejemplo, si (PIFC > (3-TXC)), entonces la interrupción se filtra, o si (PIFC < = (3-TXC)), entonces se interrumpe).

Si dicha comprobación indica que una interrupción se presentará, CONSULTA 1758, la interrupción se presenta (a saber, una interrupción se presenta al procesador, y el sistema operativo toma el control), ETAPA 1752. De lo contrario, se saltea (a saber, no se presenta interrupción alguna al procesador), ETAPA 1760. Ello concluye una realización de la lógica de filtrado.

Además de lo descrito más arriba, en una realización, cuando un episodio PER se reconoce en conjunto con cualquier otra condición de excepción de programa filtrada, lo siguiente se aplica:

• La clase de transacción y el código de condición para el episodio PER se aplican. En el presente caso, la condición de excepción PER no se filtra, y el código de condición se establece en 3.

• El código de interrupción de programa en el área de prefijo no incluye la condición de excepción no PER, ni se establecen otros parámetros de interrupción de programa no PER en el área de prefijo.

Cuando la anulación de filtrado de interrupción de programa de ejecución transaccional (bit 9 de registro de control 0) es uno, las condiciones de interrupción de programa no están sujetas al filtrado de interrupción de programa. En el presente caso, la ejecución procede como si el PIFC eficaz fuera cero.

Las condiciones de excepción de acceso reconocidas durante la captura de una instrucción no están sujetas al filtrado de interrupción de programa. En dichos casos, la condición de excepción resulta tanto en que la transacción se aborta y en una interrupción de programa.

Además del código de interrupción de programa que representa la causa de la interrupción, el bit 6 del código de

interrupción de programa en las ubicaciones reales 142-143 se establece en uno, lo cual indica que la interrupción de programa ha ocurrido durante la ejecución transaccional. Cuando una transacción de invitado se aborta debido a una interrupción de programa anfitrión, los bits 5 y 6 del código de interrupción de programa anfitrión se establecen en uno. La PSW actual se reemplaza por los contenidos de la PSW de aborto de transacción excepto que el código de condición se establece según las Figuras 10 y 16A-16B.

El filtrado se aplica tanto a condiciones de excepción de programa de invitado como anfitrión. Si una condición de excepción de programa de invitado se filtra, no ocurren condiciones de interceptación asociadas a la interrupción.

En una realización:

1. Una instrucción MONITOR CALL que de otra manera provocaría un episodio de monitoreo es una instrucción restringida. Por lo tanto, una interrupción de programa de episodio de monitoreo no ocurre mientras la CPU está en el modo de ejecución transaccional, por consiguiente, el código de monitoreo en las ubicaciones reales 176-183 no se almacena cuando la transacción se aborta.

De manera similar, cualquier otra condición de excepción de programa se enumera en las Figuras 16A-16B que tienen una clase de transacción no aplicable (-) y el código de condición no puede ocurrir dado que las instrucciones que provocan dichas excepciones están restringidas. Por consiguiente, ni la identificación de interrupción de programa ni otra información de interrupción de programa secundaria se almacenan en ubicaciones reales en el área de prefijo.

2. El filtrado de interrupción de programa puede ser útil en programas que, por varios motivos, posponen la validación de datos - a veces llamada ejecución especulativa. En lugar de establecer un entorno de recuperación potencialmente complicado, el programa simplemente ejecuta transacciones no restringidas en las cuales el control de filtrado de interrupción de programa (PIFC) eficaz es diferente de cero. Ello permite a la rutina de manejador de abortos del programa recibir control para ciertos tipos de condiciones de excepción de programa directamente - sin intervención del sistema operativo.

Un PIFC eficaz de 1 indica que el filtrado limitado se llevará a cabo por la CPU; ello puede ser útil en el reconocimiento de datos inesperados o excepciones aritméticas. Un PIFC eficaz de 2 indica que el filtrado moderado se llevará a cabo; ello puede ser útil en el reconocimiento de ubicaciones de almacenamiento inaccesibles.

Sin embargo, debe notarse que, si un programa se aborta debido a varios tipos de excepciones de acceso filtradas, ello no indica necesariamente que la ubicación será inaccesible si el programa ha intentado la ejecución no transaccional. Por ejemplo, el programa puede especificar un PIFC de 2, y posteriormente abortarse debido a una excepción de traducción de páginas. Dicha excepción puede indicar que la ubicación de almacenamiento no es parte del espacio de dirección virtual, o puede simplemente indicar que el bloque de almacenamiento se ha cancelado.

Más arriba se provee una realización para gestionar interrupciones en un entorno transaccional, en el cual las condiciones de excepción de programa que de otra manera proveerían una interrupción se filtran, de modo que una interrupción se evita.

Además, más arriba se provee un medio eficaz para actualizar múltiples objetos no contiguos en la memoria sin serialización clásica (de grano grueso) como, por ejemplo, bloqueo, que provee un potencial para la mejora de rendimiento de multiprocesador significativa. Es decir, múltiples objetos no contiguos se actualizan sin la ejecución de ordenación de acceso de almacenamiento de grano más grueso que se provee por las técnicas clásicas como, por ejemplo, bloqueos y semáforos. La ejecución especulativa se provee sin establecimiento de recuperación oneroso, y las transacciones restringidas se ofrecen para actualizaciones simples de huella pequeña.

La ejecución transaccional puede usarse en una variedad de escenarios, incluidos, pero sin limitación a ellos, expansión en línea parcial, procesamiento especulativo y elisión de bloqueo. En la expansión en línea parcial, la región parcial que se incluirá en el trayecto ejecutado se envuelve en TBEGIN/TEND. TABORT puede incluirse allí para reanudar el estado en la salida lateral. Para la especulación como, por ejemplo, en Java, las comprobaciones nulas en punteros desreferenciados pueden retrasarse para enlazar el borde mediante el uso de una transacción. Si el puntero es nulo, la transacción puede abortarse de manera segura mediante el uso de TABORT, que se incluye dentro de TBEGIN/TEND.

En cuanto a la elisión de bloqueo, un ejemplo de su uso se describe con referencia a las Figuras 18A-18B y el fragmento de código provisto más abajo.

La Figura 18A representa una lista doblemente enlazada 1800 de múltiples elementos de cola 1802a-1802d. Un nuevo elemento de cola 1802e se insertará en la lista doblemente enlazada de elementos de cola 1800. Cada elemento de cola 1802a-1802e incluye un puntero delantero 1804a- 1804e y un puntero trasero 1806a-1806e. Como se muestra en la Figura 18B, para añadir el elemento de cola 1802e entre los elementos de cola 1802b y 1802c, (1) el puntero trasero 1806e se establece para señalar el elemento de cola 1802b, (2) el puntero delantero 1804e se establece para señalar el elemento de cola 1802c, (3) el puntero trasero 1806c se establece para señalar el elemento de cola 1802e, y (4) el puntero delantero 1804b se establece para señalar el elemento de cola 1802e.

Un fragmento de código a modo de ejemplo correspondiente a las Figuras 18A-18B se describe más abajo:

* R1 - dirección del nuevo elemento de cola que se insertará.

* R2 - dirección del punto de inserción; el nuevo elemento se inserta antes del elemento señalado por R2.

NUEVO	QUE USA	QEL, R1	
ACTUAL	QUE USA	QEL, R2	
	LHI	R15, 10	Cargar cont. De reintento
BUCLE	TBEGIN	TDB, X'C000'	Iniciar transacción (guardar GRs 0-3)
	JNZ	ABORTADO	CC diferente de cero significa abortado.
	LG	R3, ACT.TRA	Señalar elemento previo.
PREV	QUE USA	QEL, R3	Se vuelve direccionable.
	STG	R1, PREV. DEL.	Actualizar ptr. delantero prev.
	STG	R1, ACT. TRA	Actualizar ptr. trasero act.
	STG	R2, NUEVO.DEL	Actualizar ptr. delantero nuevo
	STG	R3, NUEVO.TRA	Actualizar ptr. nuevo trasero
	TEND	Finalizar transacción.	
	. . .		
ABORTADO	JO	NO_REINTENTAR	CC3: Aborto no reintentable.
	JCT	R15, BUCLE	Reintentar transacción algunas veces.
	J	NO_REINTENTAR	No hay mando después de 1Ox; hacerlo de la manera difícil.

En un ejemplo, si la transacción se usa para la elisión de bloqueo, pero el trayecto de repliegue usa un bloqueo, la transacción al menos capturará la palabra de bloqueo para ver que está disponible. El procesador asegura que la transacción se aborta, si otra CPU accede al bloque de manera no transaccional.

Según su uso en la presente memoria, almacenamiento, almacenamiento central, almacenamiento principal, memoria y memoria principal se usan de manera intercambiable, a menos que se establezca lo contrario, de manera implícita por el uso o de manera explícita. Además, mientras que, en una realización, el retardo eficaz de transacción incluye retrasar el compromiso de almacenamientos transaccionales a la memoria principal hasta la finalización de una transacción seleccionada, en otra realización, un retardo eficaz de transacción incluye permitir actualizaciones transaccionales para la memoria, pero manteniendo los valores antiguos y restableciendo la memoria a los valores antiguos al momento del aborto.

Según apreciará una persona con experiencia en la técnica, uno o más aspectos pueden realizarse como un sistema, método o producto de programa de ordenador. Por consiguiente, uno o más aspectos pueden tomar la forma de una realización totalmente de hardware, una realización totalmente de software (que incluye firmware, software residente, microcódigo, etc.) o una realización que combina aspectos de software y hardware a los que se puede, en general, hacer referencia, a todos, en la presente memoria, como un "circuito", "módulo" o "sistema". Además, una o más realizaciones pueden tomar la forma de un producto de programa de ordenador realizado en uno o más medios legibles por ordenador que tienen un código de programa legible por ordenador allí realizado.

Cualquier combinación de uno o más medios legibles por ordenador puede utilizarse. El medio legible por ordenador puede ser un medio de almacenamiento legible por ordenador. Un medio de almacenamiento legible por ordenador puede ser, por ejemplo, pero sin limitación a ello, un sistema, aparato o dispositivo electrónico, magnético, óptico, electromagnético, infrarrojo o semiconductor, o cualquier combinación apropiada de los anteriores. Ejemplos más específicos (una lista no exhaustiva) del medio de almacenamiento legible por ordenador incluyen lo siguiente: una conexión eléctrica que tiene uno o más cables, un disquete de ordenador portátil, un disco duro, una memoria de acceso aleatorio (RAM, por sus siglas en inglés), una memoria de solo lectura (ROM, por sus siglas en inglés), una memoria de solo lectura programable borrable (EPROM, por sus siglas en inglés, o memoria Flash), una fibra óptica, una memoria de solo lectura de disco compacto portátil (CD-ROM, por sus siglas en inglés), un dispositivo de

almacenamiento óptico, un dispositivo de almacenamiento magnético, o cualquier combinación apropiada de los anteriores. En el contexto del presente documento, un medio de almacenamiento legible por ordenador puede ser cualquier medio tangible que pueda contener o almacenar un programa para su uso por o en conexión con un sistema, aparato o dispositivo de ejecución de instrucciones.

Con referencia, ahora, a la Figura 19, en un ejemplo, un producto de programa de ordenador 1900 incluye, por ejemplo, uno o más medios de almacenamiento legibles por ordenador no transitorios 1902 para almacenar medios o lógica de código de programa legibles por ordenador 1904 para proveer y facilitar una o más realizaciones.

El código de programa realizado en un medio legible por ordenador puede transmitirse mediante el uso de un medio apropiado, incluidos, pero sin limitación a ello, inalámbrico, línea alámbrica, cable de fibra óptica, RF, etc., o cualquier combinación apropiada de los anteriores.

El código de programa de ordenador para llevar a cabo operaciones para una o más realizaciones pueden escribirse en cualquier combinación de uno o más lenguajes de programación, incluidos un lenguaje de programación orientado a objetos como, por ejemplo, Java, Smalltalk, C++ o similares, y lenguajes de programación de procedimiento convencionales como, por ejemplo, el lenguaje de programación "C", ensamblador o lenguajes de programación similares. El código de programa puede ejecutarse totalmente en el ordenador del usuario, parcialmente en el ordenador del usuario, como un paquete de software independiente, parcialmente en el ordenador del usuario y parcialmente en un ordenador remoto o totalmente en el ordenador o servidor remoto. En el último escenario, el ordenador remoto puede conectarse al ordenador del usuario a través de cualquier tipo de red, incluida una red de área local (LAN, por sus siglas en inglés) o una red de área amplia (WAN, por sus siglas en inglés), o la conexión puede realizarse a un ordenador externo (por ejemplo, a través de Internet mediante el uso de un Proveedor de Servicios de Internet).

Una o más realizaciones se describen en la presente memoria con referencia a ilustraciones de diagrama de flujo y/o diagramas de bloques de métodos, aparatos (sistemas) y productos de programa de ordenador. Se comprenderá que cada bloque de las ilustraciones de diagrama de flujo y/o diagramas de bloques, y combinaciones de bloques en las ilustraciones de diagrama de flujo y/o diagramas de bloques, pueden implementarse por instrucciones de programa de ordenador. Dichas instrucciones de programa de ordenador pueden proveerse a un procesador de un ordenador de propósito general, ordenador de propósito especial, u otro aparato de procesamiento de datos programable para producir una máquina, de modo que las instrucciones, que se ejecutan mediante el procesador del ordenador u otro aparato de procesamiento de datos programable, crean medios para implementar las funciones/actos especificados en el bloque o bloques del diagrama de flujo y/o del diagrama de bloques.

Dichas instrucciones de programa de ordenador pueden también almacenarse en un medio legible por ordenador que puede dirigir un ordenador, otros aparatos de procesamiento de datos programables, u otros dispositivos para que funcionen en una manera particular, de modo que las instrucciones almacenadas en el medio legible por ordenador producen un artículo de fabricación que incluye instrucciones que implementan la función/acto especificado en el bloque o bloques del diagrama de flujo y/o del diagrama de bloques.

Las instrucciones del programa de ordenador pueden también cargarse a un ordenador, otros aparatos de procesamiento de datos programables, u otros dispositivos para hacer que una serie de etapas operacionales se lleven a cabo en el ordenador, que otros aparatos programables u otros dispositivos produzcan un proceso implementado por ordenador de modo que las instrucciones que se ejecutan en el ordenador u otro aparato programable provean procesos para implementar las funciones/actos especificados en el bloque o bloques del diagrama de flujo y/o del diagrama de bloques.

El diagrama de flujo y diagramas de bloques en las figuras ilustran la arquitectura, funcionalidad y operación de posibles implementaciones de sistemas, métodos y productos de programa de ordenador según varias realizaciones. En este aspecto, cada bloque en el diagrama de flujo o diagramas de bloques puede representar un módulo, segmento o porción de código, que comprende una o más instrucciones ejecutables para implementar las funciones lógicas especificadas. Debe notarse que, en algunas implementaciones alternativas, las funciones advertidas en el bloque pueden ocurrir fuera del orden establecido en las figuras. Por ejemplo, dos bloques que se muestran en sucesión pueden, de hecho, ejecutarse de manera sustancialmente concurrente, o los bloques pueden, a veces, ejecutarse en el orden inverso, dependiendo de la funcionalidad implicada. También se notará que cada bloque de los diagramas de bloques y/o ilustración de diagrama de flujo, y combinaciones de bloques en los diagramas de bloques y/o ilustración de diagrama de flujo, puede implementarse por sistemas basados en hardware de propósito especial que llevan a cabo las funciones o actos especificados, o combinaciones de hardware de propósito especial e instrucciones de ordenador.

Además de lo descrito más arriba, uno o más aspectos pueden proveerse, ofrecerse, desplegarse, gestionarse, servirse, etc., por un proveedor de servicios que ofrece la gestión de entornos de cliente. Por ejemplo, el proveedor de servicios puede crear, mantener, soportar, etc., el código de ordenador y/o infraestructura de ordenador que llevan a cabo uno o más aspectos para uno o más clientes. Como contraprestación, el proveedor de servicios puede recibir un pago del cliente según un abono y/o acuerdo de honorarios, como ejemplos. De manera adicional o alternativa, el proveedor de servicios puede recibir el pago de la venta de contenido de publicidad a uno o más

terceros.

En un aspecto, una aplicación puede desplegarse para llevar a cabo una o más realizaciones. Como un ejemplo, el despliegue de una aplicación comprende proveer infraestructura de ordenador utilizable para llevar a cabo una o más realizaciones.

Como un aspecto adicional, puede desplegarse una infraestructura de ordenador que comprende integrar el código legible por ordenador en un sistema informático, en el cual el código en combinación con el sistema informático puede llevar a cabo una o más realizaciones.

Como un aspecto incluso adicional, puede proveerse un proceso para integrar infraestructura informática que comprende integrar el código legible por ordenador a un sistema informático. El sistema informático comprende un medio legible por ordenador, en el cual el medio de ordenador comprende una o más realizaciones. El código en combinación con el sistema informático puede llevar a cabo una o más realizaciones.

Aunque varias realizaciones se describen más arriba, estas solo son ejemplos. Por ejemplo, entornos informáticos de otras arquitecturas pueden usarse para incorporar y usar una o más realizaciones. Además, diferentes instrucciones, formatos de instrucciones, campos de instrucciones y/o valores de instrucciones pueden usarse. Además, valores de filtrado y/o niveles de filtrado diferentes, otros y/o adicionales pueden proveerse/usarse. Son posibles muchas variaciones.

Además, otros tipos de entornos informáticos pueden ser beneficiosos y usarse. A modo de ejemplo, es utilizable un sistema de procesamiento de datos apropiado para almacenar y/o ejecutar el código de programa que incluye al menos dos procesadores acoplados, directa o indirectamente, a elementos de memoria a través de un bus de sistema. Los elementos de memoria incluyen, por ejemplo, memoria local empleada durante la ejecución real del código de programa, almacenamiento a granel, y memoria caché que proveen almacenamiento temporal de al menos algún código de programa con el fin de reducir la cantidad de veces en las que el código debe recuperarse del almacenamiento a granel durante la ejecución.

Dispositivos de Entrada/Salida o E/S (incluidos, pero sin limitación a ello, teclados, visualizaciones, dispositivos de enfoque, DASD, cinta, CD, DVD, memorias miniatura y otros medios de memoria, etc.) pueden acoplarse al sistema ya sea de manera directa o a través de controladores E/S intervinientes. Los adaptadores de red pueden también acoplarse al sistema para permitir que el sistema de procesamiento de datos se acople a otros sistemas de procesamiento de datos o impresoras remotas o dispositivos de almacenamiento a través de redes privadas o públicas intervinientes. Los módems, módems de cable y tarjetas de Ethernet son solo algunos de los tipos disponibles de adaptadores de red.

Con referencia a la Figura 20, se ilustran componentes representativos de un sistema de Ordenador Anfitrión 5000 para implementar una o más realizaciones. El ordenador anfitrión 5000 representativo comprende una o más CPU 5001 en comunicación con la memoria de ordenador (a saber, almacenamiento central) 5002, así como interfaces E/S a dispositivos de medios de almacenamiento 5011 y redes 5010 para comunicarse con otros ordenadores o SAN y similares. La CPU 5001 cumple con una arquitectura que tiene un conjunto de instrucciones diseñadas y funcionalidad diseñada. La CPU 5001 puede tener traducción de registro de acceso (ART) 5012, que incluye una memoria intermedia de ART lateral (ALB, por sus siglas en inglés) 5013, para seleccionar un espacio de dirección que se usará por la traducción dinámica de direcciones (DAT) 5003 para transformar direcciones de programas (direcciones virtuales) en direcciones de memoria reales. Una DAT normalmente incluye una memoria intermedia de traducción lateral (TLB, por sus siglas en inglés) 5007 para almacenar en caché traducciones de modo que los accesos posteriores al bloque de memoria de ordenador 5002 no requieren el retardo de traducción de dirección. Normalmente, una caché 5009 se emplea entre la memoria de ordenador 5002 y el procesador 5001. La caché 5009 puede ser jerárquica con una caché grande disponible para más de una CPU y cachés más pequeñas y más rápidas (nivel más bajo) entre la caché grande y cada CPU. En algunas implementaciones, las cachés de nivel más bajo se dividen para proveer cachés de nivel bajo separadas para la captura de instrucciones y accesos a datos. En una realización, para la facilidad TX, un bloque de diagnóstico de transacción (TDB) 5100 y una o más memorias intermedias 5101 pueden almacenarse en una o más de caché 5009 y memoria 5002. En un ejemplo, en el modo TX, los datos se almacenan inicialmente en una memoria intermedia TX, y cuando el modo TX finaliza (p.ej., TEND más externa), los datos en la memoria intermedia se almacenan (comprometen) en la memoria o, si hay un aborto, los datos en la memoria intermedia se descartan.

En una realización, una instrucción se captura de la memoria 5002 por una unidad de captura de instrucciones 5004 mediante una caché 5009. La instrucción se decodifica en una unidad de decodificación de instrucciones 5006 y se despacha (con otras instrucciones en algunas realizaciones) a la unidad o unidades de ejecución de instrucciones 5008. Normalmente, varias unidades de ejecución 5008 se emplean, por ejemplo, una unidad de ejecución aritmética, una unidad de ejecución de punto flotante y una unidad de ejecución de instrucción de derivación. Además, en una realización de la facilidad TX, varios controles TX 5110 pueden emplearse. La instrucción se ejecuta por la unidad de ejecución, accediendo a operandos de registros o memoria de instrucciones especificados según sea necesario. Si se accede a un operando (cargado o almacenado) desde la memoria 5002, una unidad de carga/almacenamiento 5005 normalmente maneja el acceso bajo el control de la instrucción que se está ejecutando.

Las instrucciones pueden ejecutarse en circuitos de hardware o en un microcódigo interno (firmware) o por una combinación de ambos.

Según un aspecto de la facilidad TX, el procesador 5001 también incluye una PSW 5102 (p.ej., TX y/o PSW de aborto), una profundidad de jerarquización 5104, una TDBA 5106, y uno o más registros de control 5108.

Según se advierte, un sistema informático incluye información en el almacenamiento local (o principal), así como el direccionamiento, protección y referencia y registro de cambio. Algunos aspectos del direccionamiento incluyen el formato de direcciones, el concepto de espacios de dirección, los varios tipos de direcciones, y la manera en la cual un tipo de dirección se traduce en otro tipo de dirección. Parte del almacenamiento principal incluye ubicaciones de almacenamiento permanentemente asignadas. El almacenamiento principal provee al sistema un almacenamiento de datos de rápido acceso directamente direccionable. Tanto los datos como los programas se cargarán en el almacenamiento principal (desde dispositivos de entrada) antes de que puedan procesarse.

El almacenamiento principal puede incluir uno o más almacenamientos de memoria intermedia de rápido acceso, más pequeños, a veces llamados cachés. Una caché se asocia normalmente de manera física a una CPU o a un procesador E/S. Los efectos, excepto en el rendimiento, de la construcción física y uso de distintos medios de almacenamiento no son, en general, observables por el programa.

Las cachés separadas pueden mantenerse para instrucciones y para operandos de datos. La información dentro de una caché se mantiene en bytes contiguos en un límite integral llamado un bloque caché o línea de caché (o línea, para abreviar). Un modelo puede proveer una instrucción EXTRACT CACHE ATTRIBUTE que devuelve el tamaño de una línea de caché en bytes. Un modelo puede también proveer instrucciones PREFETCH DATA Y PREFETCH DATA RELATIVE LONG que efectúan la precaptura del almacenamiento en los datos o caché de instrucción o la liberación de datos de la caché.

El almacenamiento se ve como una cadena de bits horizontal larga. Para la mayoría de las operaciones, los accesos al almacenamiento proceden en una secuencia de izquierda a derecha. La cadena de bits se subdivide en unidades de ocho bits. Una unidad de ocho bits se llama un byte, que es el bloque de construcción básico de todos los formatos de información. Cada ubicación de byte en el almacenamiento se identifica por un entero no negativo único, que es la dirección de dicha ubicación de byte o, simplemente, la dirección de byte. Las ubicaciones de bytes adyacentes tienen direcciones consecutivas, comenzando con 0 a la izquierda y continuando en una secuencia de izquierda a derecha. Las direcciones son enteros binarios sin signo y son de 24, 31 o 64 bits.

La información se transmite entre el almacenamiento y una CPU o un byte de subsistema de canal, o un grupo de bytes, por vez. Salvo que se especifique lo contrario, en, por ejemplo, z/Architecture, un grupo de bytes en el almacenamiento se dirige por el byte más a la izquierda del grupo. El número de bytes en el grupo es implícito o se especifica de manera explícita por la operación que se llevará a cabo. Cuando se usan en una operación de CPU, un grupo de bytes se llama un campo. Dentro de cada grupo de bytes, en, por ejemplo, z/Architecture, los bits se numeran en una secuencia de izquierda a derecha. En z/Architecture, a veces se hace referencia a los bits más a la izquierda como los bits "de orden alto" y a los bits más a la derecha, como los bits "de orden bajo". Los números de bit no son direcciones de almacenamiento, sin embargo. Solo los bytes pueden dirigirse. Para operar en bits individuales de un byte en el almacenamiento, se accede a todo el byte. Los bits en un byte se numeran de 0 a 7, de izquierda a derecha (en, p.ej., z/Architecture). Los bits en una dirección pueden numerarse 8-31 o 40-63 para direcciones de 24 bits, o 1-31 o 33-63 para direcciones de 31 bits; ellos se numeran 0-63 para direcciones de 64 bits. En un ejemplo, los bits 8-31 y 1-31 se aplican a direcciones que están en una ubicación (p.ej., registro) que es de 32 bits de ancho, mientras que los bits 40-63 y 33-63 se aplican a direcciones que están en una ubicación de 64 bits de ancho. Dentro de cualquier otro formato de longitud fija de múltiples bytes, los bits que conforman el formato se numeran de manera consecutiva comenzando desde 0. A los fines de detección de errores, y preferiblemente para la corrección, uno o más bits de comprobación pueden transmitirse con cada byte o con un grupo de bytes. Dichos bits de comprobación se generan automáticamente por la máquina y no pueden controlarse directamente por el programa. Las capacidades de almacenamiento se expresan en número de bytes. Cuando la longitud de un campo de operando de almacenamiento es implícita por el código de operación de una instrucción, se dice que el campo tiene una longitud fija, que puede ser de uno, dos, cuatro, ocho o dieciséis bytes. Campos más grandes pueden suponerse para algunas instrucciones. Cuando la longitud de un campo de operando de almacenamiento no está implícita, pero se establece de forma explícita, se dice que el campo tiene una longitud variable. Los operandos de longitud variable pueden variar en longitud en incrementos de un byte (o con algunas instrucciones, en múltiplos de dos bytes u otros múltiplos). Cuando la información se coloca en el almacenamiento, los contenidos de solamente dichas ubicaciones de byte se reemplazan y se incluyen en el campo designado, aunque el ancho del trayecto físico al almacenamiento puede ser mayor que la longitud del campo que se está almacenando.

Ciertas unidades de información estarán en un límite integral en el almacenamiento. Un límite se llama integral para una unidad de información cuando su dirección de almacenamiento es un múltiplo de la longitud de la unidad en bytes. Nombres especiales se proveen a campos de 2, 4, 8, 16 y 32 bytes en un límite integral. Una media palabra es un grupo de dos bytes consecutivos en un límite de dos bytes y es el bloque de construcción básico de instrucciones. Una palabra es un grupo de cuatro bytes consecutivos en un límite de cuatro bytes. Una palabra doble es un grupo de ocho bytes consecutivos en un límite de ocho bytes. Una palabra cuádruple es un grupo de 16 bytes

consecutivos en un límite de 16 bytes. Una octopalabra es un grupo de 32 bytes consecutivos en un límite de 32 bytes. Cuando las direcciones de almacenamiento designan medias palabras, palabras, palabras dobles, palabras cuádruples y octopalabras, la representación binaria de la dirección contiene uno, dos, tres, cuatro o cinco bits cero más a la derecha, respectivamente. Las instrucciones estarán en límites integrales de dos bytes. Los operandos de almacenamiento de la mayoría de las instrucciones no tienen requisitos de alineación de límites.

En dispositivos que implementan cachés separadas para instrucciones y operandos de datos, un retardo significativo puede experimentarse si el programa almacena en una línea de caché desde la cual las instrucciones se capturan posteriormente, independientemente de si el almacenamiento altera las instrucciones que se capturan posteriormente.

En un ejemplo, las realizaciones pueden practicarse por software (al que a veces se hace referencia como código interno bajo licencia, firmware, microcódigo, milicódigo, picocódigo y similares, cualquiera de los cuales será coherente con una o más realizaciones). Con referencia a la Figura 20, puede accederse al código de programa de software que realiza uno o más aspectos por el procesador 5001 del sistema anfitrión 5000 desde dispositivos de medios de almacenamiento a largo plazo 5011 como, por ejemplo, una unidad CD-ROM, unidad de cinta o disco duro. El código de programa de software puede realizarse en cualquiera de una variedad de medios conocidos para su uso con un sistema de procesamiento de datos como, por ejemplo, un disquete, disco duro o CD-ROM. El código puede distribuirse en dichos medios, o puede distribuirse a usuarios de la memoria de ordenador 5002 o almacenamiento de un sistema informático en una red 5010 a otros sistemas informáticos para su uso por usuarios de dichos otros sistemas.

El código de programa de software incluye un sistema operativo que controla la función e interacción de los varios componentes de ordenador y uno o más programas de aplicación. El código de programa se pagina normalmente del dispositivo de medios de almacenamiento 5011 al almacenamiento de ordenador 5002 de velocidad relativamente más alta donde está disponible para el procesamiento por el procesador 5001. Las técnicas y los métodos para realizar el código de programa de software en la memoria, en medios físicos y/o distribuir el código de software mediante redes son conocidos y no se describirán más en la presente memoria. Con frecuencia, se hace referencia al código de programa, cuando se crea y almacena en un medio tangible (incluidos, pero sin limitación a ellos, módulos de memoria electrónicos (RAM), memoria flash, Discos Compactos (CD), DVD, Cinta Magnética y similares, como un "producto de programa de ordenador". El medio de producto de programa de ordenador es, normalmente, legible por un circuito de procesamiento preferiblemente en un sistema informático para la ejecución por el circuito de procesamiento.

La Figura 21 ilustra una estación de trabajo representativa o sistema de hardware de servidor en el cual una o más realizaciones pueden practicarse. El sistema 5020 de la Figura 21 comprende un sistema informático base 5021 representativo como, por ejemplo, un ordenador personal, una estación de trabajo o un servidor, incluidos los dispositivos periféricos opcionales. El sistema informático base 5021 incluye uno o más procesadores 5026 y un bus empleado para conectar y permitir la comunicación entre el(los) procesador(es) 5026 y los otros componentes del sistema 5021 según las técnicas conocidas. El bus conecta el procesador 5026 a la memoria 5025 y almacenamiento a largo plazo 5027 que puede incluir un disco duro (incluido cualquiera de medios magnéticos, CD, DVD y Memoria Flash, por ejemplo) o una unidad de cinta, por ejemplo. El sistema 5021 puede también incluir un adaptador de interfaz de usuario, que conecta el microprocesador 5026 mediante el bus a uno o más dispositivos de interfaz como, por ejemplo, un teclado 5024, un ratón 5023, una impresora/un escáner 5030 y/u otros dispositivos de interfaz, que pueden ser cualquier dispositivo de interfaz de usuario como, por ejemplo, una pantalla táctil, un pad de entradas digitalizado, etc. El bus también conecta un dispositivo de visualización 5022 como, por ejemplo, una pantalla o monitor LCD, al microprocesador 5026 mediante un adaptador de visualización.

El sistema 5021 puede comunicarse con otros ordenadores o redes de ordenadores por medio de un adaptador de red que pueda comunicarse 5028 con una red 5029. Los adaptadores de red a modo de ejemplo son canales de comunicaciones, anillo con paso de testigo, Ethernet o módems. De manera alternativa, el sistema 5021 puede comunicarse mediante el uso de una interfaz inalámbrica como, por ejemplo, una tarjeta CDPD (datos de paquete digital celular). El sistema 5021 puede asociarse a dichos otros ordenadores en una Red de Área Local (LAN) o Red de Área Amplia (WAN), o el sistema 5021 puede ser un cliente en una disposición cliente/servidor con otro ordenador, etc. Todas dichas configuraciones, así como el hardware y software de comunicaciones apropiados, se conocen en la técnica.

La Figura 22 ilustra una red de procesamiento de datos 5040 en la cual una o más realizaciones pueden practicarse. La red de procesamiento de datos 5040 puede incluir múltiples redes individuales como, por ejemplo, una red inalámbrica y una red cableada, cada una de las cuales puede incluir múltiples estaciones de trabajo individuales 5041, 5042, 5043, 5044. De manera adicional, como las personas con experiencia en la técnica apreciarán, una o más LAN pueden incluirse, donde una LAN puede comprender múltiples estaciones de trabajo inteligentes acopladas a un procesador anfitrión.

Aún con referencia a la Figura 22, las redes pueden también incluir ordenadores o servidores grandes como, por ejemplo, un ordenador de pasarela (servidor de cliente 5046) o servidor de aplicación (servidor remoto 5048 que puede acceder a un depósito de datos y al que puede accederse también de forma directa desde una estación de

trabajo 5045). Un ordenador de pasarela 5046 sirve como un punto de entrada en cada red individual. Una pasarela se necesita cuando se conecta un protocolo de redes a otro. La pasarela 5046 puede acoplarse preferiblemente a otra red (Internet 5047, por ejemplo) por medio de un enlace de comunicaciones. La pasarela 5046 puede también acoplarse directamente a una o más estaciones de trabajo 5041, 5042, 5043, 5044 mediante el uso de un enlace de comunicaciones. El ordenador de pasarela puede implementarse mediante la utilización de un servidor del Sistema eServer de IBM comercializado por International Business Machines Corporation.

Con referencia, de manera concurrente, a la Figura 21 y Figura 22, puede accederse al código de programación de software 5031 que puede realizar uno o más aspectos por el procesador 5026 del sistema 5020 desde medios de almacenamiento a largo plazo 5027 como, por ejemplo, una unidad CD-ROM o disco duro. El código de programación de software puede realizarse en cualquiera de una variedad de medios conocidos para su uso con un sistema de procesamiento de datos como, por ejemplo, un disquete, disco duro o CD-ROM. El código puede distribuirse en dichos medios, o puede distribuirse a usuarios 5050, 5051 de la memoria o almacenamiento de un sistema informático en una red a otros sistemas informáticos para su uso por usuarios de dichos otros sistemas.

De manera alternativa, el código de programación puede realizarse en la memoria 5025, y puede accederse a aquel por el procesador 5026 mediante el uso del bus de procesador. Dicho código de programación incluye un sistema operativo que controla la función e interacción de los varios componentes de ordenador y uno o más programas de aplicación 5032. El código de programa se pagina normalmente de los medios de almacenamiento 5027 a la memoria de alta velocidad 5025 donde está disponible para el procesamiento por el procesador 5026. Las técnicas y los métodos para realizar el código de programación de software en la memoria, en medios físicos y/o distribuir el código de software mediante redes son conocidos y no se describirán más en la presente memoria. Con frecuencia, se hace referencia al código de programa, cuando se crea y almacena en un medio tangible (incluidos, pero sin limitación a ellos, módulos de memoria electrónicos (RAM), memoria flash, Discos Compactos (CD), DVD, Cinta Magnética y similares, como un "producto de programa de ordenador". El medio de producto de programa de ordenador es, normalmente, legible por un circuito de procesamiento preferiblemente en un sistema informático para la ejecución por el circuito de procesamiento.

La caché que disponible de manera más inmediata para el procesador (normalmente más rápida y más pequeña que otras cachés del procesador) es la caché más baja (L1 o nivel uno) 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, con frecuencia, en una caché de instrucciones (I-Caché) que mantiene instrucciones de máquina que se ejecutarán y una caché de datos (D-Caché) que mantiene operandos de datos.

Con referencia a la Figura 23, una realización de procesador a modo de ejemplo se representa para el procesador 5026. Normalmente, uno o más niveles de caché 5053 se emplean para almacenar en memoria intermedia bloques de memoria con el fin de mejorar el rendimiento del procesador. La caché 5053 es una memoria intermedia de alta velocidad que mantiene líneas de caché de datos de memoria que probablemente se usarán. Las líneas de caché típicas son 64, 128 o 256 bytes de datos de memoria. Las cachés separadas se emplean, con frecuencia, para almacenar en caché instrucciones antes que para almacenar en caché datos. La coherencia de la caché (sincronización de copias de líneas en la memoria y las cachés) se provee, con frecuencia, por varios algoritmos "espía" conocidos en la técnica. Con frecuencia, se hace referencia al almacenamiento 5025 de memoria principal de un sistema de procesador como una caché. En un sistema de procesador que tiene 4 niveles de caché 5053, con frecuencia se hace referencia al almacenamiento principal 5025 como la caché de nivel 5 (L5) dado que normalmente es más rápido y solo mantiene una porción del almacenamiento permanente (DASD, cinta, etc.) que está disponible para un sistema informático. El almacenamiento principal 5025 "almacena en caché" páginas de datos paginados en y fuera del almacenamiento principal 5025 por el sistema operativo.

Un contador de programas (contador de instrucciones) 5061 mantiene un seguimiento de la dirección de la instrucción actual que se ejecutará. Un contador de programas en un procesador z/Architecture es de 64 bits y puede truncarse a 31 o 24 bits para soportar límites de direccionamiento previos. Un contador de programas se realiza, normalmente, en una PSW (palabra de estado de programa) de un ordenador de modo que persiste durante la conmutación de contexto. Por consiguiente, un programa en progreso, que tiene un valor de contador de programas, puede interrumpirse por, por ejemplo, el sistema operativo (conmutación de contexto del entorno de programa al entorno de sistema operativo). La PSW del programa mantiene el valor de contador de programas mientras el programa no está activo, y el contador de programas (en la PSW) del sistema operativo se usa mientras el sistema operativo se está ejecutando. Normalmente, el contador de programas se incrementa en una cantidad igual al número de bytes de la instrucción actual. Las instrucciones RISC (Cómputo Reducido de Conjunto de Instrucciones) son, normalmente, de longitud fija mientras que las instrucciones CISC (Cómputo Complejo de Conjunto de Instrucciones) son, normalmente, de longitud variable. Las instrucciones de z/Architecture de IBM son instrucciones CISC que tienen una longitud de 2, 4 o 6 bytes. El contador de programas 5061 se modifica por una operación de conmutación de contexto o una operación tomada de derivación de una instrucción de derivación, por ejemplo. En una operación de conmutación de contexto, el valor de contador de programa actual se guarda en la palabra de estado de programa junto con otra información de estado sobre el programa que se está ejecutando (como, por ejemplo, códigos de condición), y un nuevo valor de contador de programas se carga señalando una instrucción de un nuevo módulo de programa que se ejecutará. Una operación tomada de derivación se lleva a cabo con el fin de permitir que el programa tome decisiones o se enlace dentro del programa mediante la carga del

resultado de la instrucción de derivación en el contador de programas 5061.

Normalmente, una unidad de captura de instrucciones 5055 se emplea para capturar instrucciones en nombre del procesador 5026. La unidad de captura captura "siguientes instrucciones secuenciales", instrucciones objetivo de instrucciones tomadas de derivación, o primeras instrucciones de un programa que siguen a una conmutación de contexto. Las unidades de captura de Instrucciones Modernas con frecuencia emplean técnicas de precaptura para precapturar, de manera especulativa, instrucciones según la probabilidad de que las instrucciones precapturadas puedan usarse. Por ejemplo, una unidad de captura puede capturar 16 bytes de instrucción que incluyen la siguiente instrucción secuencial y bytes adicionales de instrucciones secuenciales adicionales.

Las instrucciones capturadas se ejecutan entonces por el procesador 5026. En una realización, las instrucciones capturadas pasan a una unidad de despacho 5056 de la unidad de captura. La unidad de despacho decodifica las instrucciones y reenvía información sobre las instrucciones decodificadas a las unidades apropiadas 5057, 5058, 5060. Una unidad de ejecución 5057 recibirá, normalmente, información sobre instrucciones aritméticas decodificadas de la unidad de captura de instrucciones 5055 y llevará a cabo operaciones aritméticas en operandos según el código operacional de la instrucción. Los operandos se proveen a la unidad de ejecución 5057 preferiblemente desde la memoria 5025, registros diseñados 5059 o desde un campo inmediato de la instrucción que se está ejecutando. Los resultados de la ejecución, cuando se almacenan, se almacenan en la memoria 5025, registros 5059 o en otro hardware de máquina (como, por ejemplo, registros de control, registros PSW y similares).

Las direcciones virtuales se transforman en direcciones reales mediante el uso de la traducción dinámica de direcciones 5062 y, de manera opcional, mediante el uso de la traducción de registro de acceso 5063.

Un procesador 5026 normalmente tiene una o más unidades 5057, 5058, 5060 para ejecutar la función de la instrucción. Con referencia a la Figura 24A, una unidad de ejecución 5057 puede comunicarse 5081 con registros generales diseñados 5059, una unidad de decodificación/despacho 5056, una unidad de almacenamiento de carga 5060, y otras 5065 unidades de procesador por medio de lógica 5071 de interacción. Una unidad de ejecución 5057 puede emplear varios circuitos de registro 5067, 5068, 5069 para mantener información en la que la unidad lógica aritmética (ALU, por sus siglas en inglés) 5066 operará. La ALU lleva a cabo operaciones aritméticas como, por ejemplo, sumar, restar, multiplicar y dividir, así como la función lógica como, por ejemplo, y, o y o exclusivo (XOR), rotar y desplazar. Preferiblemente, la ALU soporta operaciones especializadas que dependen del diseño. Otros circuitos pueden proveer otras facilidades diseñadas 5072 que incluyen códigos de condición y lógica de soporte de recuperación, por ejemplo. Normalmente, el resultado de una operación ALU se mantiene en un circuito de registro de salida 5070 que puede reenviar el resultado a una variedad de otras funciones de procesamiento. Hay muchas disposiciones de unidades de procesador, la presente descripción solo pretende proveer una comprensión representativa de una realización.

Una instrucción ADD, por ejemplo, se ejecutará en una unidad de ejecución 5057 que tiene una funcionalidad aritmética y lógica mientras que una instrucción de punto flotante, por ejemplo, se ejecutará en una ejecución de punto flotante que tiene capacidad de punto flotante especializada. Preferiblemente, una unidad de ejecución opera en operandos identificados por una instrucción llevando a cabo una función definida por código operacional en los operandos. Por ejemplo, una instrucción ADD puede ejecutarse por una unidad de ejecución 5057 en operandos encontrados en dos registros 5059 identificados por campos de registro de la instrucción.

La unidad de ejecución 5057 lleva a cabo la suma aritmética en dos operandos y almacena el resultado en un tercer operando donde el tercer operando puede ser un tercer registro o uno de los dos registros de origen. La unidad de ejecución utiliza, preferiblemente, una Unidad Lógica Aritmética (ALU) 5066 que puede llevar a cabo una variedad de funciones lógicas como, por ejemplo, Desplazar, Rotar, Y, O y XOR, así como una variedad de funciones algebraicas que incluyen cualquiera de sumar, restar, multiplicar, dividir. Algunas ALU 5066 se diseñan para operaciones escalares y algunas para el punto flotante. Los datos pueden ser Big Endian (donde el byte menos significativo está en la dirección del byte más alto) o Little Endian (donde el byte menos significativo está en la dirección del byte más bajo) dependiendo de la arquitectura. z/Architecture de IBM es Big Endian. Los campos con signo pueden ser signo y magnitud, complemento de 1 o complemento de 2, dependiendo de la arquitectura. Un número de complemento de 2 es ventajoso en que la ALU no necesita diseñar una capacidad de resta dado que un valor negativo o un valor positivo en el complemento de 2 solo requiere una suma dentro de la ALU. Los números se describen comúnmente de manera abreviada, donde un campo de 12 bits define una dirección de un bloque de 4.096 bytes y se describe, comúnmente, como un bloque de 4 Kbytes (Kilobytes), por ejemplo.

Con referencia a la Figura 24B, la información de instrucción de derivación para ejecutar una instrucción de derivación se envía, normalmente, a una unidad de derivación 5058 que, con frecuencia, emplea un algoritmo de predicción de derivación como, por ejemplo, una tabla de historial de derivación 5082 para predecir el resultado de la derivación antes de que otras operaciones condicionales se completen. El objetivo de la instrucción de derivación actual se capturará y ejecutará, de manera especulativa, antes de que las operaciones condicionales se completen. Cuando las operaciones condicionales están completadas, las instrucciones de derivación ejecutadas de manera especulativa se completan o descartan según las condiciones de la operación condicional y el resultado especulado. Una instrucción de derivación típica puede probar códigos de condición y derivar a una dirección objetivo si los códigos de condición satisfacen el requisito de derivación de la instrucción de derivación, una dirección objetivo

puede calcularse según varios números incluidos los encontrados en campos de registro o un campo inmediato de la instrucción, por ejemplo. La unidad de derivación 5058 puede emplear una ALU 5074 que tiene múltiples circuitos de registro de entrada 5075, 5076, 5077 y un circuito de registro de salida 5080. La unidad de derivación 5058 puede comunicarse con registros generales 5059, unidad de despacho de decodificación 5056 u otros circuitos 5073, por ejemplo.

La ejecución de un grupo de instrucciones puede interrumpirse por una variedad de motivos que incluyen una conmutación de contexto iniciada por un sistema operativo, una excepción de programa o error que provoca una conmutación de contexto, una señal de interrupción E/S que provoca una conmutación de contexto, o una actividad multihilo de múltiples programas (en un entorno multihilo), por ejemplo. Preferiblemente, una acción de conmutación de contexto guarda información de estado sobre un programa que se está ejecutando actualmente y luego carga información de estado sobre otro programa que se está invocando. La información de estado puede guardarse en registros de hardware o en memoria, por ejemplo. La información de estado comprende, preferiblemente, un valor de contador de programas que señala una siguiente instrucción que se ejecutará, códigos de condición, información de traducción de memoria y contenido de registro diseñado. Una actividad de conmutación de contexto puede ejercerse por circuitos de hardware, programas de aplicación, programas de sistema operativo o código de firmware (microcódigo, picocódigo o código interno bajo licencia (LIC, por sus siglas en inglés)) solos o en combinación.

Un procesador accede a operandos según métodos definidos por instrucciones. La instrucción puede proveer un operando inmediato mediante el uso del valor de una porción de la instrucción, puede proveer uno o más campos de registro que señalan explícitamente registros de propósito general o registros de propósito especial (registros de punto flotante, por ejemplo). La instrucción puede utilizar registros implícitos identificados por un campo de código operacional como operandos. La instrucción puede utilizar ubicaciones de memoria para operandos. Una ubicación de memoria de un operando puede proveerse por un registro, un campo inmediato, o una combinación de registros y campo inmediato según lo ejemplificado por la facilidad de desplazamiento largo z/Architecture en donde la instrucción define un registro base, un registro de índice y un campo inmediato (campo de desplazamiento) que se añaden juntos para proveer la dirección del operando en la memoria, por ejemplo. La ubicación en la presente memoria normalmente implica una ubicación en la memoria principal (almacenamiento principal) salvo que se indique lo contrario.

Con referencia a la Figura 24C, un procesador accede al almacenamiento mediante el uso de una unidad de carga/almacenamiento 5060. La unidad de carga/almacenamiento 5060 puede llevar a cabo una operación de carga al obtener la dirección del operando objetivo en la memoria 5053 y al cargar el operando en un registro 5059 u otra ubicación de memoria 5053, o puede llevar a cabo una operación de almacenamiento al obtener la dirección del operando objetivo en la memoria 5053 y al almacenar datos obtenidos de un registro 5059 u otra ubicación de memoria 5053 en la ubicación de operando objetivo en la memoria 5053. La unidad de carga/almacenamiento 5060 puede ser especulativa y puede acceder a la memoria en una secuencia que está fuera de orden con respecto a la secuencia de instrucciones, sin embargo, la unidad de carga/almacenamiento 5060 es para mantener la apariencia ante programas de que las instrucciones se han ejecutado en orden. Una unidad de carga/almacenamiento 5060 puede comunicarse 5084 con registros generales 5059, unidad de decodificación/despacho 5056, interfaz de caché/memoria 5053 u otros elementos 5083 y comprende varios circuitos de registro 5086, 5087, 5088 y 5089, ALU 5085 y lógica de control 5090 para calcular direcciones de almacenamiento y para proveer secuencia de conducto para mantener las operaciones en orden. Algunas operaciones pueden estar fuera de orden, pero la unidad de carga/almacenamiento provee funcionalidad para hacer que las operaciones fuera de orden aparezcan ante el programa como si se hubieran llevado a cabo en orden, como se conoce en la técnica.

Preferiblemente, con frecuencia se hace referencia a las direcciones que un programa de aplicación "ve" como direcciones virtuales. A veces se hace referencia a las direcciones virtuales como "direcciones lógicas" y "direcciones eficaces". Dichas direcciones virtuales son virtuales en que se redirigen a la ubicación de memoria física por una de una variedad de tecnologías de traducción dinámica de direcciones (DAT) que incluyen, pero sin limitación a ello, simplemente la prefijación de una dirección virtual con un valor de desplazamiento, traducción de la dirección virtual mediante una o más tablas de traducción, las tablas de traducción comprendiendo preferiblemente al menos una tabla de segmentos y una tabla de páginas solas o en combinación, preferiblemente, la tabla de segmentos tiene una entrada que señala la tabla de páginas. En z/Architecture, se provee una jerarquía de traducción que incluye una primera tabla de región, una segunda tabla de región, una tercera tabla de región y una tabla de segmentos y una tabla de páginas opcional. El rendimiento de la traducción de direcciones se mejora, con frecuencia, mediante la utilización de la memoria intermedia de traducción lateral (TLB) que comprende entradas que mapean una dirección virtual a una ubicación de memoria física asociada. Las entradas se crean cuando la DAT traduce una dirección virtual mediante el uso de las tablas de traducción. El posterior uso de la dirección virtual puede entonces utilizar la entrada de la TLB rápida antes que los accesos de la tabla de traducción secuencial lentos. El contenido de TLB puede gestionarse por una variedad de algoritmos de reemplazo, incluido LRU (Usado de Manera Menos Reciente).

En el caso donde el procesador es un procesador de un sistema de múltiples procesadores, cada procesador tiene la responsabilidad de mantener recursos compartidos como, por ejemplo, E/S, cachés, TLB y memoria, enclavados en aras de la coherencia. Normalmente, las tecnologías "espía" se utilizarán al mantener la coherencia de la caché. En un entorno espía, cada línea de caché puede marcarse como una que está en cualquiera de un estado

compartido, un estado exclusivo, un estado cambiado, un estado inválido y similares con el fin de facilitar la compartición.

Las unidades E/S 5054 (Figura 23) proveen al procesador medios para la fijación a dispositivos periféricos que incluyen cinta, disco, impresoras, visualizaciones y redes, por ejemplo. Las unidades E/S se presentan, con frecuencia, al programa de ordenador por controladores de software. En ordenadores grandes como, por ejemplo, System z de IBM®, los adaptadores de canal y adaptadores de sistemas abiertos son y/o unidades E/S de los ordenadores grandes que proveen las comunicaciones entre el sistema operativo y dispositivos periféricos.

Además, otros tipos de entornos informáticos pueden beneficiarse de uno o más aspectos. A modo de ejemplo, un entorno puede incluir un emulador (p.ej., software u otros mecanismos de emulación), en el cual una arquitectura particular (incluidos, por ejemplo, ejecución de instrucciones, funciones diseñadas como, por ejemplo, traducción de direcciones, y registros diseñados) o un subconjunto de aquella se emula (p.ej., en un sistema informático nativo que tiene un procesador y memoria). En dicho entorno, una o más funciones de emulación del emulador pueden implementar una o más realizaciones, aunque un ordenador que ejecuta el emulador puede tener una arquitectura diferente de las capacidades que se están emulando. A modo de ejemplo, en el modo de emulación, la instrucción u operación específica que se está emulando se decodifica, y una función de emulación apropiada se construye para implementar la instrucción u operación individual.

En un entorno de emulación, un ordenador anfitrión incluye, por ejemplo, una memoria para almacenar instrucciones y datos; una unidad de captura de instrucciones para capturar instrucciones de la memoria y para, de manera opcional, proveer memoria intermedia local para la instrucción capturada; una unidad de decodificación de instrucciones para recibir las instrucciones capturadas y para determinar el tipo de instrucciones que se han capturado; y una unidad de ejecución de instrucciones para ejecutar las instrucciones. La ejecución puede incluir cargar datos en un registro desde la memoria; almacenar datos otra vez en la memoria desde un registro; o llevar a cabo algún tipo de operación aritmética o lógica, según lo determinado por la unidad de decodificación. En un ejemplo, cada unidad se implementa en software. Por ejemplo, las operaciones que se llevan a cabo por las unidades se implementan como una o más subrutinas dentro del software de emulador.

Más concretamente, en un ordenador grande, instrucciones de máquina diseñadas se usan por programadores, normalmente hoy programadores "C", con frecuencia por medio de una aplicación de compilador. Dichas instrucciones almacenadas en el medio de almacenamiento pueden ejecutarse de manera nativa en un Servidor z/Architecture IBM® o, de manera alternativa, en máquinas que ejecutan otras arquitecturas. Estas pueden emularse en servidores de ordenadores grandes IBM® existentes y futuros y en otras máquinas de IBM® (p.ej., Servidores Power Systems y Servidores System x). Estas pueden ejecutarse en máquinas que ejecutan Linux en una amplia variedad de máquinas mediante el uso de hardware fabricado por IBM®, Intel®, AMD y otros. Además de la ejecución en dicho hardware bajo z/Architecture, Linux puede usarse, así como máquinas que usan la emulación por Hercules, UMX, o FSI (Fundamental Software, Inc), donde, en general, la ejecución es en un modo de emulación. En el modo de emulación, el software de emulación se ejecuta por un procesador nativo para emular la arquitectura de un procesador emulado.

El procesador nativo normalmente ejecuta software de emulación que comprende firmware o un sistema operativo nativo para llevar a cabo la emulación del procesador emulado. El software de emulación es responsable de la captura y ejecución de instrucciones de la arquitectura de procesador emulada. El software de emulación mantiene un contador de programa emulado para mantener un seguimiento de límites de instrucciones. El software de emulación puede capturar una o más instrucciones de máquina emuladas por vez y convertir la única o más instrucciones de máquina emuladas en un grupo correspondiente de instrucciones de máquina nativas para la ejecución por el procesador nativo. Dichas instrucciones convertidas pueden almacenarse en caché de modo que una conversión más rápida puede lograrse. Sin perjuicio de ello, el software de emulación es para mantener las reglas de arquitectura de la arquitectura de procesador emulada para asegurar que los sistemas operativos y aplicaciones escritas para el procesador emulado operen de manera correcta. Además, el software de emulación es para proveer recursos identificados por la arquitectura de procesador emulado que incluye, pero sin limitación a ello, registros de control, registros de propósito general, registros de punto flotante, función de traducción dinámica de direcciones que incluyen tablas de segmentos y tables de páginas, por ejemplo, mecanismos de interrupción, mecanismos de conmutación de contexto, relojes Hora del Día (TOD, por sus siglas en inglés) e interfaces diseñadas a subsistemas E/S de modo que un sistema operativo o un programa de aplicaciones diseñado para ejecutarse en el procesador emulado, puede ejecutarse en el procesador nativo que tiene el software de emulación.

Una instrucción específica que se está emulando se decodifica y una subrutina se llama para llevar a cabo la función de la instrucción individual. Una función de software de emulación que emula una función de un procesador emulado se implementa, por ejemplo, en una subrutina o unidad "C" o controlador, o algún otro método para proveer un controlador para el hardware específico como será parte de la experiencia de aquellos en la técnica después de comprender la descripción de la realización preferida. Varias patentes de emulación de software y hardware incluyen, pero sin limitación a ello, la Patente Real de Estados Unidos No. 5,551,013, titulada "*Multiprocessor for Hardware Emulation*", por Beausoleil y otros; y la Patente Real de Estados Unidos No. 6,009,261, titulada "*Preprocessing of Stored Target Routines for Emulating Incompatible Instructions on a Target Processor*", por Scalzi y otros; y la Patente Real de Estados Unidos No. 5,574,873, titulada "*Decoding Guest Instruction to Directly Access*

Emulation Routines that Emulate the Guest Instructions", por Davidian y otros; y la Patente Real de Estados Unidos No. 6,308,255, titulada "*Symmetrical Multiprocessing Bus and Chipset Used for Coprocessor Support Allowing Non-Native Code to Run in a System*", por Gorishek y otros; y la Patente Real de Estados Unidos No. 6,463,582, titulada "*Dynamic Optimizing Object Code Translator for Architecture Emulation and Dynamic Optimizing Object Code Translation Method*", por Lethin y otros; y la Patente Real de Estados Unidos No. 5,790,825, titulada "*Method for Emulating Guest Instructions on a Host Computer Through Dynamic Recompilation ofHost Instructions*", por Eric Traut; y muchas otras, ilustran una variedad de maneras conocidas para lograr la emulación de un formato de instrucción diseñado para una máquina diferente para una máquina objetivo disponible para aquellos con experiencia en la técnica.

En la Figura 25, se provee un ejemplo de un sistema informático anfitrión emulado 5092 que emula un sistema informático anfitrión 5000' de una arquitectura de anfitrión. En el sistema informático anfitrión emulado 5092, el procesador anfitrión (CPU) 5091 es un procesador anfitrión emulado (o procesador anfitrión virtual) y comprende un procesador de emulación 5093 que tiene una arquitectura de conjunto de instrucciones nativas diferente de la del procesador 5091 del ordenador anfitrión 5000'. El sistema informático anfitrión emulado 5092 tiene memoria 5094 accesible al procesador de emulación 5093. En la realización a modo de ejemplo, la memoria 5094 se particiona en una porción de memoria de ordenador anfitrión 5096 y una porción de rutinas de emulación 5097. La memoria de ordenador anfitrión 5096 está disponible para programas del ordenador anfitrión emulado 5092 según la arquitectura de ordenador anfitrión. El procesador de emulación 5093 ejecuta instrucciones nativas de un conjunto de instrucciones diseñadas de una arquitectura diferente de la del procesador emulado 5091, las instrucciones nativas obtenidas de la memoria de rutinas de emulación 5097, y puede acceder a una instrucción de anfitrión para la ejecución desde un programa en la memoria de ordenador anfitrión 5096 mediante el empleo de una o más instrucciones obtenidas en una rutina de secuencia & acceso/decodificación que puede decodificar la(s) instrucción(es) de anfitrión a la(s) que se accede para determinar una rutina de ejecución de instrucción nativa para emular la función de la instrucción de anfitrión a la que se accede. Otras facilidades que se definen para la arquitectura del sistema informático anfitrión 5000' pueden emularse por rutinas de facilidades diseñadas, incluidas facilidades como registros de propósito general, registros de control, traducción dinámica de direcciones y soporte de subsistema E/S y caché de procesador, por ejemplo. Las rutinas de emulación también pueden sacar ventaja de funciones disponibles en el procesador de emulación 5093 (como, por ejemplo, registros generales y traducción dinámica de direcciones virtuales) para mejorar el rendimiento de las rutinas de emulación. El hardware especial y los motores de descarga pueden también proveerse para ayudar al procesador 5093 en la emulación de la función del ordenador anfitrión 5000'.

La terminología usada en la presente memoria es a los fines de describir realizaciones particulares solamente y no pretende ser restrictiva. Según su uso en la presente memoria, las formas singulares "un", "una/o" y "el/la" pretenden incluir las formas plurales también, a menos que el contexto indique claramente lo contrario. Se comprenderá además que los términos "comprende(n)" y/o "que comprende(n)", cuando se usan en la presente memoria descriptiva, especifican la presencia de características, enteros, etapas, operaciones, elementos y/o componentes establecidos, pero no excluyen la presencia o incorporación de una o más de otras características, enteros, etapas, operaciones, elementos, componentes y/o grupos de ellos.

La descripción de una o más realizaciones se ha presentado en aras de la ilustración y descripción y no pretende ser exhaustiva o limitarse a la forma descrita. Muchas modificaciones y variaciones serán aparentes para las personas con experiencia ordinaria en la técnica. La realización se ha elegido y descrito con el fin de explicar mejor varios aspectos y la aplicación práctica, y para permitir que otros con experiencia ordinaria en la técnica comprendan varias realizaciones con varias modificaciones dado que son apropiadas para el uso particular contemplado.

REIVINDICACIONES

1. Un método para gestionar interrupciones en un entorno informático, el método comprendiendo las etapas de iniciar, por un procesador, una transacción mediante la ejecución de una instrucción *transaction begin* (200), la transacción retrasando, de manera eficaz, el compromiso de almacenamientos transaccionales con la memoria principal hasta la finalización de una transacción seleccionada, el método caracterizado por que:

la instrucción *transaction begin* incluye un campo (216) que especifica un control de filtrado de interrupción de programa, el control de filtrado de interrupción de programa controlando si ciertas clases de condiciones de excepción de programa que ocurren mientras el procesador está en el modo de ejecución transaccional resultan en una interrupción;

detecta, por el procesador, durante el procesamiento transaccional, una condición de excepción de programa (1700), la condición de excepción de programa teniendo al menos una clase de ejecución transaccional dependiendo de la gravedad de la condición de excepción de programa;

aborta la transacción, según la detección de la condición de excepción de programa;

determina, según la detección de la condición de excepción de programa, si una interrupción se presentará (1702), en donde la determinación emplea el control de filtrado de interrupción de programa (1754) y una clase de ejecución transaccional de la condición de excepción de programa (1756);

según la determinación que indica que la interrupción no se presentará, se evita que las condiciones de excepción de programa provoquen una interrupción (1758, 1760); y

reejecuta una o más instrucciones de la transacción seleccionada según el aborto de la transacción y la determinación de que la interrupción no se presentará.

2. El método de la reivindicación 1, en donde el control de filtrado de interrupción de programa se establece en un valor de múltiples valores, los múltiples valores indicando múltiples niveles de filtrado de interrupción.

3. El método de la reivindicación 2, en donde un primer nivel de los múltiples niveles no indica filtrado en el cual las condiciones de excepción de programa resultan en una interrupción, un segundo nivel indica filtrado limitado en el cual las condiciones de excepción de programa de una clase de transacciones seleccionadas no resultan en una interrupción, y un tercer nivel indica filtrado moderado en el cual las condiciones de excepción de programa de múltiples clases de transacciones seleccionadas no resultan en una interrupción.

4. El método de la reivindicación 3, en donde para el segundo nivel, las condiciones de excepción de programa que tienen un valor de clase de transacción de tres no resultan en una interrupción, y para el tercer nivel, las condiciones de excepción de programa que tienen un valor de clase de transacción de dos o tres no resultan en una interrupción.

5. El método de la reivindicación 3, en donde el único valor en el cual el control de filtrado de interrupción de programa se establece indica un nivel de filtrado de los múltiples niveles, y en donde la determinación comprende:

determinar una clase de transacción para la condición de excepción de programa (1756); y

comprobar el nivel de filtrado para saber si la clase de transacción resulta en una interrupción (1758), en donde la determinación indica que la interrupción no se presentará según la comprobación que indica que la clase de transacción para dicho nivel de filtrado no resulta en una interrupción (1760).

6. El método de la reivindicación 1, en donde el control de filtrado de interrupción de programa se incluye en una instrucción *transaction begin* (216), la instrucción *transaction begin* siendo una instrucción *transaction begin* de múltiples instrucciones *transaction begin* (600, 602, 604) usadas para crear un nido de transacciones, y en donde una o más de las instrucciones *transaction begin* incluyen un control de filtrado de interrupción de programa, y en donde el control de filtrado de interrupción de programa empleado es un valor más alto del único o más controles de filtrado de interrupción de programa.

7. El método de la reivindicación 1, en donde la transacción es una transacción restringida y el control de filtrado de interrupción de programa se establece en un valor por defecto.

8. El método de la reivindicación 1, en donde según la determinación que indica que la interrupción se presentará (1708), el método además comprende:

almacenar contenidos en una o más ubicaciones de memoria asignadas a la condición de excepción de programa (1708), en donde el almacenamiento comprende establecer un código de longitud de instrucción de una identificación de interrupción de programa con respecto a una instrucción en la cual la condición de excepción de programa ha ocurrido (1710); y

almacenar una palabra de estado de programa de aborto de transacción como una palabra de estado de programa

antiguo de programa (1712).

9. El método de la reivindicación 1, en donde el método además comprende abortar una transacción según la detección de la condición de excepción de programa, la transacción siendo una transacción no restringida y el control de filtrado de interrupción de programa indicando que la interrupción no se presentará.

10. Un sistema que comprende medios adaptados para llevar a cabo todas las etapas del método según cualquier reivindicación del método precedente.

11. Un programa de ordenador que comprende instrucciones para llevar a cabo todas las etapas del método según cualquier reivindicación del método precedente, cuando dicho programa de ordenador se ejecuta en un sistema informático.

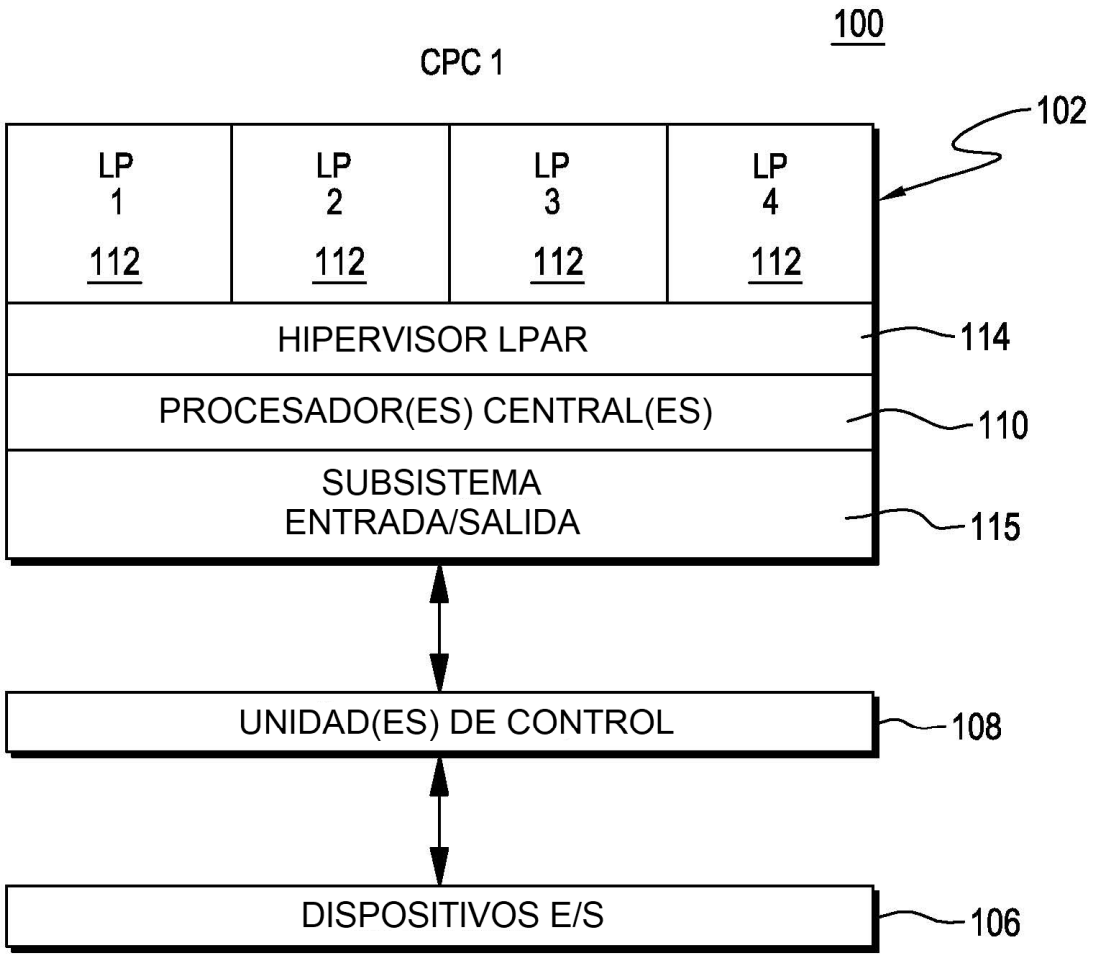
100
CPC 1
102
LP 1
112
LP 2
112
LP 3
112
LP 4
112
HIPERVISOR LPAR
114
PROCESADOR(ES) CENTRAL(ES)
110
SUBSISTEMA ENTRADA/SALIDA
115
UNIDAD(ES) DE CONTROL
108
DISPOSITIVOS E/S
106

FIG. 1

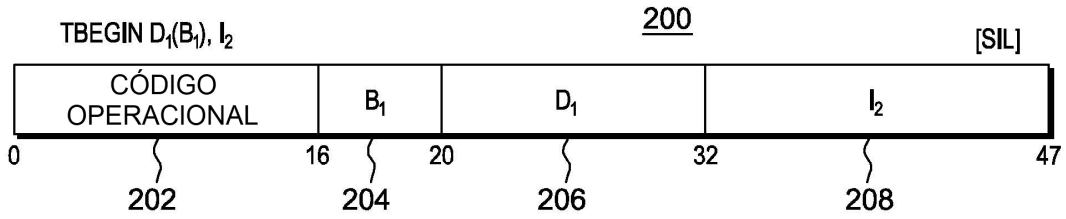

FIG. 2A

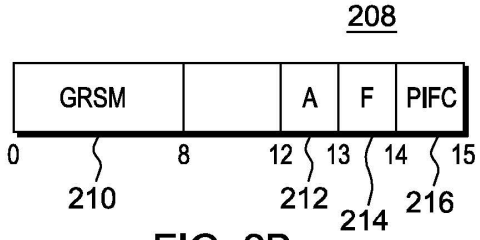

FIG. 2B

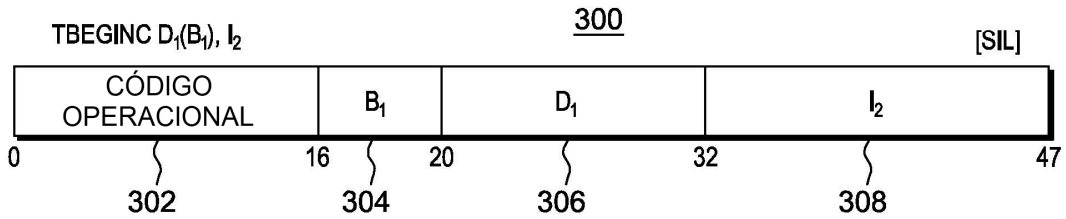

FIG. 3A

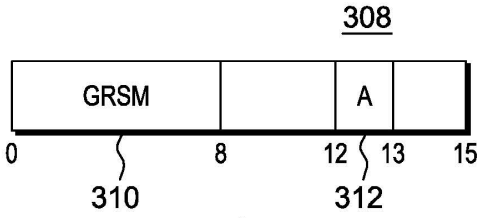

FIG. 3B

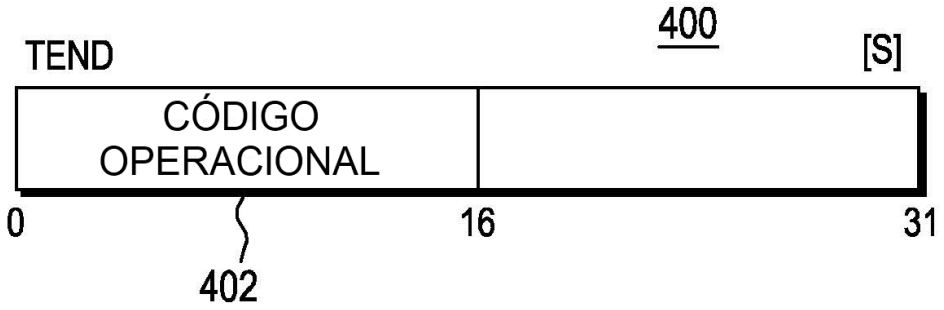
400
TEND
[S]
CÓDIGO OPERACIONAL
0
16
31
402

FIG. 4

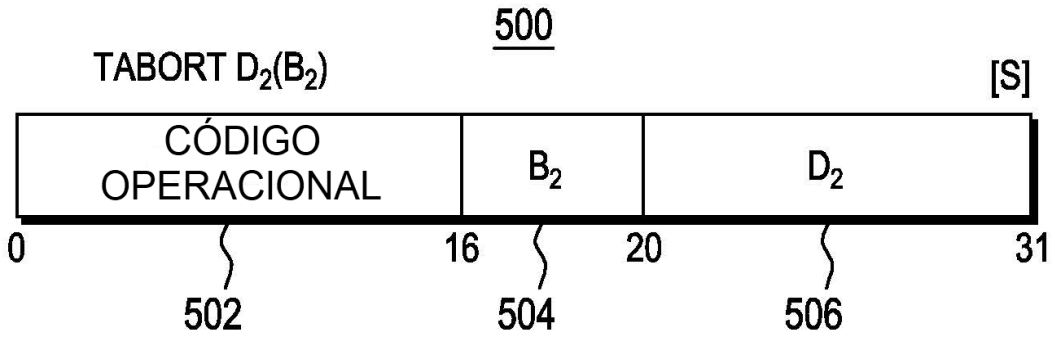
500
TABORT $D_2(B_2)$
[S]
CÓDIGO OPERACIONAL
B_2
D_2
0
16
20
31
502
504
506

FIG. 5

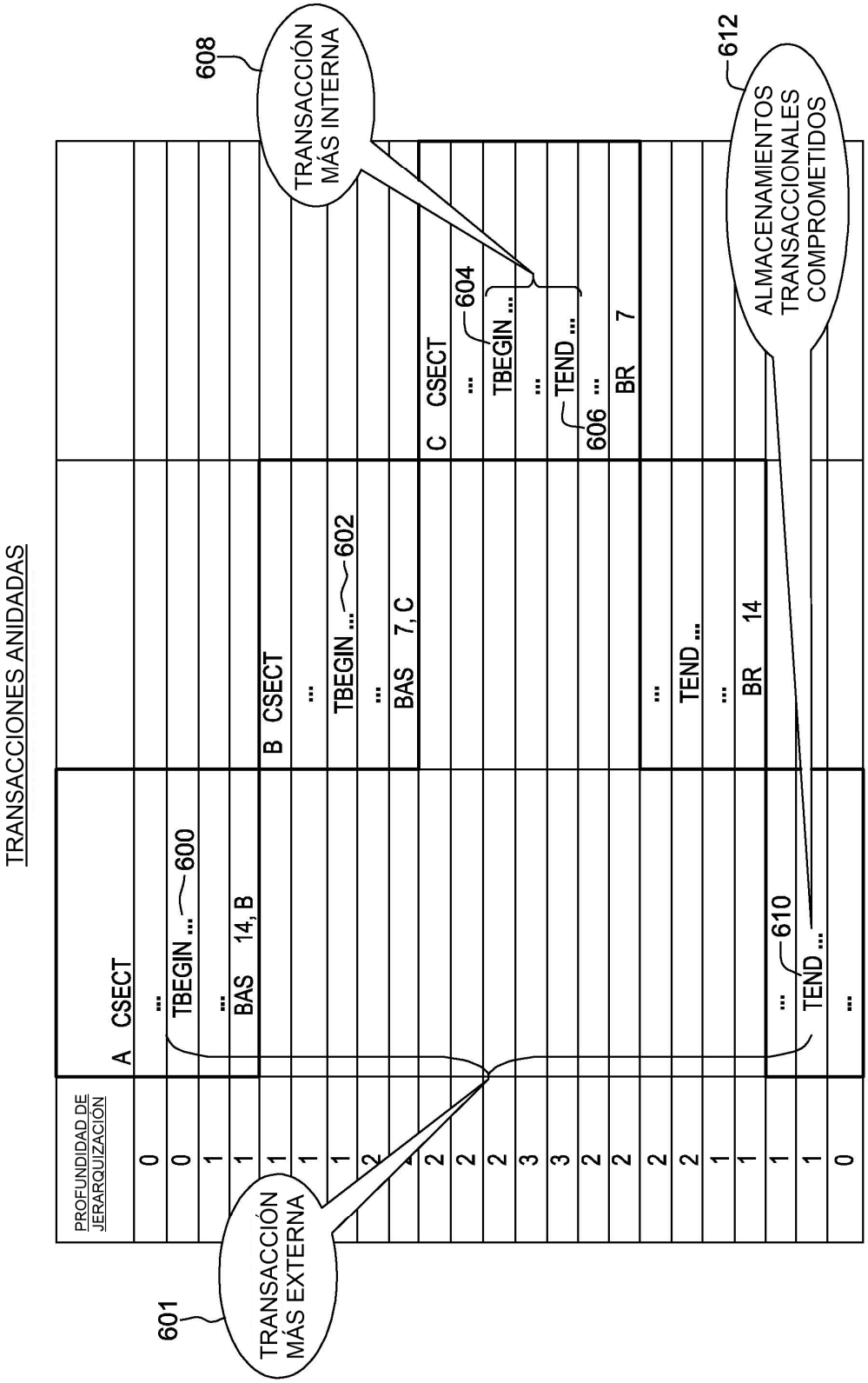
TRANSACCIONES ANIDADAS
PROFUNDIDAD DE JERARQUIZACIÓN
A CSECT
... 0
TBEGIN ... 600 0
... 1
BAS 14, B 1
B CSECT 1
... 1
TBEGIN ... 602 1
... 2
BAS 7, C 2
C CSECT 2
... 2
TBEGIN ... 604 2
... 3
TEND ... 3
606 ... 2
BR 7 2
... 2
TEND ... 2
... 1
BR 14 1
... 1
TEND ... 610 1
... 0
601
TRANSACCIÓN MÁS EXTERNA
608
TRANSACCIÓN MÁS INTERNA
612
ALMACENAMIENTOS TRANSACCIONALES COMPROMETIDOS

FIG. 6

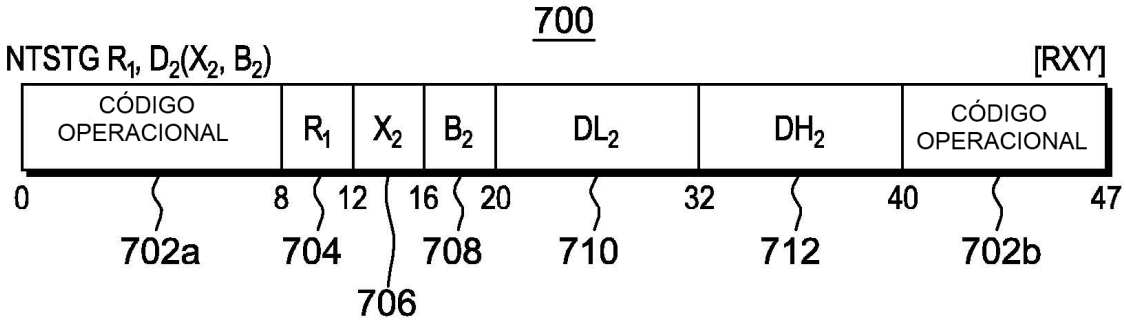
700
NTSTG R1, D2(X2, B2)
[RXY]
CÓDIGO OPERACIONAL
R1
X2
B2
DL2
DH2
CÓDIGO OPERACIONAL
0
8
12
16
20
32
40
47
702a
704
706
708
710
712
702b

FIG. 7

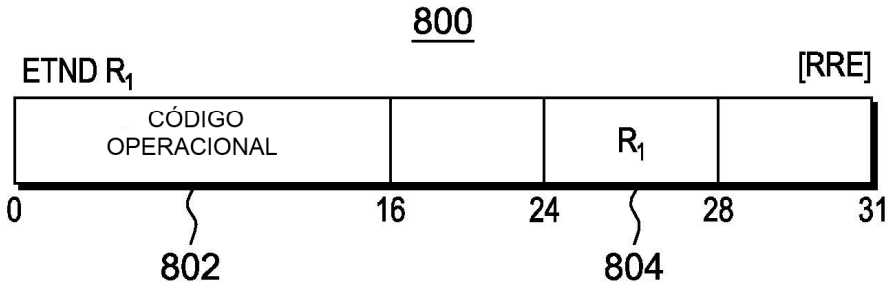
800
ETND R1
[RRE]
CÓDIGO OPERACIONAL
R1
0
16
24
28
31
802
804

FIG. 8

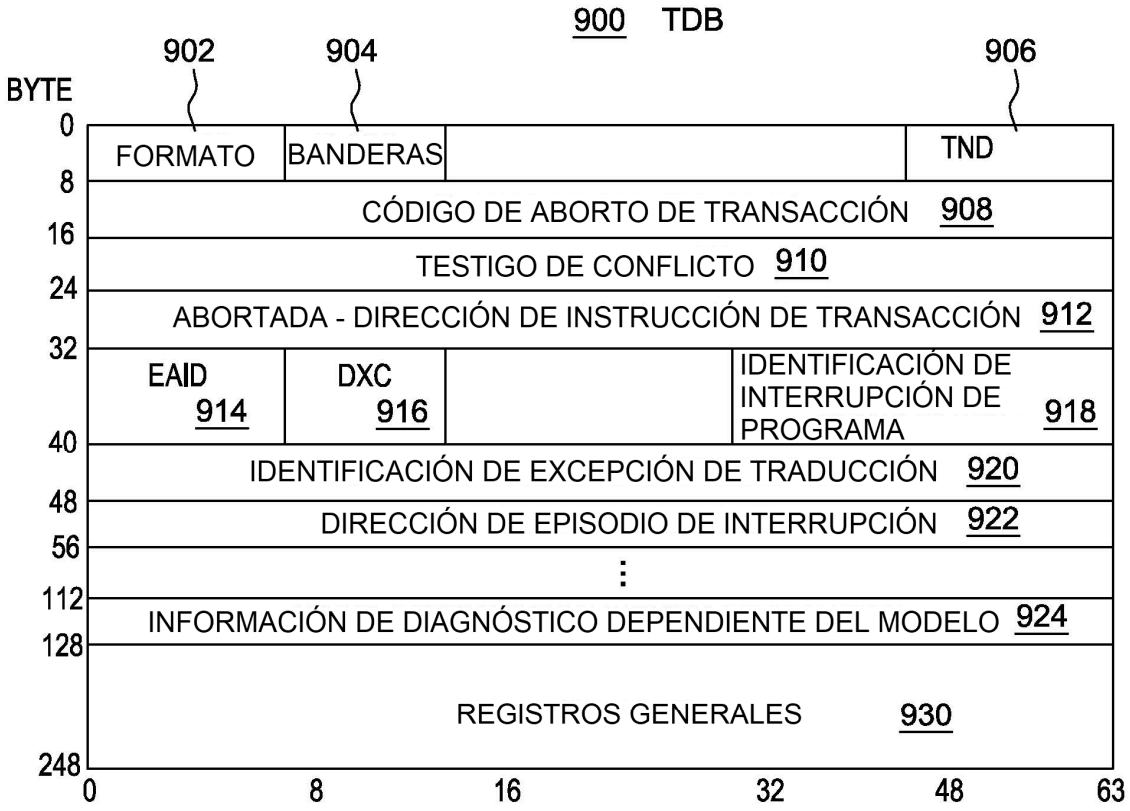
900 TDB
902
904
906
BYTE
0
8
16
24
32
40
48
56
112
128
248
FORMATO
BANDERAS
TND
CÓDIGO DE ABORTO DE TRANSACCIÓN 908
TESTIGO DE CONFLICTO 910
ABORTADA - DIRECCIÓN DE INSTRUCCIÓN DE TRANSACCIÓN 912
EAID 914
DXC 916
IDENTIFICACIÓN DE INTERRUPCIÓN DE PROGRAMA 918
IDENTIFICACIÓN DE EXCEPCIÓN DE TRADUCCIÓN 920
DIRECCIÓN DE EPISODIO DE INTERRUPCIÓN 922
INFORMACIÓN DE DIAGNÓSTICO DEPENDIENTE DEL MODELO 924
REGISTROS GENERALES 930
0
8
16
32
48
63

FIG. 9

CÓDIGO	MOTIVO PARA EL ABORTO	CONJUNTO CC
2	INTERRUPCIÓN EXTERNA	2
4	INTERRUPCIÓN DE PROGRAMA (NO FILTRADA)	2 o 3+
5	MÁQUINA - VERIFICAR INTERRUPCIÓN	2
6	INTERRUPCIÓN E/S	2
7	CAPTURAR SUBREFLUJO	2 o 3
8	ALMACENAR SUBREFLUJO	2 o 3
9	CAPTURAR CONFLICTO	2
10	ALMACENAR CONFLICTO	2
11	INSTRUCCIÓN RESTRINGIDA	3
12	CONDICIÓN DE EXCEPCIÓN DE PROGRAMA (FILTRADA)	3
13	PROFUNDIDAD DE JERARQUIZACIÓN SUPERADA	3
14	CAPTURA DE CACHÉ RELACIONADA	2 o 3
15	ALMACENAMIENTO DE CACHÉ RELACIONADO	2 o 3
16	ALMACENAR EN CACHÉ OTROS	2 o 3
255	CONDICIÓN DIVERSA	2 o 3
>255	INSTRUCCIÓN TABORT	2 o 3
‡	NO PUEDE DETERMINARSE; NO SE ALMACENA TDB	1

EXPLICACIÓN:

\+ EL CÓDIGO DE CONDICIÓN SE BASA EN EL CÓDIGO DE INTERRUPCIÓN

‡ LA PRESENTE SITUACIÓN OCURRE CUANDO UNA TRANSACCIÓN SE ABORTA, PERO EL TDB SE HA CONVERTIDO EN INACCESIBLE DESPUÉS DE LA EJECUCIÓN EXITOSA DE LA INSTRUCCIÓN TBEGIN MÁS EXTERNA. NO SE ALMACENA TDB DE TBEGIN ESPECIFICADA, Y EL CÓDIGO DE CONDICIÓN SE ESTABLECE EN 1.

FIG. 10

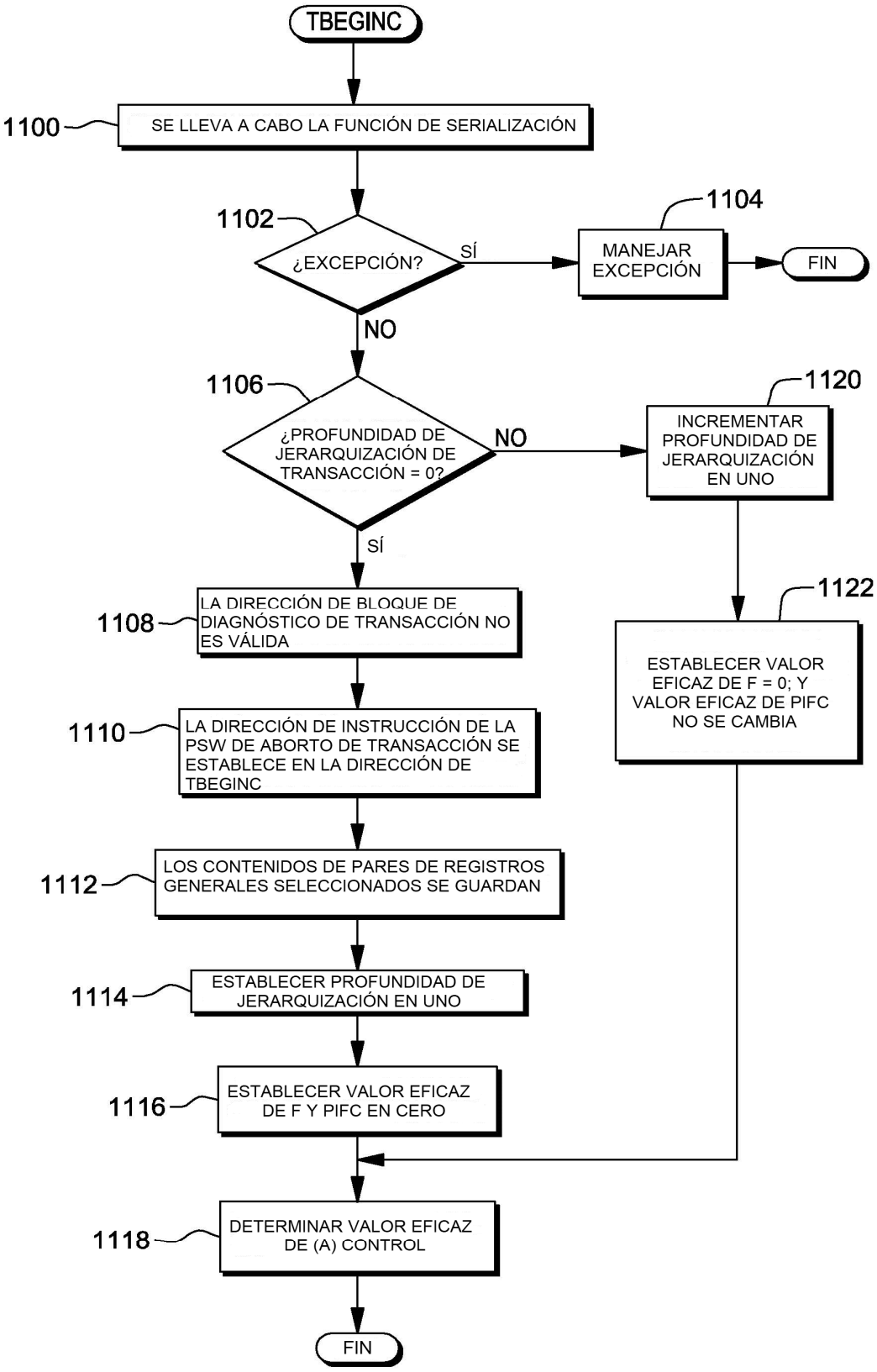
TBEGINC
1100
SE LLEVA A CABO LA FUNCIÓN DE SERIALIZACIÓN
1102
¿EXCEPCIÓN?
SÍ
1104
MANEJAR EXCEPCIÓN
FIN
NO
1106
¿PROFUNDIDAD DE JERARQUIZACIÓN DE TRANSACCIÓN = 0?
NO
1120
INCREMENTAR PROFUNDIDAD DE JERARQUIZACIÓN EN UNO
SÍ
1108
LA DIRECCIÓN DE BLOQUE DE DIAGNÓSTICO DE TRANSACCIÓN NO ES VÁLIDA
1122
ESTABLECER VALOR EFICAZ DE F = 0; Y VALOR EFICAZ DE PIFC NO SE CAMBIA
1110
LA DIRECCIÓN DE INSTRUCCIÓN DE LA PSW DE ABORTO DE TRANSACCIÓN SE ESTABLECE EN LA DIRECCIÓN DE TBEGINC
1112
LOS CONTENIDOS DE PARES DE REGISTROS GENERALES SELECCIONADOS SE GUARDAN
1114
ESTABLECER PROFUNDIDAD DE JERARQUIZACIÓN EN UNO
1116
ESTABLECER VALOR EFICAZ DE F Y PIFC EN CERO
1118
DETERMINAR VALOR EFICAZ DE (A) CONTROL
FIN

FIG. 11

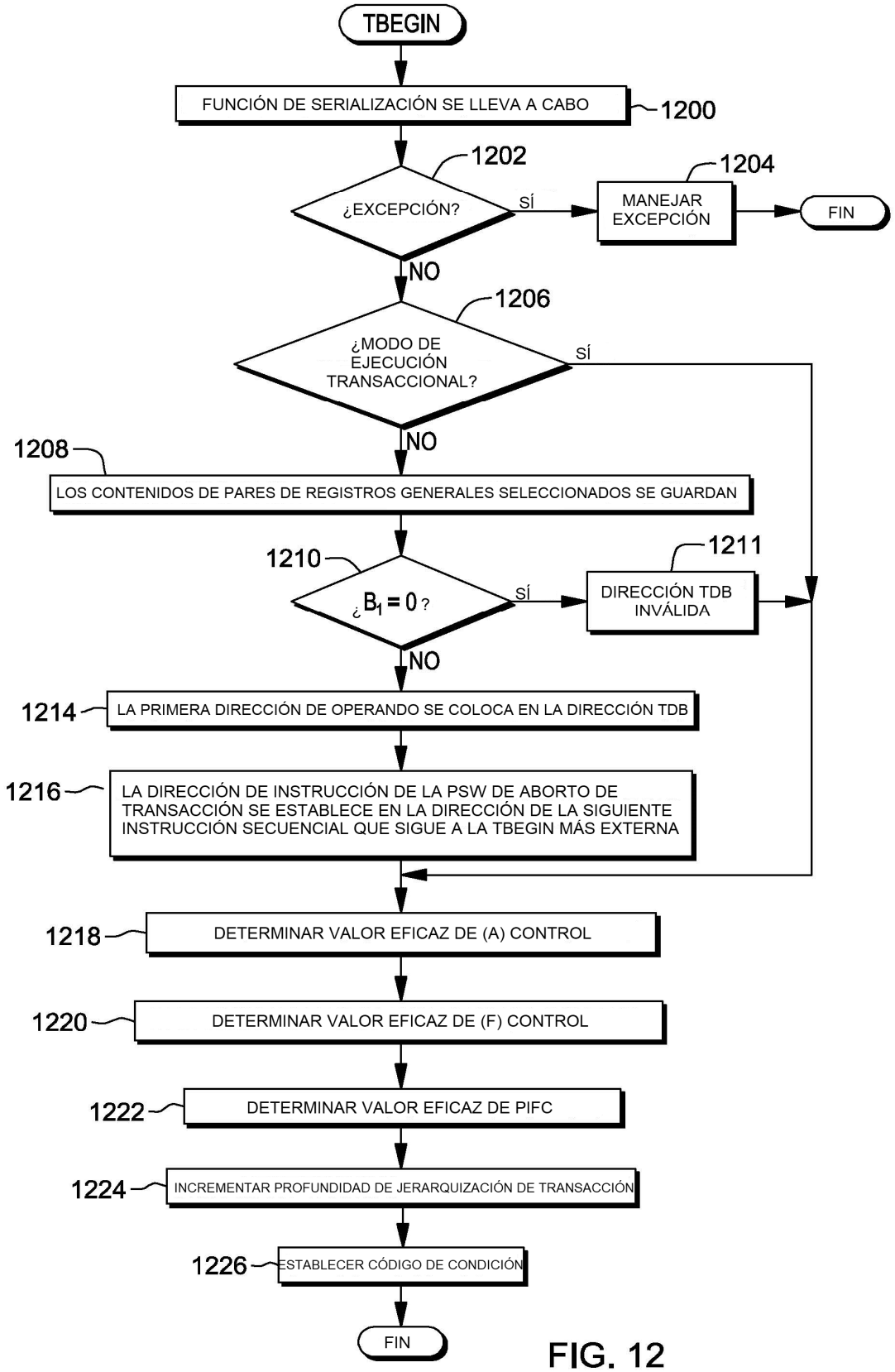
TBEGIN
FUNCIÓN DE SERIALIZACIÓN SE LLEVA A CABO
1200
1202
¿EXCEPCIÓN?
SÍ
1204
MANEJAR EXCEPCIÓN
FIN
NO
1206
¿MODO DE EJECUCIÓN TRANSACCIONAL?
SÍ
NO
1208
LOS CONTENIDOS DE PARES DE REGISTROS GENERALES SELECCIONADOS SE GUARDAN
1210
¿B1 = 0?
SÍ
1211
DIRECCIÓN TDB INVÁLIDA
NO
1214
LA PRIMERA DIRECCIÓN DE OPERANDO SE COLOCA EN LA DIRECCIÓN TDB
1216
LA DIRECCIÓN DE INSTRUCCIÓN DE LA PSW DE ABORTO DE TRANSACCIÓN SE ESTABLECE EN LA DIRECCIÓN DE LA SIGUIENTE INSTRUCCIÓN SECUENCIAL QUE SIGUE A LA TBEGIN MÁS EXTERNA
1218
DETERMINAR VALOR EFICAZ DE (A) CONTROL
1220
DETERMINAR VALOR EFICAZ DE (F) CONTROL
1222
DETERMINAR VALOR EFICAZ DE PIFC
1224
INCREMENTAR PROFUNDIDAD DE JERARQUIZACIÓN DE TRANSACCIÓN
1226
ESTABLECER CÓDIGO DE CONDICIÓN
FIN

FIG. 12

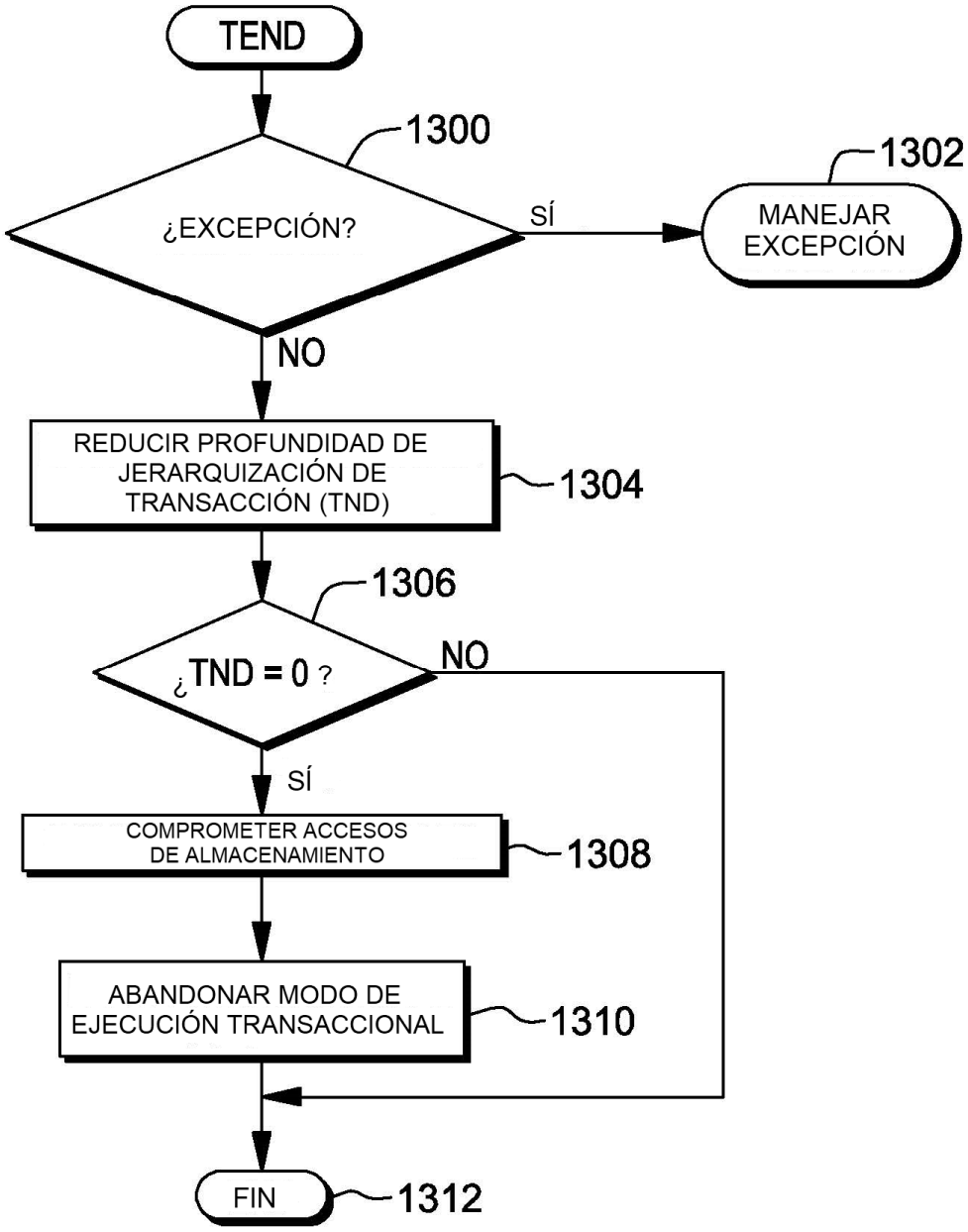
TEND
1300
¿EXCEPCIÓN?
SÍ
1302
MANEJAR EXCEPCIÓN
NO
REDUCIR PROFUNDIDAD DE JERARQUIZACIÓN DE TRANSACCIÓN (TND)
1304
1306
¿TND = 0 ?
NO
SÍ
COMPROMETER ACCESOS DE ALMACENAMIENTO
1308
ABANDONAR MODO DE EJECUCIÓN TRANSACCIONAL
1310
FIN
1312

FIG. 13

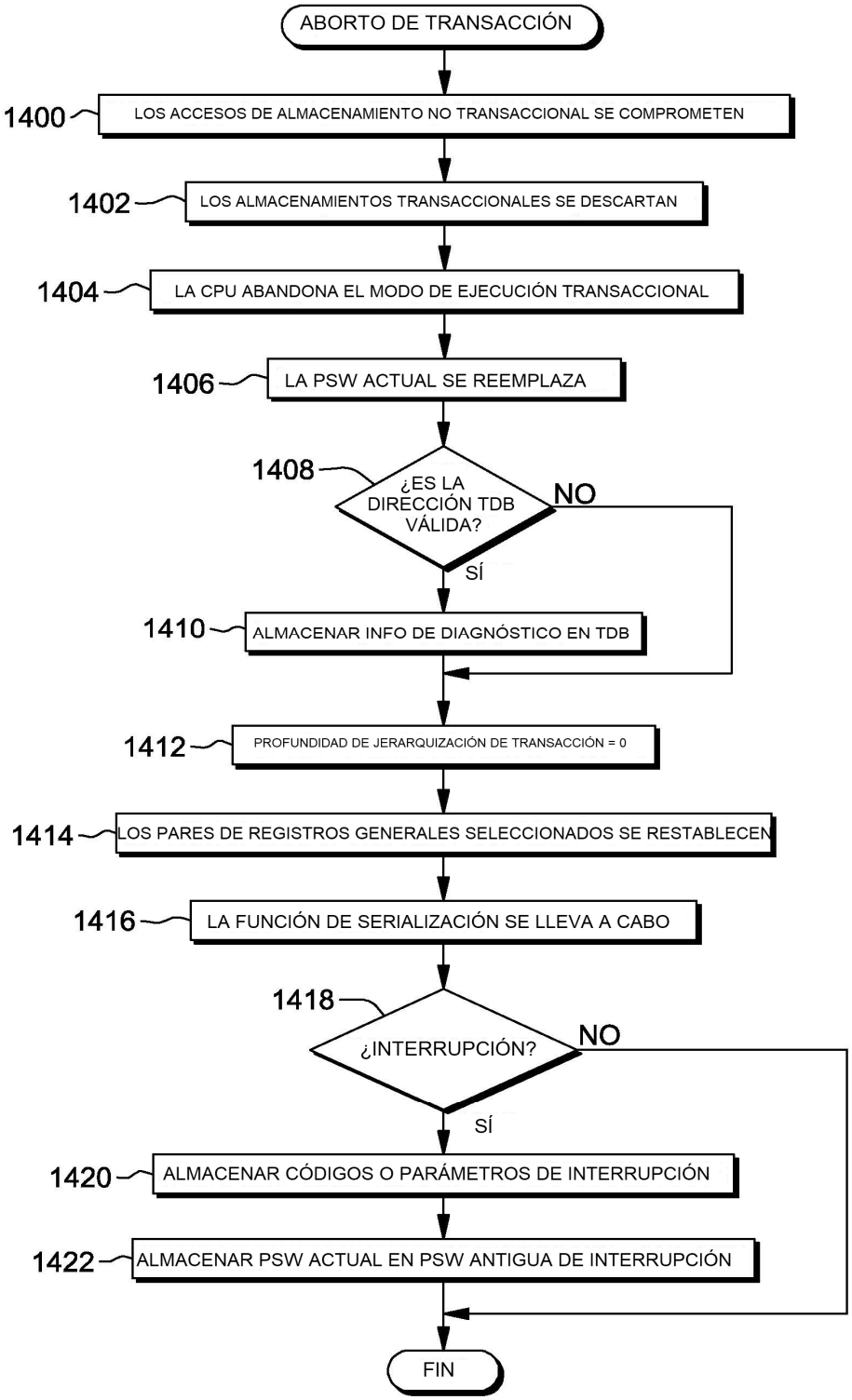
ABORTO DE TRANSACCIÓN
1400
LOS ACCESOS DE ALMACENAMIENTO NO TRANSACCIONAL SE COMPROMETEN
1402
LOS ALMACENAMIENTOS TRANSACCIONALES SE DESCARTAN
1404
LA CPU ABANDONA EL MODO DE EJECUCIÓN TRANSACCIONAL
1406
LA PSW ACTUAL SE REEMPLAZA
1408
¿ES LA DIRECCIÓN TDB VÁLIDA?
NO
SÍ
1410
ALMACENAR INFO DE DIAGNÓSTICO EN TDB
1412
PROFUNDIDAD DE JERARQUIZACIÓN DE TRANSACCIÓN = 0
1414
LOS PARES DE REGISTROS GENERALES SELECCIONADOS SE RESTABLECEN
1416
LA FUNCIÓN DE SERIALIZACIÓN SE LLEVA A CABO
1418
¿INTERRUPCIÓN?
NO
SÍ
1420
ALMACENAR CÓDIGOS O PARÁMETROS DE INTERRUPCIÓN
1422
ALMACENAR PSW ACTUAL EN PSW ANTIGUA DE INTERRUPCIÓN
FIN

FIG. 14

PIFC eficaz	Filtrado de Interrupción de Programa	Resultado Según Clase de Ejecución Transaccional (TX)		
		1	2	3
0	Ninguno	Interrumpir	Interrumpir	Interrumpir
1	Limitado	Interrumpir	Interrumpir	Filtrado
2	Moderado	Interrumpir	Filtrado	Filtrado

Explicación :

PIFC Control de Filtrado de Interrupción de Programa (bits 14-15 del I_2 campo de la instrucción TBEGIN; ceros para TBEGINC); el PIFC eficaz es el valor más alto especificado (o implícito) para las instrucciones TRANSACTION BEGIN actuales y todas las de nivel externo.

FIG. 15

Código (Hex)	Condición de Excepción	Clase TX	Conjunto CC
0001	Operación	1	3
0002	Operación Privilegiada	1	3
0003	Ejecutar	1	3
0004	Protección	1 o 2 †	2 o 3 ‡
0005	Direccionamiento	1 o 2 †	2 o 3 ‡
0006	Especificación	3	2 o 3 ‡
0007	Datos (DXC 01, 02 y 03)	1	2
0007	Datos (todos los otros DXC)	3	2 o 3 ‡
0008	Sobreflujo de punto fijo	3	2 o 3 ‡
0009	Dividir punto fijo	3	2 o 3 ‡
000A	Sobreflujo de decimal	3	2 o 3 ‡
000B	Dividir decimal	3	2 o 3 ‡
000C	Sobreflujo de exponente HFP	3	2 o 3 ‡
000D	Subdesbordamiento de exponente HFP	3	2 o 3 ‡
000E	Significado HFP	3	2 o 3 ‡
000F	Dividir HFP	3	2 o 3 ‡
0010	Traducción de segmentos	1 o 2 †	2 o 3 ‡
0011	Traducción de páginas	1 o 2 †	2 o 3 ‡
0012	Especificación de traducción	1	3
0013	Operación especial	1	3
0015	Operando	-	-
0016	Tabla de seguimiento	-	-
0018	Limitación de transacción	1	3

FIG. 16A

Código (Hex)	Condición de Excepción	Clase TX	Conjunto CC
001C	Episodio de conmutación de espacio	-	-
001D	Raíz cuadrada HFP	3	2 o 3 ‡
001F	Especificación de traducción PC	-	-
0020	Traducción AFX	-	-
0021	Traducción ASX	-	-
0022	Traducción LX	-	-
0023	Traducción EX	-	-
0024	Autoridad primaria	-	-
0025	Autoridad secundaria	-	-
0026	Traducción LFX	-	-
0027	Traducción LSX	-	-
0028	Especificación ALET	2	2 o 3 ‡
0029	Traducción ALEN	2	2 o 3 ‡
002A	Secuencia ALE	2	2 o 3 ‡
002B	Validez ASTE	2	2 o 3 ‡
002C	Secuencia ASTE	2	2 o 3 ‡
002D	Autoridad extendida	2	2 o 3 ‡
002E	Secuencia LSTE	-	-
002F	Instancia ASTE	-	-
0030	Pila llena	-	-
0031	Vaciar pila	-	-
0032	Especificación de pila	-	-
0033	Tipo pila	-	-
0034	Operación de pila	-	-
0038	Tipo ASCE	1 o 2 †	2 o 3 ‡
0039	Primera traducción de región	1 o 2 †	2 o 3 ‡
003A	Segunda traducción de región	1 o 2 †	2 o 3 ‡
003B	Tercera traducción de región	1 o 2 †	2 o 3 ‡
0040	Episodio de monitoreo	-	-
0080	Episodio PER	1	3★

Explicación :

- No aplicable; la excepción no puede ocurrir en el modo de ejecución transaccional porque la instrucción que provoca la excepción es una instrucción restringida.

† Las excepciones de acceso reconocidas durante la captura de una instrucción en el modo de ejecución transaccional son TX clase 1 (es decir, no pueden filtrarse). Las excepciones de acceso, cuando se accede a un operando de almacenamiento en el modo de ejecución transaccional, son TX clase 2.

‡ Cuando la condición de excepción no se filtra, el código de condición es 2; cuando la condición se filtra, el código de condición es 3.

★ La CPU establece el código de condición 3. Un programa de control o hipervisor puede cambiar el código de condición en la PSW antigua de programa a 2.

FIG. 16B

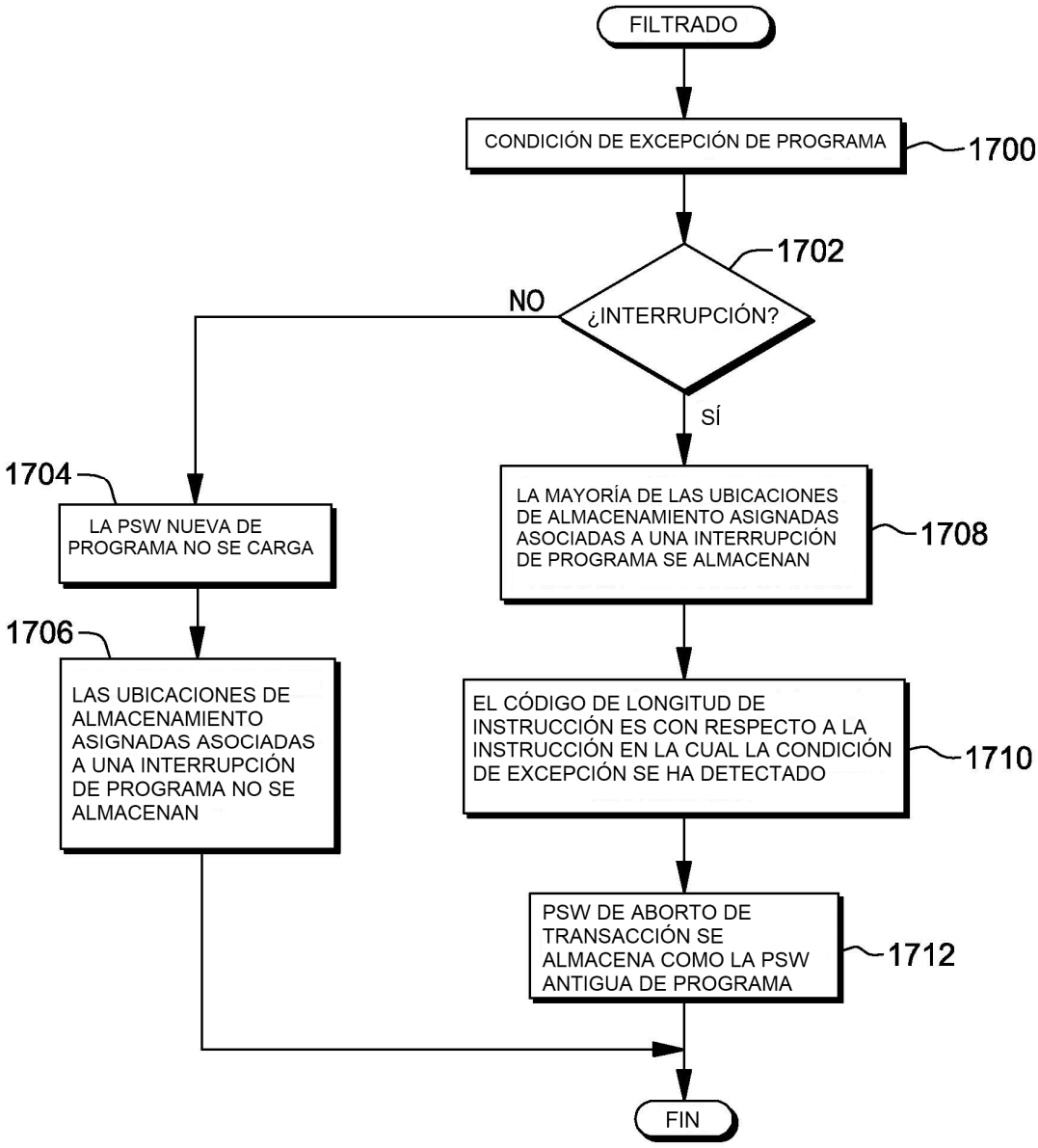
FILTRADO
CONDICIÓN DE EXCEPCIÓN DE PROGRAMA
1700
1702
¿INTERRUPCIÓN?
NO
SÍ
1704
LA PSW NUEVA DE PROGRAMA NO SE CARGA
1706
LAS UBICACIONES DE ALMACENAMIENTO ASIGNADAS ASOCIADAS A UNA INTERRUPCIÓN DE PROGRAMA NO SE ALMACENAN
LA MAYORÍA DE LAS UBICACIONES DE ALMACENAMIENTO ASIGNADAS ASOCIADAS A UNA INTERRUPCIÓN DE PROGRAMA SE ALMACENAN
1708
EL CÓDIGO DE LONGITUD DE INSTRUCCIÓN ES CON RESPECTO A LA INSTRUCCIÓN EN LA CUAL LA CONDICIÓN DE EXCEPCIÓN SE HA DETECTADO
1710
PSW DE ABORTO DE TRANSACCIÓN SE ALMACENA COMO LA PSW ANTIGUA DE PROGRAMA
1712
FIN

FIG. 17A

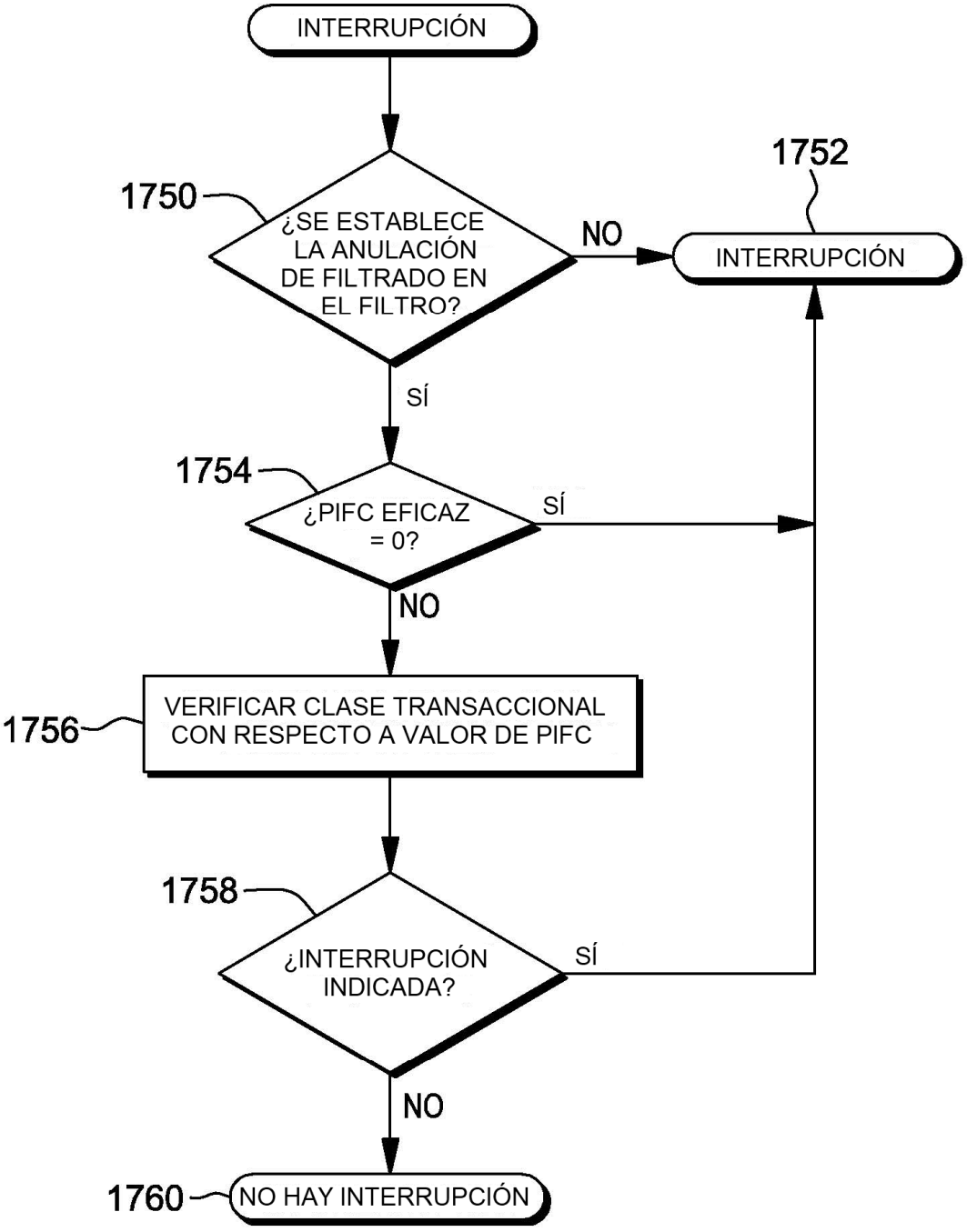
INTERRUPCIÓN
1750
¿SE ESTABLECE LA ANULACIÓN DE FILTRADO EN EL FILTRO?
NO
1752
INTERRUPCIÓN
SÍ
1754
¿PIFC EFICAZ = 0?
SÍ
NO
1756
VERIFICAR CLASE TRANSACCIONAL CON RESPECTO A VALOR DE PIFC
1758
¿INTERRUPCIÓN INDICADA?
SÍ
NO
1760
NO HAY INTERRUPCIÓN

FIG. 17B

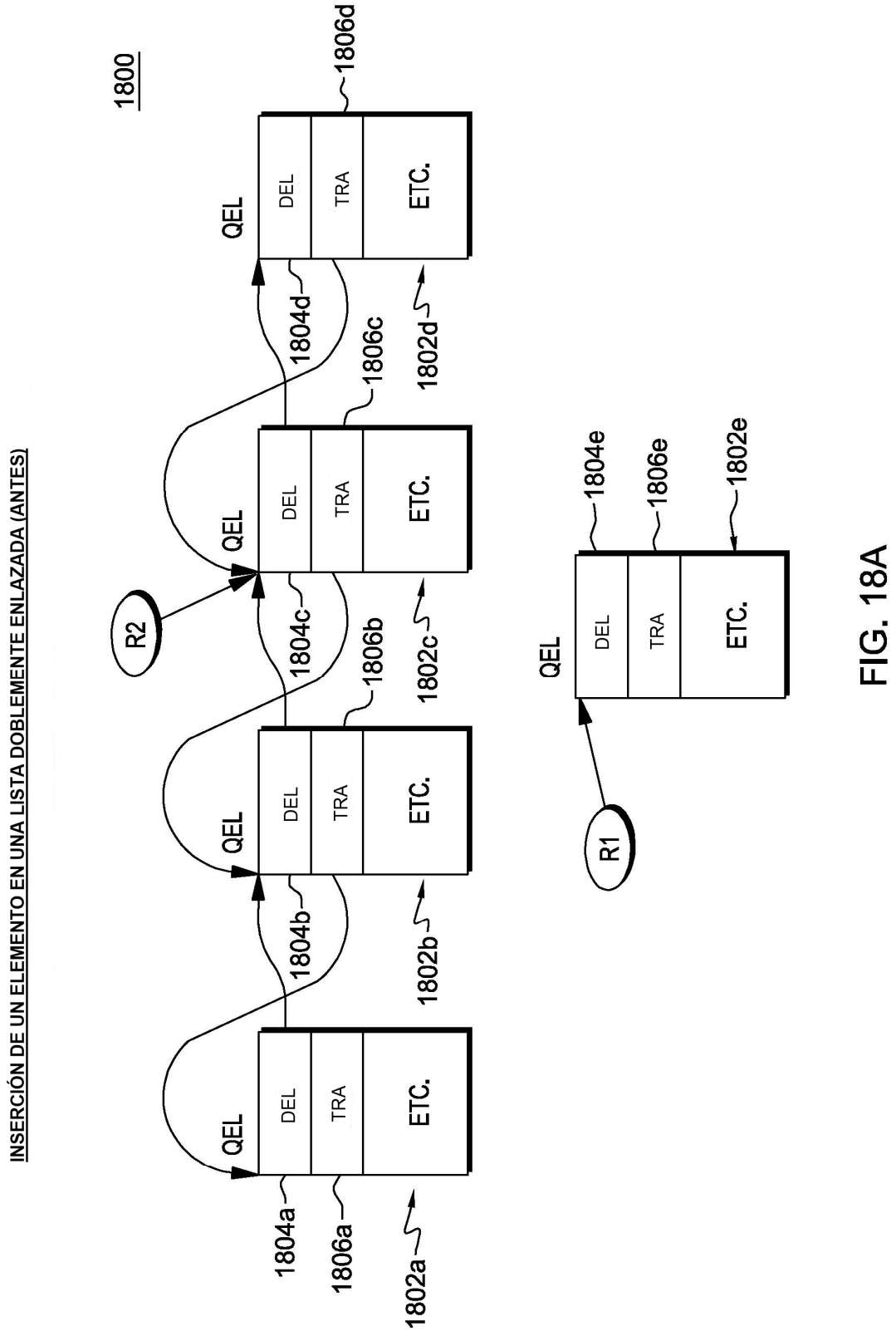
INSERCIÓN DE UN ELEMENTO EN UNA LISTA DOBLEMENTE ENLAZADA (ANTES)
1800
QEL
DEL
TRA
ETC.
1804a
1806a
1802a
1804b
1806b
1802b
1804c
1806c
1802c
1804d
1806d
1802d
R2
1804e
1806e
1802e
R1

FIG. 18A

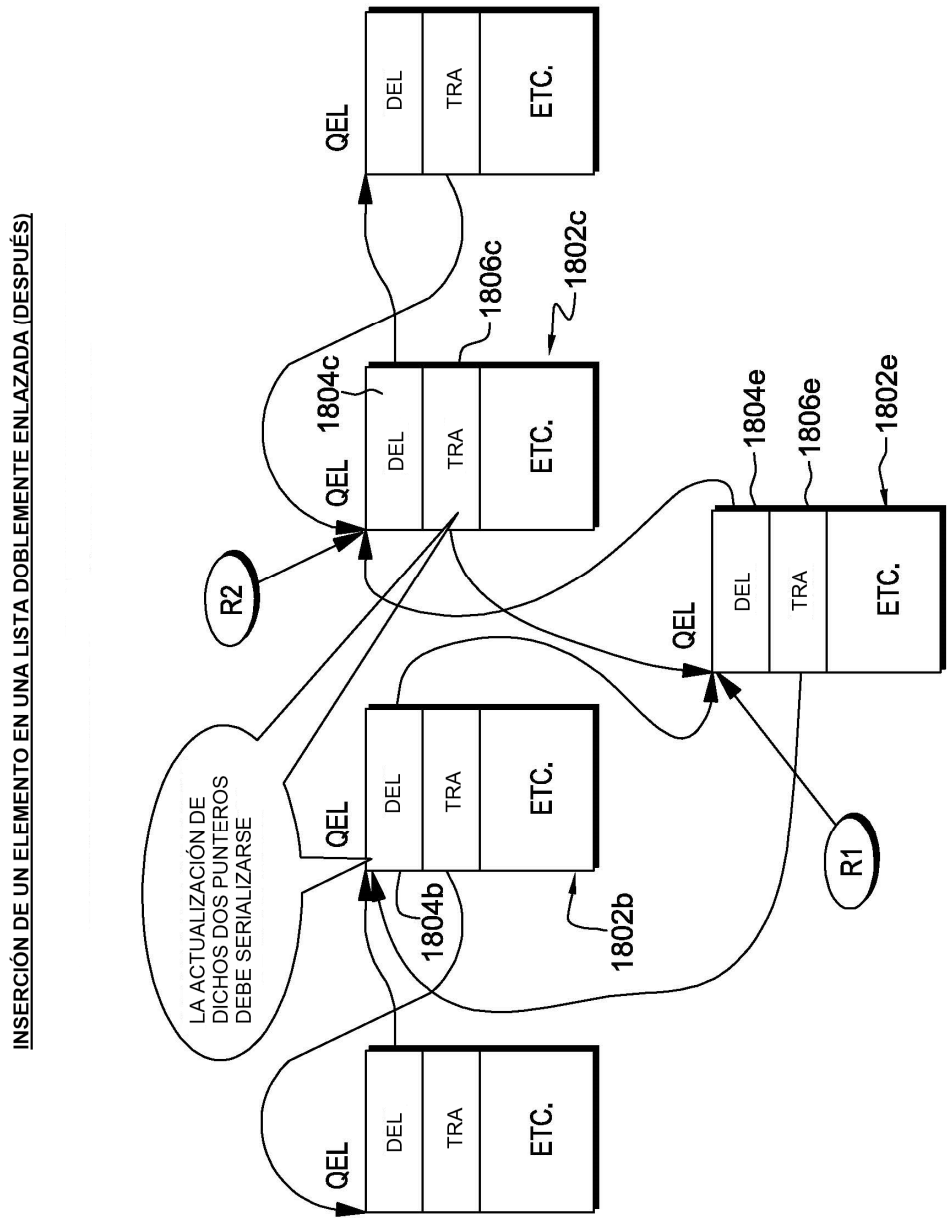
INSERCIÓN DE UN ELEMENTO EN UNA LISTA DOBLEMENTE ENLAZADA (DESPUÉS)
QEL
DEL
TRA
ETC.
1806c
1802c
1804c
R2
1804e
1806e
1802e
LA ACTUALIZACIÓN DE DICHOS DOS PUNTEROS DEBE SERIALIZARSE
1804b
1802b
R1

FIG. 18B

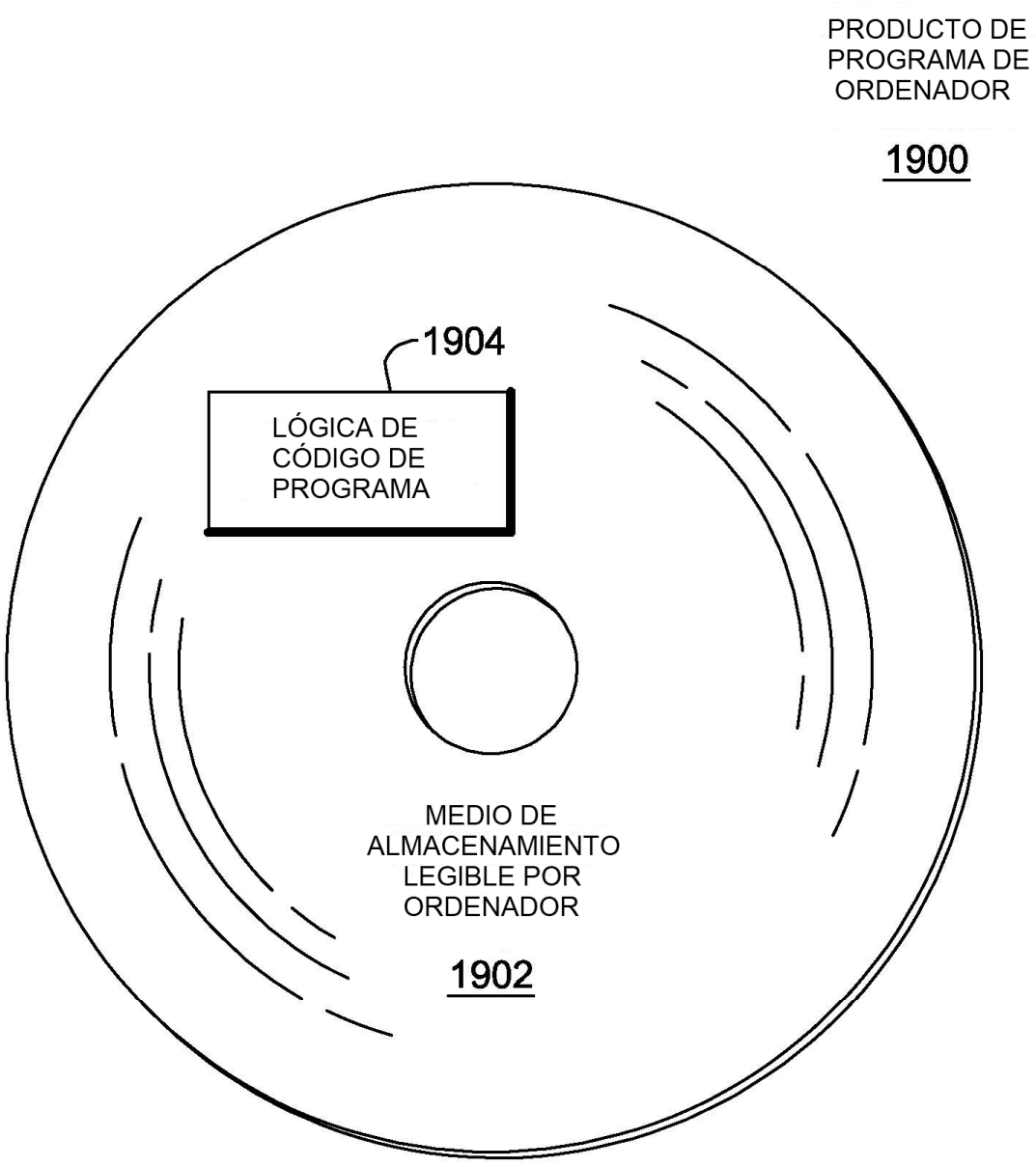
PRODUCTO DE PROGRAMA DE ORDENADOR
1900
1904
LÓGICA DE CÓDIGO DE PROGRAMA
MEDIO DE ALMACENAMIENTO LEGIBLE POR ORDENADOR
1902

FIG. 19

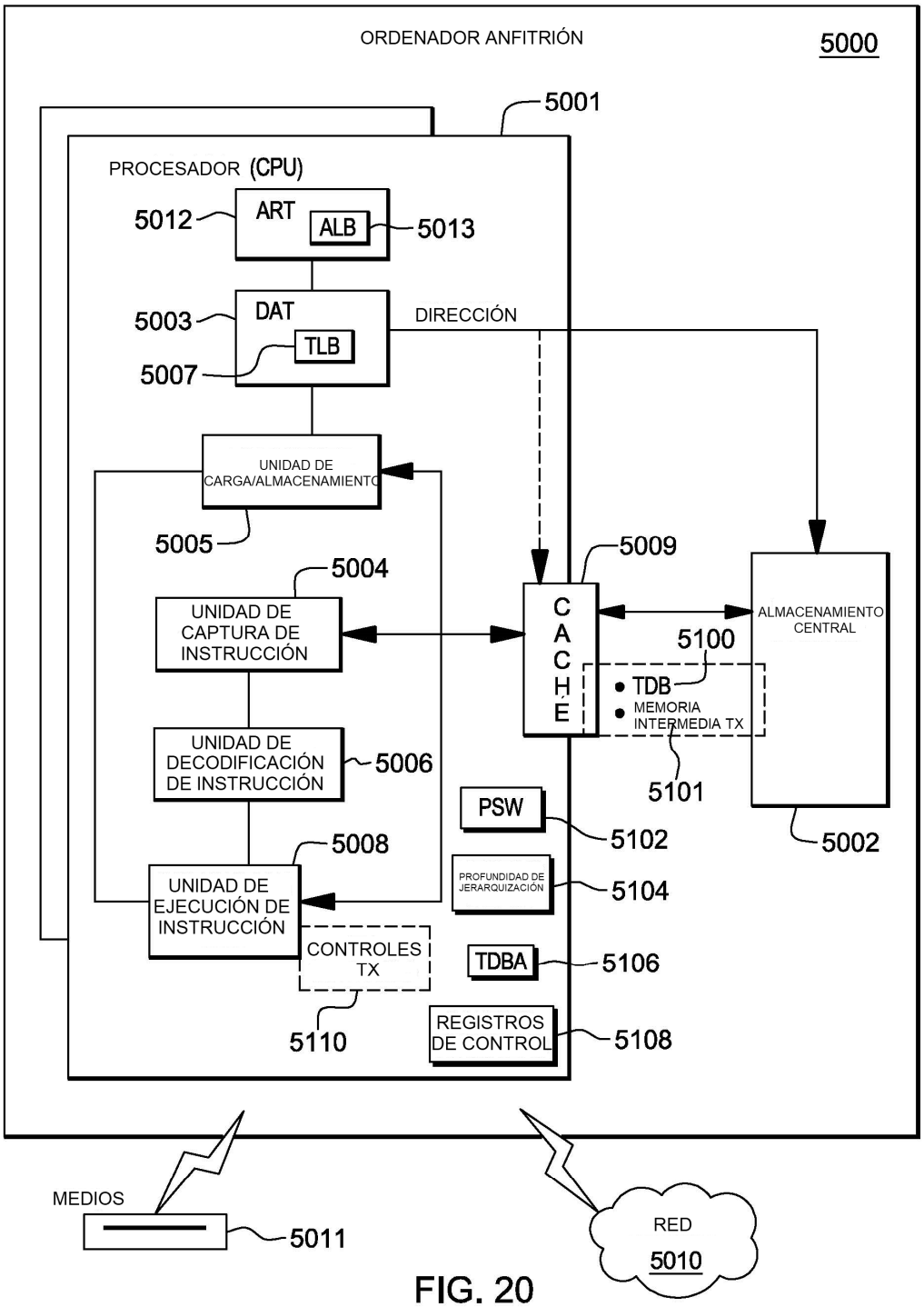
ORDENADOR ANFITRIÓN
5000
5001
PROCESADOR (CPU)
5012
ART
ALB
5013
5003
DAT
TLB
5007
DIRECCIÓN
UNIDAD DE CARGA/ALMACENAMIENTO
5005
5004
UNIDAD DE CAPTURA DE INSTRUCCIÓN
UNIDAD DE DECODIFICACIÓN DE INSTRUCCIÓN
5006
5008
UNIDAD DE EJECUCIÓN DE INSTRUCCIÓN
CONTROLES TX
5110
5009
CACHE
5100
TDB
MEMORIA INTERMEDIA TX
5101
ALMACENAMIENTO CENTRAL
5002
PSW
5102
PROFUNDIDAD DE JERARQUIZACIÓN
5104
TDBA
5106
REGISTROS DE CONTROL
5108
MEDIOS
5011
RED
5010

FIG. 20

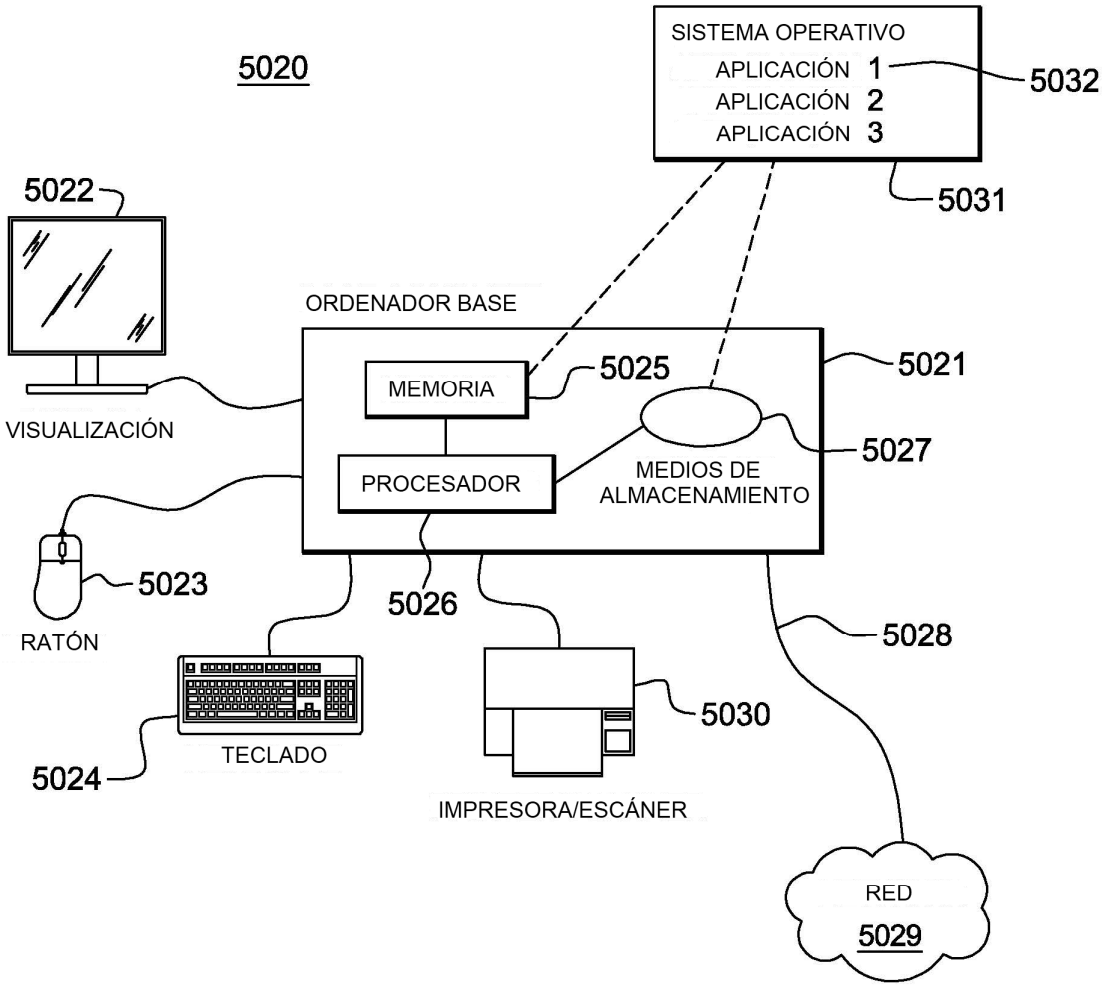
5020
SISTEMA OPERATIVO
APLICACIÓN 1
APLICACIÓN 2
APLICACIÓN 3
5032
5031
5022
ORDENADOR BASE
MEMORIA
5025
5021
5027
PROCESADOR
MEDIOS DE ALMACENAMIENTO
VISUALIZACIÓN
5023
RATÓN
5026
5028
5030
TECLADO
5024
IMPRESORA/ESCÁNER
RED
5029

FIG. 21

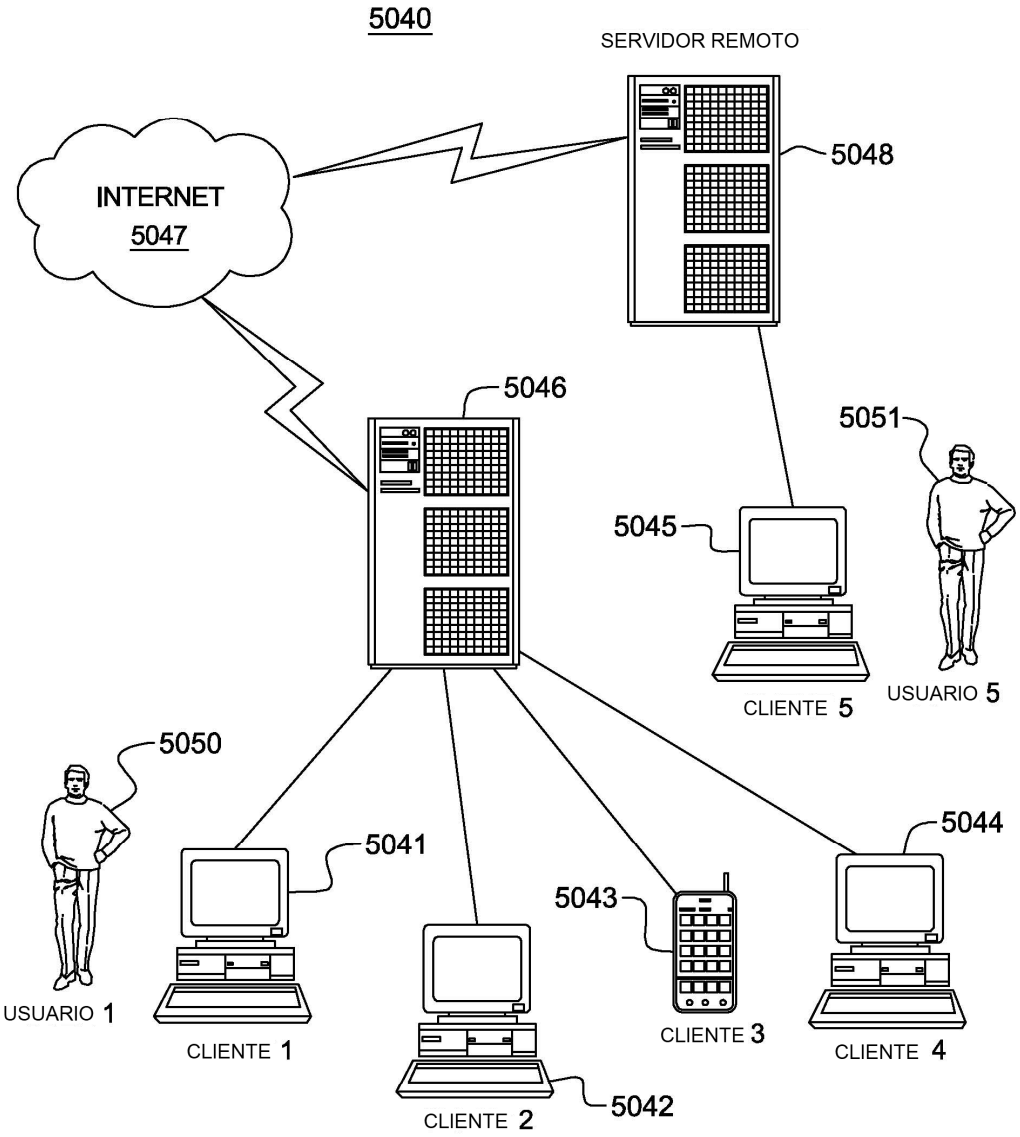
5040
SERVIDOR REMOTO
5048
INTERNET
5047
5046
5051
5045
CLIENTE 5
USUARIO 5
5050
5041
5044
5043
USUARIO 1
CLIENTE 1
CLIENTE 3
CLIENTE 4
CLIENTE 2
5042

FIG. 22

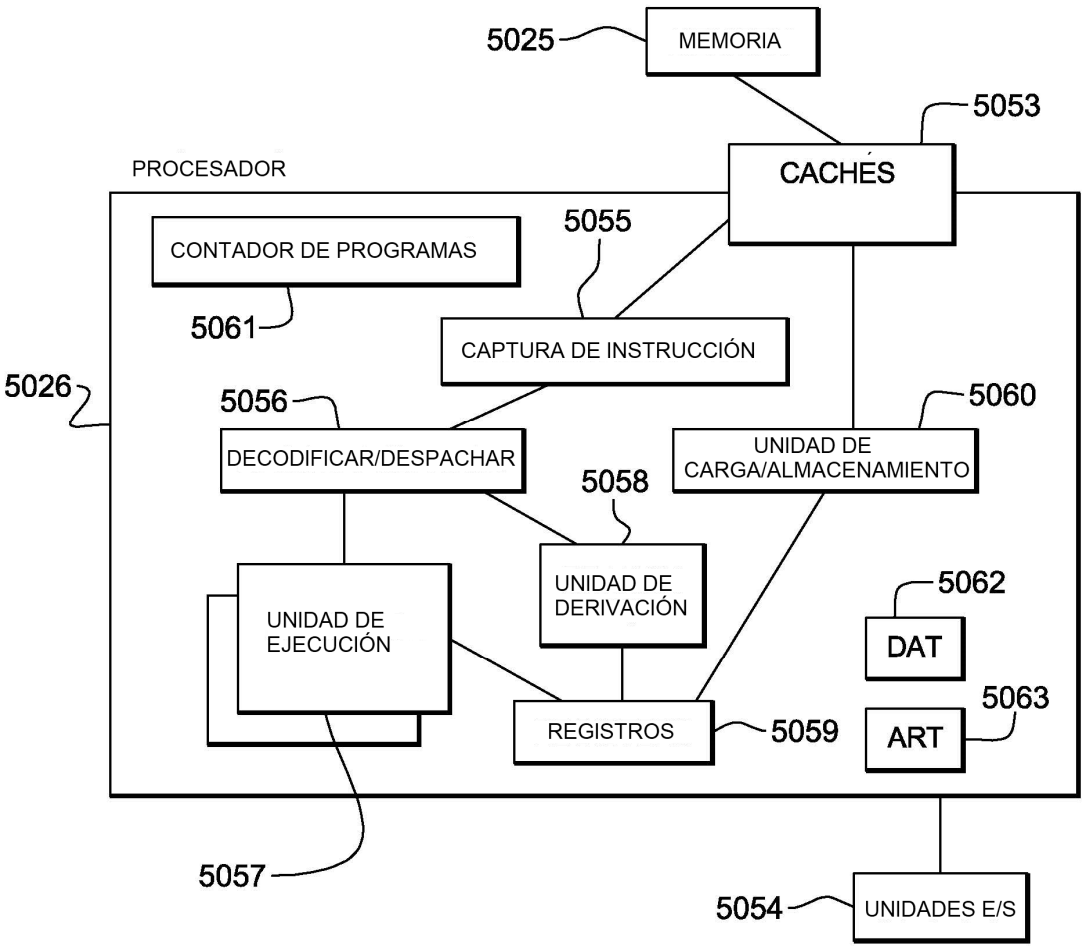
5025
MEMORIA
5053
CACHÉS
PROCESADOR
CONTADOR DE PROGRAMAS
5055
5061
CAPTURA DE INSTRUCCIÓN
5026
5056
5060
DECODIFICAR/DESPACHAR
UNIDAD DE CARGA/ALMACENAMIENTO
5058
UNIDAD DE DERIVACIÓN
UNIDAD DE EJECUCIÓN
5062
DAT
5063
ART
REGISTROS
5059
5057
5054
UNIDADES E/S

FIG. 23

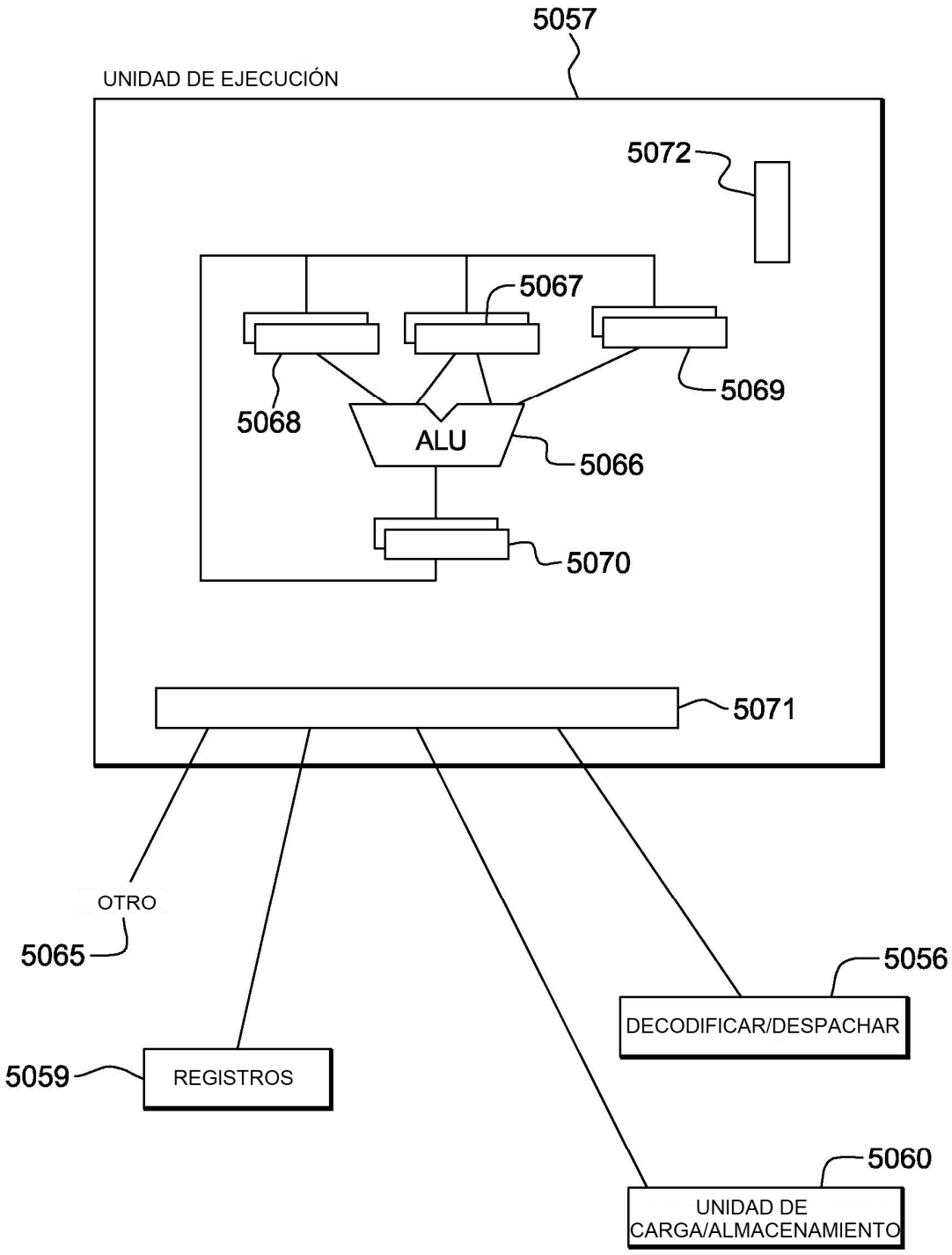
5057
UNIDAD DE EJECUCIÓN
5072
5067
5068
5069
ALU
5066
5070
5071
OTRO
5065
5056
DECODIFICAR/DESPACHAR
5059
REGISTROS
5060
UNIDAD DE CARGA/ALMACENAMIENTO

FIG. 24A

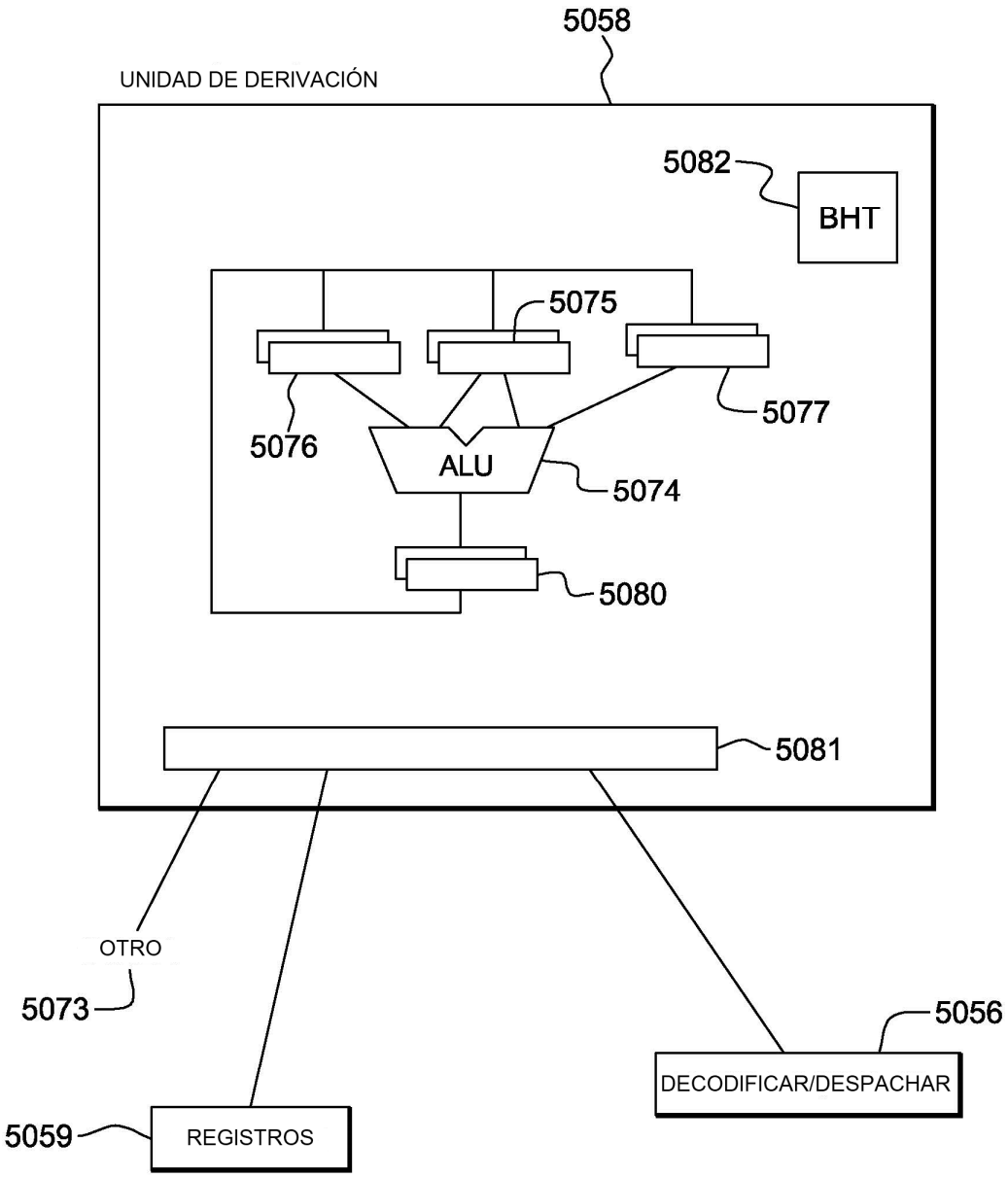
5058
UNIDAD DE DERIVACIÓN
5082
BHT
5075
5077
5076
ALU
5074
5080
5081
OTRO
5073
5056
DECODIFICAR/DESPACHAR
5059
REGISTROS

FIG. 24B

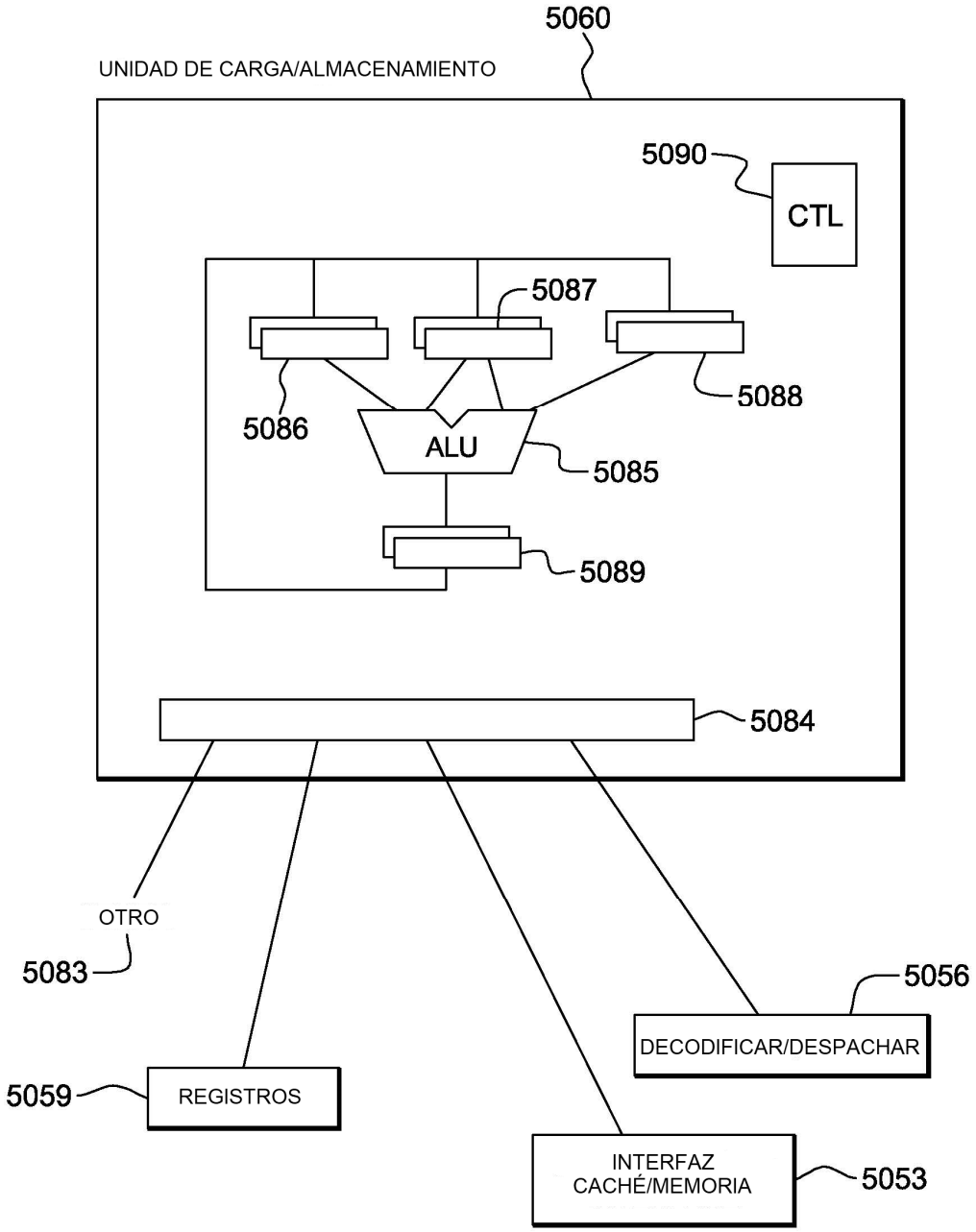
5060
UNIDAD DE CARGA/ALMACENAMIENTO
5090
CTL
5087
5086
5088
ALU
5085
5089
5084
OTRO
5083
5056
DECODIFICAR/DESPACHAR
5059
REGISTROS
INTERFAZ
CACHÉ/MEMORIA
5053

FIG. 24C

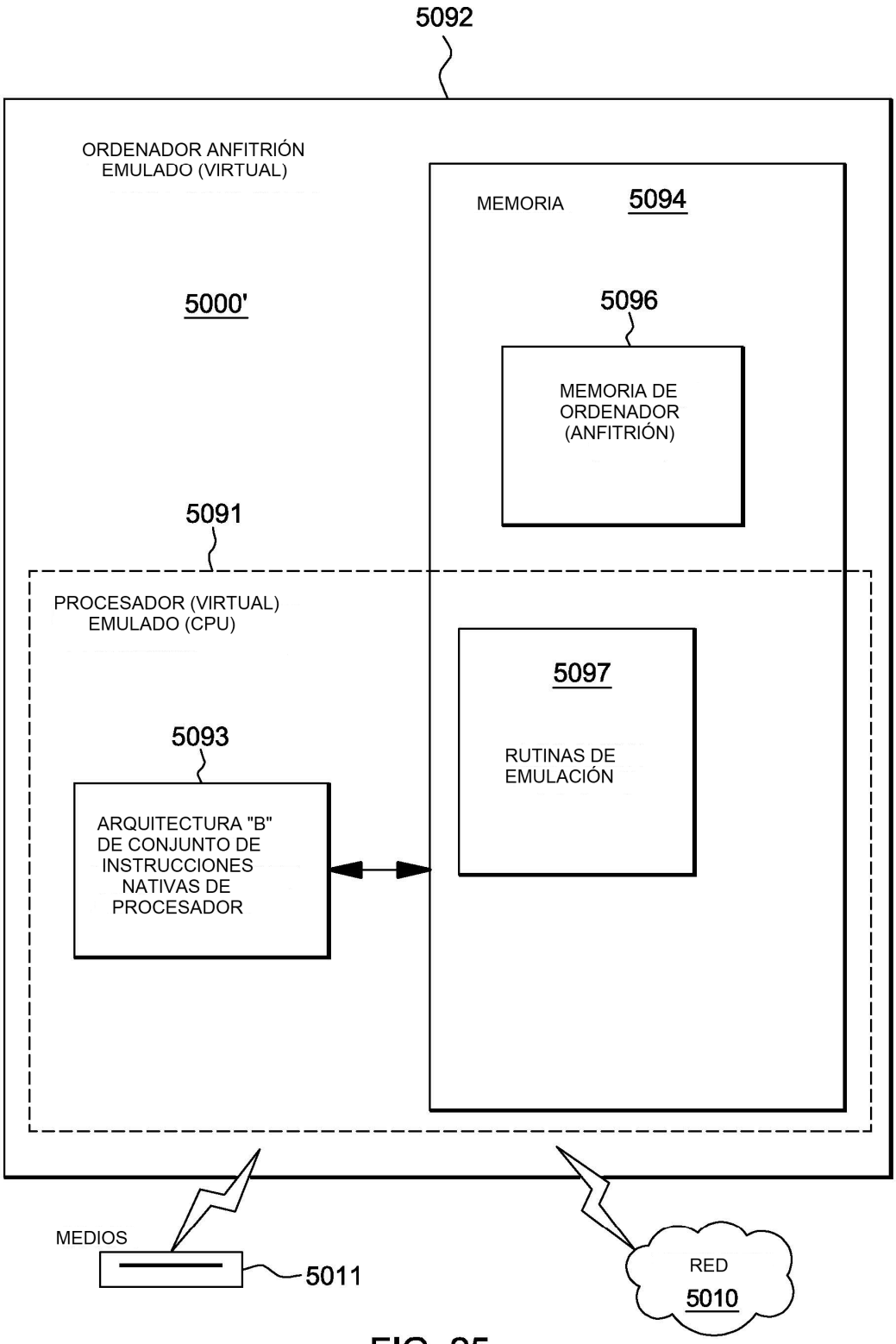
5092
ORDENADOR ANFITRIÓN EMULADO (VIRTUAL)
MEMORIA
5094
5000'
5096
MEMORIA DE ORDENADOR (ANFITRIÓN)
5091
PROCESADOR (VIRTUAL) EMULADO (CPU)
5097
RUTINAS DE EMULACIÓN
5093
ARQUITECTURA "B" DE CONJUNTO DE INSTRUCCIONES NATIVAS DE PROCESADOR
MEDIOS
5011
RED
5010

FIG. 25