(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 717 480**

(51) Int. CI.:

G06F 12/00 (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: **12.06.2013 PCT/IB2013/054813**

(87) Fecha y número de publicación internacional: **19.12.2013 WO13186722**

(96) Fecha de presentación y número de la solicitud europea: **12.06.2013 E 13805110 (7)**

(97) Fecha y número de publicación de la concesión europea: **27.02.2019 EP 2862071**

(54) Título: **Control de manera selectiva de ejecución de instrucciones en procesamiento transaccional**

(30) Prioridad:

15.06.2012 US 201213524898

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
21.06.2019

(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
ROGERS, ROBERT

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Control de manera selectiva de ejecución de instrucciones en procesamiento transaccional

Antecedentes

Uno o más aspectos se relacionan, en general, con entornos informáticos multiproceso, y en particular, con procesamiento transaccional dentro de tales entornos informáticos.

Un desafío permanente en la programación multiprocesador es el de las actualizaciones a la misma ubicación de almacenamiento por múltiples unidades centrales de procesamiento (CPU). Muchas instrucciones que actualizan las ubicaciones de almacenamiento, incluso operaciones lógicas simples, tales como AND, lo hacen así con múltiples accesos a la ubicación. Por ejemplo, primero, se obtiene la ubicación de almacenamiento, y luego, el resultado actualizado se almacena de nuevo.

Con el fin de que múltiples CPU actualicen de manera segura la misma ubicación de almacenamiento, se serializa el acceso a la ubicación. Una instrucción, la instrucción PROBAR Y ESTABLECER, introducida con la arquitectura S/360 anteriormente ofrecida por International Business Machines Corporation, proporcionaba una actualización entrelazada de una ubicación de almacenamiento. Actualización entrelazada significa que, como se observa por otras CPU y en el subsistema de entrada/salida (I/O) (por ejemplo, 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 COMPARAR E INTERCAMBIAR y COMPARAR DOBLE E INTERCAMBIAR que proporcionan unos medios más sofisticados de realización de actualización entrelazada, y permiten la implementación de lo que se conoce comúnmente como una palabra de bloqueo (o semáforo). Instrucciones añadidas recientemente han proporcionado capacidades adicionales de actualización entrelazada, incluyendo COMPARAR E INTERCAMBIAR Y PURGAR, y COMPARAR E INTERCAMBIAR Y ALMACENAR. No obstante, todas estas instrucciones proporcionan entrelazado solamente para una única ubicación de almacenamiento.

Técnicas de programa más complejas pueden requerir la actualización entrelazada de múltiples ubicaciones de almacenamiento, tal como cuando se añade un elemento a una lista doblemente enlazada. En tal operación, un puntero tanto hacia adelante como hacia atrás ha de parecer que se actualiza simultáneamente, como se observa por otras CPU y el subsistema de I/O. Con el fin de efectuar tal actualización de ubicación múltiple, se fuerza a que el programa use un punto de serialización separado y único, tal como una palabra de bloqueo. No obstante, las palabras de bloqueo pueden proporcionar un nivel de serialización mucho más alto que se garantiza; por ejemplo, las palabras de bloqueo pueden serializar una cola entera de millones de elementos, aún cuando solamente se estén actualizando dos elementos. El programa puede estructurar los datos para usar una serialización de grano más fino (por ejemplo, una jerarquía de puntos de bloqueo), pero eso introduce problemas adicionales, tales como situaciones de punto muerto potenciales si se viola la jerarquía, y problemas de recuperación si el programa encuentra un error mientras que mantiene uno o más bloqueos o si no se puede adquirir el bloqueo.

Además de lo anterior, hay numerosos escenarios donde un programa puede ejecutar una secuencia de instrucciones que pueden dar o no como resultado una condición de excepción. Si no ocurre una condición de excepción, entonces el programa continúa; no obstante, si se reconoce una excepción, entonces el programa puede tomar medidas correctivas para eliminar la condición de excepción. Java, como ejemplo, puede explotar tal ejecución, por ejemplo, en ejecución especulativa, alineación parcial de una función y/o en la secuenciación de nuevo de comprobación de puntero nulo.

En entornos de sistemas operativos clásicos, tales como z/OS y sus predecesores ofrecidos por International Business Machines Corporation, el programa establece un entorno de recuperación para interceptar cualquier condición de programa de excepción que pueda encontrar. Si el programa no intercepta la excepción, el sistema operativo típicamente termina de manera anormal el programa por excepciones que el sistema operativo no está preparado para manejar. Establecer y explotar tal entorno es costoso y complicado.

La técnica anterior siguiente es relevante como material de antecedentes.

El documento US 2010/023703 A1 (CHRISTIE DAVID S [US] ET AL) 28 de enero de 2010 describe soporte de memoria transaccional de hardware para accesos de memoria compartida protegidos y no protegidos en una sección especulativa.

DAVE CHRISTIE ET AL: “Evaluation of AMD’s advanced synchronization facility within a complete transactional memory stack”, ACTAS DE LA 5ª CONFERENCIA EUROPEA SOBRE SISTEMAS INFORMÁTICOS, EUROSYS ‘10, 1 de enero de 2010 (01-01-2010), página 27, XP055163245, Nueva York, Nueva York, EE.UU. describe antecedentes útiles.

El documento US 2010/205608 A1 (NEMIROVSKY MARIO D [US] ET AL) 12 de agosto de 2010 describe un mecanismo para gestionar el bloqueo de recursos en un entorno de múltiples subprocesos.

El documento US 5 471 591 A (EDMONDSON JOHN H [US] ET AL) 28 de noviembre de 1995 describe un cuadro de indicadores combinado de cola de escritura de operando y de dependencia de lectura después de escritura.

El documento US 6 754 809 B1 (GUTTAG KARL M [US] ET AL) 22 de junio de 2004 (22-06-2004) describe un aparato de procesamiento de datos con acceso indirecto a archivos de registro.

El documento US 2007/162520 A1 (PETERSEN LEAF [US] ET AL) 12 de julio de 2007 describe transacciones de hardware anidadas asistidas por software.

El documento US 6 035 313 A (MARCHANT JEFFREY DAVID [US]) 7 de marzo de 2000 describe un generador de direcciones de memoria para una Transformada de Fourier Rápida.

Breve compendio

La invención proporciona un método de control de ejecución de instrucciones dentro de transacciones de un entorno informático, como se reivindica, y el sistema y programa de ordenador correspondientes.

Breve descripción de las diversas vistas de los dibujos

Uno o más aspectos se apuntan particularmente y reivindican claramente como ejemplos en las reivindicaciones a la conclusión de la especificación. Lo precedente y otros objetos, características y ventajas son evidentes a partir de la siguiente descripción detallada tomada junto con los dibujos que se acompañan, en los cuales:

la FIG. 1 representa una realización de un entorno informático;

la FIG. 2A representa un ejemplo de una instrucción Comienzo de Transacción (TBEGIN);

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

la FIG. 3A representa en el ejemplo de una instrucción Comienzo de Transacción restringida (TBEGINC);

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

la FIG. 4 muestra un ejemplo de una instrucción Fin de Transacción (TEND);

la FIG. 5 representa un ejemplo de una instrucción Abortar de Transacción (TABORT);

la FIG. 6 representa un ejemplo de transacciones anidadas;

la FIG. 7 representa un ejemplo de una instrucción de ALMACÉN NO TRANSACCIONAL (NTSTG);

la FIG. 8 representa un ejemplo de una instrucción de EXTRAER PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIONES (ETND);

la FIG. 9 representa un ejemplo de un bloque de diagnóstico de transacciones;

la FIG. 10 representa razones de ejemplo para abortar, junto con códigos de abortar y códigos de condición asociados;

la FIG. 11 representa una realización de la lógica asociada con la ejecución de una instrucción TBEGINC;

la FIG. 12 representa una realización de la lógica asociada con ejecutar una instrucción TBEGIN;

la FIG. 13 representa una realización de la lógica asociada con ejecutar una instrucción TEND;

la FIG. 14 muestra un ejemplo de la lógica para actualizar controles para permitir selectivamente instrucciones restringidas;

las FIG. 15A-15B representan un ejemplo de inserción de un elemento de cola en una lista doblemente enlazada de elementos de cola;

la FIG. 16 representa una realización de un producto de programa de ordenador;

la FIG. 17 representa una realización de un sistema de ordenador central;

la FIG. 18 representa un ejemplo adicional de un sistema informático;

la FIG. 19 representa otro ejemplo de un sistema informático que comprende una red informática;

la FIG. 20 representa una realización de diversos elementos de un sistema informático;

la FIG. 21A representa una realización de la unidad de ejecución del sistema informático de la FIG. 20;

la FIG. 21B representa una realización de la unidad de ramificación del sistema informático de la FIG. 20;

la FIG. 21C representa una realización de la unidad de carga/almacenamiento del sistema informático de la FIG. 20; y

la FIG. 22 representa una realización de un sistema de ordenador central emulado.

Descripción detallada

Según un aspecto, se proporciona una facilidad de ejecución transaccional (TX). Esta facilidad proporciona procesamiento transaccional para las instrucciones, y en una o más realizaciones, ofrece diferentes modos de ejecución, como se describe a continuación, así como niveles anidados de procesamiento transaccional.

La facilidad de ejecución transaccional introduce un estado de CPU llamado modo de ejecución transaccional (TX). Siguiendo a un reinicio de la CPU, la CPU no está en el modo TX. La CPU entra en el modo TX por una instrucción COMIENZO DE TRANSACCIÓN. La CPU abandona el modo TX por o bien (a) una instrucción de FIN DE TRANSACCIÓN más externa (más detalles sobre interna y externa próximamente), o bien (b) la transacción que se aborta. Mientras que está en el modo TX, los accesos de almacenamiento por la CPU parecen ser bloques concurrentes, como se observa por otras CPU y el subsistema de I/O. Los accesos de almacenamiento están o bien (a) comprometidos con el almacenamiento cuando termina la transacción más externa sin abortar (es decir, por ejemplo, actualizaciones hechas en memoria caché o almacenamiento temporal local para la CPU se propagan y almacenan en memoria real y visibles por otras CPU), o bien (b) se descartan si la transacción se aborta.

Las transacciones se pueden anidar. Es decir, mientras que la CPU está en el modo TX, puede ejecutar otra instrucción COMIENZO DE TRANSACCIÓN. La instrucción que hace que la CPU entre en el modo TX se llama el COMIENZO DE TRANSACCIÓN más externa; de manera similar, se dice que el programa está en la transacción más externa. Las ejecuciones posteriores de COMIENZO DE TRANSACCIÓN se llaman instrucciones internas; y el programa está ejecutando una transacción interna. El modelo proporciona una profundidad de anidamiento mínima y una profundidad de anidamiento máxima dependiente del modelo. Una instrucción EXTRAER PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIONES devuelve el valor de profundidad de anidamiento actual y, en una realización adicional, puede devolver un valor de profundidad de anidamiento máximo. Esta técnica usa un modelo llamado “anidado aplanado” en el que una condición de abortar en cualquier profundidad de anidamiento hace que todos los niveles de la transacción se aborten, y el control se devuelva a la instrucción que sigue a COMENZAR TRANSACCIÓN más externa.

Durante el procesamiento de una transacción, se dice que un acceso transaccional hecho por una CPU entra en conflicto con o bien (a) un acceso transaccional o un acceso no transaccional hecho por otra CPU, o bien (b) un acceso no transaccional hecho por el subsistema de I/O, si ambos los accesos son a cualquier ubicación dentro de la misma línea de caché, y uno o ambos de los accesos son un almacén. En otras palabras, con el fin de que la ejecución transaccional sea productiva, la CPU no ha de ser observada haciendo accesos transaccionales hasta que se comprometa. Este 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. No obstante, puede ser menos eficaz, si hay mucha disputa por las ubicaciones de almacenamiento que están siendo accedidas transaccionalmente.

En un modelo de ejecución transaccional (conocido en la presente memoria como una transacción no restringida), cuando se aborta una transacción, el programa puede o bien intentar reconducir la transacción con la esperanza de que la condición de abortar ya no esté presente, o bien el programa puede “retroceder” a un camino no transaccional equivalente. En otro modelo de ejecución transaccional (conocido en la presente memoria como transacción restringida), una transacción abortada se reconduce automáticamente por la CPU; en ausencia de violaciones de restricción, la transacción restringida tiene asegurada la terminación final.

Cuando se inicia una transacción, el programa puede especificar diversos controles, tales como (a) qué registros generales se restauran a sus contenidos originales si se aborta la transacción, (b) si se permite que la transacción modifique el contexto de registro de punto flotante, incluyendo, por ejemplo, los registros de punto flotante y el registro de control de punto flotante, (c) si se permite que la transacción modifique los registros de acceso (AR), y (d) si se ha de bloquear que ciertas condiciones de excepción de programa causen una interrupción. Si se aborta una transacción no restringida, se puede proporcionar diversa información de diagnóstico. Por ejemplo, la instrucción TBEGIN más externa que inicia una transacción no restringida puede designar un bloque de diagnóstico de transacciones (TDB) especificado por programa. Además, el TDB en el área de prefijo de la CPU o designado por la descripción del estado del ordenador central también se puede usar si se aborta la transacción debido a una interrupción del programa o una condición que hace que la ejecución interpretativa termine, respectivamente.

Están indicados anteriormente diversos tipos de registros. Éstos se explican con más detalle en la presente memoria. Los registros generales se pueden usar como acumuladores en operaciones aritméticas y lógicas generales. En una realización, cada registro contiene posiciones de 64 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 proporcionan direccionamiento de múltiples registros generales teniendo varios campos R. Para algunas instrucciones, el uso de un registro general específico es implícito en lugar de ser 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 se usan también como dirección base y registros de índice en la generación de direcciones. En estos casos, los registros se designan mediante un campo B de cuatro bits o un campo X en una instrucción. Un valor de cero en el campo B o X especifica que no ha de ser aplicada ninguna base o índice y, de este modo, el registro general 0 no ha de ser designado como que contiene una dirección base o índice.

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 las instrucciones de punto flotante. Cada registro de punto flotante es de 64 bits de longitud y puede contener o bien un operando de punto flotante corto (32 bits) o bien uno largo (64 bits).

Un registro de control de punto flotante (FPC) 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 que tiene posiciones de 64 bits. Las posiciones de bit en los registros se asignan a facilidades particulares en el sistema, tales como la Grabación de Eventos de Programa (PER) (que se trata a continuación), y se usan o bien para especificar que puede tener lugar una operación o bien para proporcionar 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), como se describe a continuación.

La CPU tiene, por ejemplo, 16 registros de acceso numerados 0-15. Un registro de acceso consta de posiciones de 32 bits que contienen una especificación indirecta de un elemento de control de espacio de direcciones (ASCE). Un elemento de control de espacio de direcciones es un parámetro usado por el mecanismo de traducción dinámica de direcciones (DAT) para traducir referencias a un espacio de direcciones 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)), 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 direcciones especificado por el registro de acceso se usa por DAT para la referencia que se hace. Para algunas instrucciones, se usa un campo R en lugar de un campo B. Se proporcionan 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 direcciones. El registro de acceso 0 designa el espacio de instrucciones primario. Cuando uno de los registros de acceso 1-15 se usa para designar un espacio de direcciones, la CPU determina qué espacio de direcciones se designa traduciendo los contenidos del registro de acceso. Cuando el registro de acceso 0 se usa para designar un espacio de direcciones, la CPU trata el registro de acceso como que designa un espacio de instrucciones 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 instrucciones primario y un máximo de otros 15 espacios.

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

Por ejemplo, en la z/Architecture, cuando se usa una dirección virtual por una CPU para acceder al almacenamiento principal (también conocido como memoria principal), primero se convierte, por medio de traducción dinámica de direcciones (DAT), a una dirección real, y luego, por medio de prefijos, a 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 direcciones específico se denomina elemento de control de espacio de direcciones, y se encuentra para su uso por DAT en un registro de control o como se especifica por un registro de acceso. Alternativamente, el elemento de control de espacio de direcciones para un espacio de direcciones puede ser una designación de espacio real, que indica que DAT ha de traducir la dirección virtual simplemente tratándola como una dirección real y sin usar ninguna tabla.

DAT usa, en diferentes momentos, los elementos de control de espacio de direcciones en diferentes registros de control o especificados por los registros de acceso. La elección se determina por el modo de traducción especificado en la PSW actual. Están disponibles cuatro modos de traducción: modo de espacio primario, modo de espacio secundario, modo de registro de acceso y modo de espacio inicial. Diferentes espacios de direcciones son direccionables dependiendo del modo de traducción.

En cualquier instante cuando la CPU está en el modo de espacio primario o en el modo de espacio secundario, la CPU puede traducir las direcciones virtuales que pertenecen a dos espacios de direcciones – el espacio de

direcciones primario y el segundo espacio de direcciones. En cualquier instante cuando la CPU está en el modo de registro de acceso, puede traducir direcciones virtuales de hasta 16 espacios de direcciones - el espacio de direcciones primario y hasta 15 espacios de direcciones especificados por AR. En cualquier instante cuando la CPU esté en el modo de espacio de inicio, puede traducir direcciones virtuales del espacio de direcciones de inicio.

El espacio de direcciones primario se identifica como tal porque consta de direcciones virtuales primarias, que se traducen por medio del elemento de control de espacio de direcciones (ASCE) primario. De manera similar, el espacio de direcciones secundario consiste en direcciones virtuales secundarias traducidas por medio del ASCE secundario; los espacios de direcciones especificados por AR consisten en direcciones virtuales especificadas por AR traducidas por medio de los ASCE especificados por AR; y el espacio de direcciones de inicio consiste en direcciones virtuales de inicio traducidas por medio del ASCE de inicio. Los ASCE primario y secundario están en los registros de control 1 y 7, respectivamente. Los ASCE especificados por AR están en las entradas de la segunda tabla de ASN que se sitúan a través de un proceso llamado traducción de registros de acceso (ART) que usa los registros de control 2, 5 y 8. El ASCE de inicio 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 FIG. 1.

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

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

Como ejemplo, el entorno informático 100 incluye un complejo de procesador de ordenador central (CPC) 102 acoplado a uno o más dispositivos de entrada/salida (I/O) 106 a través de una o más unidades de control 108. El complejo de procesador de ordenador central 102 incluye, por ejemplo, uno o más procesadores centrales 110, una o más particiones 112 (por ejemplo, particiones lógicas (LP)), un hipervisor de particiones lógicas 114 y un subsistema de entrada/salida 115, cada uno de los cuales se describe a continuación.

Los procesadores centrales 110 son recursos de procesador físicos asignados a las particiones lógicas. En particular, cada partición lógica 112 tiene uno o más procesadores lógicos, cada uno de los cuales representa todo o una parte de un procesador físico 110 asignado a la partición. Los procesadores lógicos de una partición particular 112 pueden estar o bien dedicados a la partición, de modo que el recurso de procesador subyacente 110 esté reservado para esa partición; o bien compartidos con otra partición, de modo que el recurso de procesador subyacente esté potencialmente disponible para otra partición.

Una partición lógica funciona como un sistema separado y tiene una o más aplicaciones y, opcionalmente, un sistema operativo residente en la misma, que puede diferir para cada partición lógica. En una realización, 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 mediante un hipervisor de partición lógica 114, que se implementa mediante el microprograma que se ejecuta en los procesadores 110. Como se usa en la presente memoria, el microprograma incluye, por ejemplo, el microcódigo y/o el milicódigo del procesador. Incluye, por ejemplo, las instrucciones a nivel de hardware y/o las estructuras de datos usadas en la implementación de código máquina de nivel más alto. En una realización, incluye, por ejemplo, un código propietario que se entrega típicamente como microcódigo que incluye un software de confianza o un microcódigo específico para el hardware subyacente y controla el acceso del sistema operativo al hardware del sistema.

Las particiones lógicas y el hipervisor de partición lógica comprenden cada uno, uno o más programas que residen en particiones respectivas de almacenamiento central asociado con los procesadores centrales.

Un ejemplo de hipervisor de partición lógica 114 es el Processor Resource/System Manager (PR/SM), ofrecido por International Business Machines Corporation, Armonk, Nueva York.

El subsistema de entrada/salida 115 dirige el flujo de información entre los dispositivos de entrada/salida 106 y el almacenamiento principal (también conocido como memoria principal). Está acoplado al complejo de procesamiento central, en el sentido de que puede ser parte del complejo de procesamiento central o estar separado del mismo. El subsistema de I/O libera 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 proporcionar comunicaciones, el subsistema de I/O emplea adaptadores de comunicaciones de I/O. Hay diversos tipos de adaptadores de comunicaciones incluyendo, por ejemplo, canales, adaptadores de I/O, tarjetas PCI, tarjetas Ethernet, tarjetas de Pequeña Interfaz de Almacenamiento Informático (SCSI), etc. En el ejemplo particular descrito en la presente memoria, los adaptadores de comunicaciones de I/O son canales y, por lo

tanto, el subsistema de I/O se conoce en la presente memoria como subsistema de canal. No obstante, éste es solamente un ejemplo. Se pueden usar otros tipos de subsistemas de I/O.

El subsistema de I/O usa uno o más caminos de entrada/salida como enlaces de comunicación en la gestión del flujo de información hacia o desde los dispositivos de entrada/salida 106. En este ejemplo particular, estos caminos se denominan caminos de canal, dado que los adaptadores de comunicación son canales.

El entorno informático descrito anteriormente es solamente un ejemplo de un entorno informático que se puede usar. Se pueden usar otros entornos, incluyendo, pero no limitados a, entornos no particionados, otros entornos particionados y/o entornos emulados; las realizaciones no están limitadas a ningún entorno.

Según uno o más aspectos, la facilidad de ejecución transaccional es una mejora de la CPU que proporciona los medios por los cuales la CPU puede ejecutar una secuencia de instrucciones - conocida como transacción - que puede acceder a múltiples ubicaciones de almacenamiento, incluyendo la actualización de esas ubicaciones. Como se observa por otras CPU y el subsistema de I/O, la transacción o bien (a) se completa en su totalidad como una única operación atómica, o bien (b) se aborta, no dejando potencialmente ninguna evidencia de que alguna vez se ejecutó (excepto para ciertas condiciones descritas en la presente memoria). De este modo, una transacción completada con éxito puede actualizar numerosas ubicaciones de almacenamiento sin ningún 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, incluyendo ejecución restringida y no restringida; y procesamiento de abortar, cada uno de los cuales se describe además a continuación.

En una realización, tres controles de propósito especial, incluyendo una Palabra de Estado de Programa (PSW) de abortar transacción, una dirección de bloque de diagnóstico de transacciones (TDB) y una profundidad de anidamiento de transacciones; cinco bits de registro de control; y seis instrucciones generales, incluyendo COMIENZO DE TRANSACCIÓN (restringida y no restringida), FIN DE TRANSACCIÓN, EXTRAER PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIONES, ABORTAR TRANSACCIÓN y ALMACÉN NO TRANSACCIONAL, se usan para controlar la facilidad de ejecución transaccional. Cuando la facilidad está instalada, se instala, por ejemplo, en todas las CPU en la configuración. Una indicación de facilidad, el bit 73 en una implementación, cuando hay una, indica que la facilidad de ejecución transaccional está instalada.

Cuando la facilidad de ejecución transaccional está instalada, la configuración proporciona una facilidad de ejecución transaccional no restringida y, opcionalmente, una facilidad de ejecución transaccional restringida, cada una de las cuales se describe a continuación. Cuando las indicaciones de facilidad 50 y 73, como ejemplos, son ambas uno, la facilidad de ejecución transaccional restringida está instalada. Ambas indicaciones de facilidad se almacenan en la memoria en ubicaciones especificadas.

Como se usa en la presente memoria, el nombre de la instrucción COMIENZO DE TRANSACTION se refiere a las instrucciones que tienen los mnemotécnicos TBEGIN (Comienzo de Transacción para una transacción no restringida) y TBEGINC (Comienzo de Transacción para una transacción restringida). Las discusiones pertenecientes a una instrucción específica se indican por el nombre de la instrucción seguido del mnemotécnico entre paréntesis o corchetes, o simplemente por el mnemotécnico.

Una realización de un formato de una instrucción de COMIENZO DE TRANSACCIÓN (TBEGIN) se representa en las FIG. 2A-2B. Como ejemplo, una instrucción TBEGIN 200 incluye un campo de código de operación 202 que incluye un código de operación que especifica una operación no restringida de comienzo de transacción; 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 distinto de cero, los contenidos del registro general especificado por B_1 204 se añaden a D_1 206 para obtener la dirección del primer operando.

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

- Cuando la profundidad de anidamiento de transacciones es inicialmente cero, la dirección del primer operando designa la ubicación del bloque de diagnóstico de transacciones de 256 bytes, denominado TDB especificado por TBEGIN (descrito además a continuación) en la que se puede almacenar diversa información de diagnóstico si se aborta la transacción. Cuando la CPU está en el modo de espacio primario o el modo de registro de acceso, la dirección del primer operando designa una ubicación en el espacio de direcciones primario. Cuando la CPU está en el modo de espacio secundario o espacio de inicio, la dirección del primer operando designa una ubicación en el espacio de direcciones secundario o de inicio, respectivamente. Cuando DAT está apagada, la dirección del bloque de diagnóstico de transacciones (TDB) (TDBA) designa una ubicación en el almacenamiento real.

  Se determina la accesibilidad al almacén para el primer operando. Si es accesible, la dirección lógica del operando se pone en la dirección de bloque de diagnóstico de transacciones (TDBA), y la TDBA es válida.

- Cuando la CPU ya está en el modo de ejecución transaccional no restringida, la TDBA no se modifica, y es impredecible si el primer operando se prueba para su 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 no es válida.

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

Máscara de Guardado de Registro General (GRSM) 210 (FIG. 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 registros correspondiente no se guarda. Cuando un bit en la GRSM de la instrucción TBEGIN más externa 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 restauran a sus contenidos cuando se ejecutó la instrucción TBEGIN más externa. Los contenidos de todos los otros registros generales (no guardados) no se restauran cuando una transacción se aborta.

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

Modificación de AR permitida (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 la modificación de AR permitida eficaz es el AND lógico del control A de la instrucción TBEGIN para el nivel de anidamiento actual y para todos los niveles externos.

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

Operación de punto flotante (F) permitida 214: El control F, el bit 13 del campo I_2, controla si se permite que la transacción ejecute instrucciones de punto flotante específicas. El control de operación de punto flotante permitida eficaz es el AND lógico del control F en la instrucción TBEGIN para el nivel de anidamiento actual y para todos los niveles externos.

Si el control F eficaz es cero, entonces (a) la transacción se abortará con el código de abortar 11 (instrucción restringida) si se hace 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) no se establecerá por ninguna 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 hace un intento de ejecutar una instrucción de punto flotante (en ausencia de cualquier otra condición de abortar), y (b) el DXC en el FPCR se puede establecer por una condición de excepción de programa de excepción de datos.

Control de Filtrado de Interrupción de Programa (PIFC) 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 (por ejemplo, excepción de direccionamiento, excepción de datos, excepción de operación, excepción de protección, etc.) que ocurren mientras la CPU está en el modo de ejecución transaccional dan como resultado una interrupción.

El PIFC eficaz es el valor más alto del PIFC en la instrucción TBEGIN para el nivel de anidamiento actual y para todos los niveles externos. Cuando el PIFC eficaz es cero, todas las condiciones de excepción de programa dan como resultado 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 dan como resultado 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 dan como resultado una interrupción. Un PIFC de 3 está reservado.

Los bits 8-11 del campo I_2 (bits 40-43 de la instrucción) están reservados y deberían contener ceros; de otro modo, el programa puede no operar de manera compatible en el futuro.

Una realización de un formato de una instrucción de Comienzo de Transacción restringida (TBEGINC) se describe con referencia a las FIG. 3A-3B. En un ejemplo, TBEGINC 300 incluye un campo de código de operación 302 que incluye un código de operación que especifica una operación de comienzo de 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 dirección del primer operando. No obstante, con la instrucción de comienzo de transacción restringida, la dirección del primer operando no se usa para acceder al almacenamiento. En su lugar, el campo B_1 de la instrucción incluye ceros; de otro modo, se reconoce una excepción de especificación.

En una realización, el campo I_2 incluye diversos controles, un ejemplo de los cuales se representa en la FIG. 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 registro 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 accesible directamente por el programa.

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

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

Modificación de AR permitida (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 modificación AR permitida eficaz es el AND lógico del control A en la instrucción TBEGINC para el nivel de anidamiento actual y para cualquier instrucción TBEGIN o TBEGINC externa.

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

Los bits 8-11 y 13-15 del campo I_2 (los bits 40-43 y 45-47 de la instrucción) están reservados y deberían contener ceros.

El final de una instrucción de Comienzo de Transacción se especifica mediante una instrucción FIN DE TRANSACCIÓN (TEND), un formato de la cual se representa en la FIG. 4. Como ejemplo, una instrucción TEND 400 incluye un campo de código de operación 402 que incluye un código de operación que especifica una operación de fin de transacción.

Se usan una serie de términos con respecto a la facilidad de ejecución transaccional y, por lo tanto, únicamente por conveniencia, se proporciona a continuación una lista de términos en orden alfabético. En una realización, estos términos tienen la siguiente definición:

Abortar: Una transacción se aborta cuando se termina antes de una instrucción FIN DE TRANSACCIÓN que da como resultado una profundidad de anidamiento de transacciones de cero. Cuando una transacción se aborta, ocurre lo siguiente, en una realización:

- Se descartan (es decir, no se comprometen) los accesos de almacén transaccionales hechos por cualquiera y todos los niveles de la transacción.
- Se comprometen los accesos de almacén no transaccionales hechos por cualquiera y todos los niveles de la transacción.
- Los registros designados por la máscara de guardado de registro general (GRSM) de la instrucción COMIENZO DE TRANSACCIÓN más externa se restauran a sus contenidos antes de la ejecución transaccional (es decir, a sus contenidos en la ejecución de la instrucción COMIENZO DE TRANSACCIÓN más externa). No se restauran los registros generales no designados por la máscara de guardado de registro general de la instrucción COMIENZO DE TRANSACCIÓN más externa.
- No se restauran los registros de acceso, los registros de punto flotante y el registro de control de punto flotante. Cualquier cambio hecho a estos registros durante la ejecución de la transacción se conserva cuando se aborta la transacción.

Una transacción se puede abortar debido a una variedad de razones, incluyendo el intento de ejecución de una instrucción restringida, el intento de modificación de un recurso restringido, el conflicto transaccional, exceder varios recursos de la CPU, cualquier condición de interceptación de ejecución interpretativa, cualquier interrupción, una instrucción ABORTAR TRANSACCIÓN y otras razones. Un código de abortar transacción proporciona razones específicas de por qué se puede abortar una transacción.

Un ejemplo de un formato de una instrucción ABORTAR TRANSACCIÓN (TABORT) se describe con referencia a la FIG. 5. Como ejemplo, una instrucción TABORT 500 incluye un campo de código de operación 502 que incluye un código de operación 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 distinto de cero, los contenidos del registro general especificado

por B_2 504 se añaden a D_2 506 para obtener una dirección del segundo operando; de otro modo, se forma la dirección del segundo operando únicamente a partir del campo D_2, y se ignora el campo B_2. La dirección del segundo operando no se usa para direccionar datos; en su lugar, la dirección forma el código de abortar transacción que se pone en un bloque de diagnóstico de transacciones durante el procesamiento de abortar. El cálculo de dirección para la dirección del segundo operando sigue las reglas de la aritmética de direcciones: en el modo de direccionamiento de 24 bits, los bits 0-29 se establecen en cero; en el modo de direccionamiento de 31 bits, los bits 0-32 se establecen en ceros.

Comprometer: A la terminación de una instrucción FIN DE TRANSACCIÓN más externa, la CPU compromete los accesos de almacén hechos por la transacción (es decir, la transacción más externa y cualquier nivel anidado) de manera que sean visibles por otras CPU y el subsistema de I/O. Como se observa por otras CPU y por el subsistema de I/O, todos los accesos de búsqueda y almacén hechos por todos los niveles anidados de la transacción parecen ocurrir como una única operación concurrente cuando ocurre el comprometer.

Los contenidos de los registros generales, los registros de acceso, los registros de punto flotante y el registro de control de punto flotante no se modifican por el proceso de comprometer. Cualquier cambio hecho a estos registros durante la ejecución transaccional se conserva cuando se comprometen los almacenes de la transacción.

Conflicto: Un acceso transaccional hecho por una CPU entra en conflicto con o bien (a) un acceso transaccional o un acceso no transaccional hecho por otra CPU, o bien (b) el acceso no transaccional hecho por el subsistema de I/O, si ambos accesos son a cualquier ubicación dentro de la misma línea de caché, y uno o más de los accesos son un almacén.

Un conflicto se puede detectar por la ejecución especulativa de instrucciones de una CPU, aún cuando el conflicto no se pueda detectar en la secuencia conceptual.

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

- Está disponible un subconjunto de las instrucciones generales.
- Se puede ejecutar un número limitado de instrucciones.
- Se puede acceder a un número limitado de ubicaciones de operandos de almacenamiento.
- La transacción está limitada a un único nivel de anidamiento.

En ausencia de interrupciones o conflictos repetidos con otras CPU o el subsistema de I/O, una transacción restringida se completa finalmente, de este modo no se requiere una rutina controlador de abortar. Las transacciones restringidas se describen en detalle a continuación.

Cuando se ejecuta una instrucción COMIENZO DE TRANSACCIÓN restringida (TBEGINC) mientras que 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 restringida: Cuando la profundidad de anidamiento de transacciones es cero, y una transacción se inicia por una instrucción TBEGINC, la CPU entra en el modo de ejecución transaccional restringida. Mientras que la CPU está en el modo de ejecución transaccional restringida, la profundidad de anidamiento de transacciones es uno.

Transacción anidada: Cuando la instrucción COMIENZO DE TRANSACCIÓN se emite mientras que la CPU está en el modo de ejecución transaccional no restringida, se anida la transacción.

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

Un ejemplo de transacciones anidadas se representa en la FIG. 6. Como se muestra, un primer TBEGIN 600 inicia una transacción más externa 601, TBEGIN 602 inicia una primera transacción anidada, y TBEGIN 604 inicia una segunda transacción anidada. En este ejemplo, TBEGIN 604 y TEND 606 definen una transacción más interna 608. Cuando se ejecuta TEND 610, los almacenes 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 restringida. Aunque una transacción no restringida no está limitada en la manera como una transacción restringida, todavía se puede abortar debido a una variedad de causas.

Modo de ejecución transaccional no restringida: Cuando se inicia una transacción por la instrucción TBEGIN, la CPU entra en el modo de ejecución transaccional no restringida. Mientras que la CPU está en el modo de ejecución transaccional no restringida, la profundidad de anidamiento de transacciones puede variar de uno a la profundidad de anidamiento de transacciones máxima.

Acceso no transaccional: Los accesos no transaccionales son accesos de operando de almacenamiento hechos 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 hechos por el subsistema de I/O son accesos no transaccionales. Además, la instrucción ALMACÉN NO TRANSACCIONAL se puede usar para causar un acceso de almacén no transaccional mientras la CPU está en el modo de ejecución transaccional no restringida.

Una realización de un formato de una instrucción ALMACÉN NO TRANSACCIONAL se describe con referencia a la FIG. 7. Como ejemplo, una instrucción ALMACÉN NO TRANSACCIONAL 700 incluye una pluralidad de campos de código de operación 702a, 702b que especifican un código de operación que designa una operación de almacén no transaccional; un campo de registro (R_1) 704 que especifica un registro, los contenidos del cual se denominan 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 dirección del segundo operando. Cuando cualquiera o ambos campos X_2 o B_2 son cero, el registro correspondiente no toma parte en la adición.

El primer operando de 64 bits se coloca sin cambios de manera no transaccional en la ubicación del segundo operando.

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

El segundo operando ha de ser alineado en un límite de palabra doble; de otro modo, se reconoce una excepción de especificación y se suprime la operación.

Transacción externa/más externa: Una transacción con una profundidad de anidamiento de transacciones numerada menor es una transacción externa. Una transacción con un valor de profundidad de anidamiento de transacciones de uno es la transacción más externa.

Una instrucción COMIENZO DE TRANSACCIÓN más externa es una que se ejecuta cuando la profundidad de anidamiento de transacciones es inicialmente cero. Una instrucción FIN DE TRANSACCIÓN más externa es una que hace que la profundidad de anidamiento de transacciones pase de uno a cero. Una transacción restringida es la transacción más externa, en esta realización.

Filtrado de interrupción de programa: Cuando una transacción se aborta debido a ciertas condiciones de excepción de programa, el programa puede, opcionalmente, evitar que ocurra la interrupción. Esta técnica se llama filtrado de interrupción de programa. El filtrado de interrupción de programa está sometido a la clase transaccional de la interrupción, el control de filtrado de interrupción de programa eficaz de la instrucción COMIENZO DE TRANSACCIÓN, y la anulación de filtrado de interrupción de programa de ejecución transaccional en el registro de control 0.

Transacción: Una transacción incluye los accesos de operando de almacenamiento hechos, y los registros generales seleccionados alterados, mientras que la CPU está en el modo de ejecución de transacción. Para una transacción no restringida, los accesos de operandos de almacenamiento pueden incluir tanto accesos transaccionales como accesos no transaccionales. Para una transacción restringida, los accesos de operandos de almacenamiento están limitados a los accesos transaccionales. Como se observa por otras CPU y por el subsistema de I/O, todos los accesos de operandos de almacenamiento hechos por la CPU mientras que está en el modo de ejecución de transacción parece que ocurren como una única operación concurrente. Si se aborta una transacción, se descartan los accesos de almacén transaccional, y cualquier registro designado por la máscara de guardado de registro general de la instrucción COMIENZO DE TRANSACCIÓN más externa se restaura a sus contenidos antes de la ejecución transaccional.

Accesos transaccionales: Los accesos transaccionales son accesos de operando de almacenamiento hechos mientras que la CPU está en el modo de ejecución transaccional, con la excepción de los accesos hechos por la instrucción ALMACÉN NO TRANSACCIONAL.

Modo de ejecución transaccional: El término modo de ejecución transaccional (también conocido como modo de ejecución de transacción) describe la operación común de tanto el modo de ejecución transaccional no restringida como la restringida. De este modo, cuando se describe la operación, los términos no restringida y restringida se usan para calificar el modo de ejecución transaccional.

Cuando la profundidad de anidamiento de transacciones es cero, la CPU no está en el modo de ejecución transaccional (también denominado modo de ejecución no transaccional).

Como se observa por la CPU, las búsquedas y los almacenes hechos en el modo de ejecución transaccional no son diferentes de los hechos mientras que no están en el modo de ejecución transaccional.

En una realización de la 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 anidamiento de transacciones, la dirección del bloque de diagnóstico de transacciones y la palabra de estado de programa (PSW) de abortar transacción.

Siguiendo a un reinicio de la CPU inicial, los contenidos de las posiciones de bits 8-9 del registro de control 0, las posiciones de bits 62-63 del registro de control 2 y la profundidad de anidamiento de transacciones se establecen en cero. Cuando el control de ejecución transaccional, el bit 8 del registro de control 0, es cero, la CPU no se puede poner en el modo de ejecución transaccional.

Se describen a continuación detalles adicionales con respecto a los diversos controles.

Como se ha indicado, 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 las siguientes, en una realización:

Control de ejecución transaccional (TXC): El bit 8 del registro de control cero es el control de ejecución transaccional. Este bit proporciona un mecanismo por el cual el programa de control (por ejemplo, el sistema operativo) puede indicar si la facilidad de ejecución transaccional es o no utilizable por el programa. El bit 8 ha de ser uno para entrar con éxito en el modo de ejecución transaccional.

Cuando el bit 8 del registro de control 0 es cero, el intento de ejecución de las instrucciones EXTRAER PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIONES, COMIENZO DE TRANSACCIÓN y FIN DE TRANSACCIÓN da como resultado una ejecución de operación especial.

Una realización de un formato de una instrucción EXTRAER PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIONES se describe con referencia a la FIG. 8. Como ejemplo, una instrucción EXTRAER PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIONES 800 incluye un campo de código de operación 802 que especifica un código de operación que indica la operación de extraer profundidad de anidamiento de transacciones; y un campo de registro R_1 804 que designa un registro general.

La profundidad de anidamiento de transacciones actual se pone 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 una realización adicional, la profundidad de anidamiento de transacciones máxima también se pone en el registro general R_1, tal como en los bits 16-31.

Anulación de Filtrado de Interrupción de Programa (PIFO) de ejecución de transacción: El bit 9 del registro de control cero es la anulación de filtrado de interrupción de programa de ejecución transaccional. Este bit proporciona un mecanismo por el cual el programa de control puede asegurar que cualquier condición de excepción de programa que ocurra mientras que la CPU está en el modo de ejecución transaccional da como resultado una interrupción, independientemente del control de filtrado de interrupción de programa eficaz especificado o implícito por la instrucción o las instrucciones COMIENZO DE TRANSACCIÓN.

Bits del registro de control 2: Las asignaciones de bits son de la siguiente manera, en una realización:

Alcance de Diagnóstico de Transacción (TDS): El bit 61 del registro de control 2 controla la aplicabilidad del control de diagnóstico de transacción (TDC) 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 de 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 Transacciones (TDC): Los bits 62-63 del registro de control 2 son un número entero sin signo de 2 bits que se pueden usar para hacer que las transacciones se aborten aleatoriamente con propósitos de diagnóstico. La codificación del TDC es la siguiente, en un ejemplo:

TDS

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 FIN DE TRANSACCIÓN más externa.
2	Abortar transacciones aleatorias en una instrucción aleatoria.
3	Reservado.

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

- El código de abortar se establece en cualquiera de los códigos 7-11, 13-16 o 255, con el valor del código elegido aleatoriamente por la CPU; el código de condición se establece correspondiente al código de abortar. Los códigos de abortar se describen además a continuación.
- Para una transacción no restringida, el código de condición se establece en uno. En este caso, el código de abortar no es aplicable.

Es dependiente del modelo si se implementa el valor de TDC de 1. Si no se implementa, un valor de 1 actúa como si se hubiese especificado 2.

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

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

Dirección de Bloque de Diagnóstico de Transacciones (TDBA)

Se establece una dirección de bloque de diagnóstico de transacciones (TDBA) válida a partir de la dirección del primer operando de la instrucción COMIENZO DE TRANSACCIÓN (TBEGIN) más externa cuando el campo B_1 de la instrucción es distinto de cero. Cuando la CPU está en el espacio primario o en el modo de registro de acceso, la TDBA designa una ubicación en el espacio de direcciones primario. Cuando la CPU está en el espacio secundario, o en el modo de espacio de inicio, la TDBA designa una ubicación en el espacio de dirección de inicio o secundario, respectivamente. Cuando DAT (Traducción Dinámica de Direcciones) está apagada, la TDBA designa una ubicación en un almacenamiento real.

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

Cuando el campo B_1 de una instrucción COMIENZO DE TRANSACCIÓN (TBEGIN) más externa es cero, la dirección de bloque de diagnóstico transaccional no es válida, y no se almacena ningún TDB especificado por TBEGIN si se aborta posteriormente la transacción.

PSW de Abortar Transacción (TAPSW)

Durante la ejecución de la instrucción COMIENZO DE TRANSACCIÓN (TBEGIN) cuando la profundidad de anidamiento es inicialmente cero, la PSW de abortar transacción se establece en los contenidos de la PSW actual; y la dirección de instrucciones de PSW de abortar transacción designa la siguiente instrucción secuencial (es decir, la instrucción que sigue a la TBEGIN más externa). Durante la ejecución de la instrucción COMIENZO DE TRANSACCIÓN restringida (TBEGINC), cuando la profundidad de anidamiento es inicialmente cero, la PSW de abortar transacción se establece en los contenidos de la PSW actual, excepto que la dirección de instrucción de la PSW de abortar transacción designe la instrucción TBEGINC (en lugar de la siguiente instrucción secuencial que sigue a la TBEGINC).

Cuando se aborta una transacción, el código de condición en la PSW de abortar transacción se reemplaza con un código que indica la gravedad de la condición abortada. Posteriormente, si la transacción se abortó debido a causas que no dan como resultado una interrupción, la PSW se carga a partir de la PSW de abortar transacción; si la transacción se abortó debido a causas que dan como resultado una interrupción, la PSW de abortar transacción se almacena como la PSW antigua de interrupción.

La PSW de abortar transacción no se altera durante la ejecución de ninguna instrucción COMIENZO DE TRANSACCIÓN interna.

Profundidad de Anidamiento de Transacciones (TND)

La profundidad de anidamiento de transacciones es, por ejemplo, un valor sin signo de 16 bits que se aumenta cada vez que se completa una instrucción COMIENZO DE TRANSACCIÓN con el código de condición 0 y se disminuye cada vez que se completa una instrucción FIN DE TRANSACCIÓN. La profundidad de anidamiento de transacciones se reinicia a cero cuando se aborta una transacción o por un reinicio de la CPU.

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

En una implementación, cuando la CPU está en el modo de ejecución transaccional restringida, la profundidad de anidamiento de transacciones es uno. Además, aunque la TND máxima se puede representar como un valor de 4 bits, la TND se define que es un valor de 16 bits para facilitar su inspección en el bloque de diagnóstico de transacciones.

Bloque Diagnóstico de Transacciones (TDB)

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

1. TDB especificado por TBEGIN: Para una transacción no restringida, cuando el campo B_1 de la instrucción TBEGIN más externa es distinto de cero, la dirección del primer operando de la instrucción designa el TDB especificado por TBEGIN. Éste es una ubicación específica de programa de aplicación que se puede examinar por el controlador de abortar de la aplicación.

2. TDB de Interrupción de Programa (PI): Si una transacción no restringida se aborta debido a una condición de excepción de programa no filtrada, o si se aborta una transacción restringida debido a cualquier condición de excepción de programa (es decir, cualquier condición que da como resultado que una interrupción de programa sea reconocida), el TDB de PI se almacena en ubicaciones en el área de prefijo. Éste está disponible para que el sistema operativo inspeccione y cierre sesión en cualquier informe de diagnóstico que pueda proporcionar.

3. TDB de interceptación: Si la transacción se aborta debido a cualquier condición de excepción de programa que da como resultado una interceptación (es decir, la condición hace que la ejecución interpretativa finalice y el control vuelva al programa del ordenador central), un TDB se almacena en una ubicación especificada en el bloque de descripción de estado para el sistema operativo invitado.

El TDB especificado por TBEGIN solamente se almacena, en una realización, cuando la dirección de TDB es válida (es decir, cuando el campo B_1 de la instrucción TBEGIN más externa es distinto de cero).

Para abortados debidos a las condiciones de excepción de programa no filtrada, solamente se almacenará uno de o bien el TDB de PI o bien el TDB de interceptación. De este modo, puede haber cero, uno o dos TDB almacenados para un abortar.

Se describirán a continuación detalles adicionales con respecto a un ejemplo de cada uno de los TDB:

TDB especificado por TBEGIN: La ubicación de 256 bytes especificada por una dirección de bloque de diagnóstico de transacciones válida. Cuando la dirección de bloque de diagnóstico de transacciones es válida, el TDB especificado por TBEGIN se almacena en un abortar transacción. El TDB especificado por TBEGIN está sometido a todos los mecanismos de protección de almacenamiento que están vigentes en la ejecución de la instrucción COMIENZO DE TRANSACCIÓN más externa. Se detecta un evento de alteración de almacenamiento de PER (Grabación de Eventos de Programa) para cualquier parte del TDB especificado por TBEGIN durante la ejecución de la TBEGIN más externa, no durante el procesamiento de abortar transacción.

Un propósito de PER es ayudar en la depuración de programas. Permite que el programa sea alertado de los siguientes tipos de eventos, como ejemplos:

- Ejecución de una instrucción de ramificación con éxito. Se proporciona la opción de que ocurra un evento solamente cuando la ubicación de objetivo de ramificación está dentro del área de almacenamiento designada.
- Búsqueda de una instrucción del área de almacenamiento designada.
- Alteración de los contenidos del área de almacenamiento designada. Se proporciona la opción de tener un evento que ocurre solamente cuando el área de almacenamiento está dentro de los espacios de direcciones designados.
- Ejecución de una instrucción ALMACENAR USANDO DIRECCIÓN REAL.
- Ejecución de la instrucción FIN DE TRANSACCIÓN.

El programa puede especificar selectivamente que uno o más de los tipos de eventos anteriores se reconozcan, excepto que el evento para ALMACENAR USANDO DIRECCIÓN REAL se pueda especificar solamente junto con el

evento de alteración de almacenamiento. La información que concierne a un evento de PER se proporciona al programa por medio de una interrupción de programa, con la causa de la interrupción que se identifica en el código de interrupción.

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

TDB de interrupción de programa: Las ubicaciones reales 6144-6399 (1800-18FF hex). El TDB de interrupción de programa se almacena cuando se aborta una transacción debido a una interrupción de programa. Cuando se aborta una transacción debido a otras causas, los contenidos del TDB de interrupción de programa son impredecibles.

El TDB de interrupción de programa no está sometido a ningún mecanismo de protección. Los eventos de alteración de almacenamiento de 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 ordenador central de 256 bytes especificada por las ubicaciones 488-495 de la descripción de estado. El TDB de interceptación se almacena cuando una transacción abortada da como resultado una interceptación de interrupción de programa invitado (es decir, el código de interceptación 8). Cuando se aborta una transacción debido a otras causas, los contenidos del TDB de interceptación son impredecibles. El TDB de interceptación no está sometido a ningún mecanismo de protección.

Como se representa en la FIG. 9, los campos de un bloque de diagnóstico de transacciones 900 son de la siguiente manera, en una realización:

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

Valor	Significado
0	Los campos restantes del TDB son impredecibles.
1	Un TDB de formato 1, los campos restantes del cual se describen a continuación.
2-255	Reservados

Un TDB en el que el campo de formato es cero se conoce como un TDB nulo.

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

Validez de Testigo de Conflicto (CTV): Cuando se aborta una transacción debido a un conflicto de búsqueda o almacén (es decir, los códigos de abortar 9 o 10, respectivamente), el bit 0 del byte 1 es la indicación de validez de testigo de conflicto. Cuando la indicación de CTV es uno, el testigo de conflicto 910 en los bytes 16-23 del TDB contiene la dirección lógica en la que se detectó el conflicto. Cuando la indicación de CTV es cero, los bytes 16-23 del TDB son impredecibles.

Cuando se aborta una transacción debido a cualquier otra razón distinta de un conflicto de búsqueda o almacén, el bit 0 del byte 1 se almacena como cero.

Indicación de Transacción Restringida (CTI): Cuando la CPU está en el modo de ejecución transaccional restringida, el bit 1 del byte 1 se establece en uno. Cuando la CPU está en el modo de ejecución transaccional no restringida, 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 Anidamiento de Transacciones (TND) 906: Los bytes 6-7 contienen la profundidad de anidamiento de transacciones cuando se abortó la transacción.

Código de Abortar Transacción (TAC) 908: Los bytes 8-15 contienen un código de abortar transacción sin signo de 64 bits. Cada punto de código indica una razón para que una transacción se aborte.

Es dependiente del modelo si el código de abortar transacción se almacena en el TDB de interrupción de programa cuando se aborta una transacción debido a condiciones distintas de una interrupción de programa.

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

Cuando el bit de 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 accedería necesariamente por la secuencia de ejecución conceptual de la transacción.

Dirección de Instrucción de Transacción Abortada (ATIA) 912: Los bytes 24-31 contienen una dirección de instrucción que identifica la instrucción que estaba ejecutándose cuando se detectó un abortar. Cuando una transacción se aborta debido a los códigos de abortar 2, 5, 6, 11, 13 o 256 o superior, o cuando una transacción se aborta debido a los códigos de abortar 4 o 13 y la condición de excepción de programa es nula, la ATIA apunta directamente a la Instrucción que estaba siendo ejecutada. Cuando una transacción se aborta debido a códigos de abortar 4 o 12, y la condición de excepción de programa no es nula, la ATIA apunta pasada la instrucción que estaba siendo ejecutada.

Cuando se aborta una transacción, debido a los códigos de abortar 7-10, 14-16 o 255, la ATIA no necesariamente indica la instrucción exacta que causa la cancelación, sino que puede apuntar a una instrucción anterior o posterior dentro de la transacción.

Si se aborta una transacción debido a una instrucción que es el objetivo de una instrucción de tipo ejecutar, la ATIA identifica la instrucción de tipo ejecutar, o bien que apunta a la instrucción o bien la pasa, dependiendo del código de abortar como se ha descrito anteriormente. La ATIA no indica el objetivo de la instrucción de tipo ejecutar.

La ATIA se somete al modo de direccionamiento cuando se aborta la transacción. 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.

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

Cuando se aborta una transacción debido a un código de abortar 4 o 12, y la condición de excepción de programa no está anulando, la ATIA no apunta a la instrucción que causa el abortar. Sustrayendo el número de medias palabras indicado por el código de longitud de interrupción (ILC) de la ATIA, la instrucción que causa el abortar se puede identificar en condiciones que están suprimiendo o terminando, o para eventos que no son PER que se están completando. Cuando una transacción se aborta debido a un evento de PER, y no está presente ninguna otra condición de excepción de programa, la ATIA es impredecible.

Cuando la dirección de bloque de diagnóstico de transacciones es válida, el ILC se puede examinar en la identificación de interrupción de programa (PIID) en los bytes 36-39 del TDB especificado por TBEGIN. Cuando no se aplica filtrado, el ILC se puede examinar en la PIID en la ubicación 140-143 en un almacenamiento real.

Identificación de Acceso de Excepción (EAID) 914: Para las transacciones que se abortan debido a ciertas condiciones de excepción de programa filtradas, el byte 32 del TDB especificado por TBEGIN contiene la identificación de acceso de excepción. En un ejemplo de la z/Architecture, el formato de la EAID, y los casos para los que se almacena, son los mismos que los descritos en la ubicación real 160 cuando la condición de excepción da como resultado una interrupción, como se describe en los Principios de Operación incorporados anteriormente por referencia.

Para las transacciones que se abortan por otras razones, incluyendo cualquier condición de excepción que da como resultado una interrupción de programa, el byte 32 es impredecible. El byte 32 es impredecible en el TDB de interrupción de programa.

Este campo se almacena solamente en el TDB designado por la dirección de bloque de diagnóstico de transacciones; de otro modo, el campo está reservado. La EAID se almacena solamente para la lista de acceso controlada o protección de DAT, tipo de ASCE, traducción de páginas, región de primera traducción, región de segunda traducción, región de tercera traducción y condiciones de excepción de programa de traducción de segmento.

Código de Excepción de Datos (DXC) 916: Para las transacciones que se abortan debido a las condiciones de excepción de programa de excepción de datos filtrada, el byte 33 del TDB especificado por TBEGIN contiene el código de excepción de datos. En un ejemplo de la z/Architecture, el formato del DXC, y los casos para los que se almacena, son los mismos que los descritos en la ubicación real 147 cuando la condición de excepción da como resultado una interrupción, como se describe en los Principios de Operación incorporados anteriormente por referencia. En un ejemplo, la ubicación 147 incluye el DXC.

Para las transacciones que se abortan por otras razones, incluyendo cualquier condición de excepción que da como resultado una interrupción de programa, el byte 33 es impredecible. El byte 33 es impredecible en el TDB de interrupción de programa.

Este campo se almacena solamente en el TDB designado por la dirección de bloque de diagnóstico de transacciones; de otro modo, el campo está reservado. El DXC se almacena solamente para las condiciones de excepción de programa de datos.

Identificación de Interrupción de Programa (PIID) 918: Para las transacciones que se abortan debido a condiciones de excepción de programa filtrada, los bytes 36-39 del TDB especificado por TBEGIN contienen la identificación de interrupción de programa. En un ejemplo de la z/Architecture, el formato de la PIID es el mismo que el descrito en las ubicaciones reales 140-143 cuando la condición da como resultado una interrupción (como se describe en los Principios de Operación incorporados anteriormente por referencia), excepto que el código de longitud de instrucción en los bits 13-14 de la PIID sea respectivo a la instrucción en la que se detectó la condición de excepción.

Para las transacciones que se abortan por otras razones, incluyendo las condiciones de excepción que dan como resultado una interrupción de programa, los bytes 36-39 son impredecibles. Los bytes 36-39 son impredecibles en el TDB de interrupción de programa.

Este campo se almacena solamente en el TDB designado por la dirección de bloque de diagnóstico de transacciones; de otro modo, el campo está reservado. La identificación de interrupción de programa se almacena solamente para las condiciones de excepción de programa.

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

- Lista de acceso controlada o protección de DAT
- Tipo de ASCE
- Traducción de páginas
- Región de primera traducción
- Región de segunda traducción
- Región de tercera traducción
- Excepción de traducción de segmento

En un ejemplo de la z/Architecture, el formato de la TEID es el mismo que el descrito en las ubicaciones reales 168-175 cuando la condición da como resultado una interrupción, como se describe en los Principios de Operación incorporados anteriormente por referencia.

Para las transacciones que se abortan por otras razones, incluyendo las condiciones de excepción que dan como resultado una interrupción de programa, los bytes 40-47 son impredecibles. Los bytes 40-47 son impredecibles en el TDB de interrupción de programa.

Este campo se almacena solamente en el TDB designado por la dirección de bloque de diagnóstico de transacciones; de otro modo, el campo está reservado.

Dirección de Evento de Ruptura 922: Para las transacciones que se abortan debido a las condiciones de excepción de programa filtrada, los bytes 48-55 del TDB especificado por TBEGIN contienen la dirección de evento de ruptura. En un ejemplo de la z/Architecture, el formato de la dirección de evento de ruptura es el mismo que el descrito en las ubicaciones reales 272-279 cuando la condición da como resultado una interrupción, como se describe en los Principios de Operación incorporados anteriormente por referencia.

Para las transacciones que se abortan por otras razones, incluyendo las condiciones de excepción que dan como resultado una interrupción de programa, los bytes 48-55 son impredecibles. Los bytes 48-55 son impredecibles en el TDB de interrupción de programa.

Este campo se almacena solamente en el TDB designado por la dirección de bloque de diagnóstico de transacciones; de otro modo, el campo está reservado.

Se describen a continuación detalles adicionales relacionados con los eventos de ruptura.

En una realización de la z/Architecture, cuando la facilidad de PER-3 está instalada, dota al programa con la dirección de la última instrucción para causar una ruptura en la ejecución secuencial de la CPU. La grabación de la dirección de evento de ruptura se puede usar como una ayuda de depuración para detección de ramificación comodín. Esta facilidad proporciona, por ejemplo, un registro de 64 bits en la CPU, denominado registro de direcciones de eventos de ruptura. Cada vez que una instrucción distinta de ABORTAR TRANSACCIÓN causa una ruptura en la ejecución de instrucción secuencial (es decir, se reemplaza la dirección de instrucción en la PSW, en

lugar de aumentarse en la longitud de la instrucción), la dirección de esa instrucción se pone en el registro de direcciones de eventos de ruptura. Siempre que ocurre una interrupción de programa, ya sea que se indique PER o no, los contenidos actuales del registro de dirección de evento de ruptura se ponen en las ubicaciones de almacenamiento real 272-279.

Si la instrucción que causa el evento de ruptura es el objetivo de una instrucción de tipo ejecutar (EJECUTAR o EJECUTAR RELATIVA LARGA), entonces la dirección de instrucción usada para obtener la instrucción de tipo ejecutar se pone en el registro de direcciones de eventos de ruptura.

En una realización de la z/Architecture, se considera que un evento de ruptura ocurre siempre que una de las siguientes instrucciones causa ramificación: RAMIFICAR Y ENLAZAR (BAL, BALR); RAMIFICAR Y GUARDAR (BAS, BASR); RAMIFICAR Y GUARDAR Y ESTABLECER MODO (BASSM); RAMIFICAR Y ESTABLECER MODO (BSM); RAMIFICAR Y APILAR (BAKR); RAMIFICAR BAJO CONDICIÓN (BC, BCR); RAMIFICAR BAJO RECUENTO (BCT, BCTR, BCTG, BCTGR); RAMIFICAR BAJO ÍNDICE ALTA (BXH, BXHG); RAMIFICAR BAJO ÍNDICE BAJA O IGUAL (BXLE, BXLEG); RAMIFICAR RELATIVA BAJO CONDICIÓN (BRC); RAMIFICAR RELATIVA BAJO CONDICIÓN LARGA (BRCL); RAMIFICAR RELATIVA BAJO RECUENTO (BRCT, BRCTG); RAMIFICAR RELATIVA BAJO ÍNDICE ALTA (BRXH, BRXHG); RAMIFICAR RELATIVA BAJO ÍNDICE BAJA O IGUAL (BRXLE, BRXLG); COMPARAR Y RAMIFICAR (CRB, CGRB); COMPARAR Y RAMIFICAR RELATIVA (CRJ, CGRJ); COMPARAR INMEDIATA Y RAMIFICAR (CIB, CGIB); COMPARAR INMEDIATA Y RAMIFICAR RELATIVA (CIJ, CGIJ); COMPARAR LÓGICA Y RAMIFICAR (CLRB, CLGRB); COMPARAR LÓGICA Y RAMIFICAR RELATIVA (CLRJ, CLGRJ); COMPARAR LÓGICA INMEDIATA Y RAMIFICAR (CLIB, CLGIB); y COMPARAR LÓGICA INMEDIATA Y RAMIFICAR RELATIVA (CLIJ, CLGIJ).

También se considera que un evento de ruptura ocurre siempre que se completa una de las siguientes instrucciones: RAMIFICAR Y ESTABLECER AUTORIDAD (BSA); RAMIFICAR EN GRUPO DE SUBSPACIO (BSG); RAMIFICAR RELATIVA Y GUARDAR (BRAS); RAMIFICAR RELATIVA Y GUARDAR LARGA (BRASL); CARGAR PSW (LPSW); CARGAR PSW EXTENDIDA (LPSWE); LLAMADA DE PROGRAMA (PC); VUELTA DE PROGRAMA (PR); TRANSFERENCIA DE PROGRAMA (PT); TRANSFERENCIA DE PROGRAMA CON INSTANCIA (PTI); REANUDAR PROGRAMA (RP); y TRAMPA (TRAMPA2, TRAMPA4).

No se considera que un evento de ruptura ocurra como resultado de una transacción que se aborta (o bien implícitamente o bien como resultado de la instrucción ABORTAR TRANSACTION).

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

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

En una realización, la información de diagnóstico dependiente del modelo incluye lo siguiente:

- Los bytes 112-119 contienen un vector de 64 bits denominados indicaciones de ramificación de ejecución transaccional (TXBI). Cada uno de los primeros 63 bits del vector indica los resultados de ejecutar una instrucción de ramificación mientras la CPU estaba en el modo de ejecución transaccional, de la siguiente manera:

Valor	Significado
0	La instrucción completada sin ramificación.
1	La instrucción completada con ramificación.

El bit 0 representa el resultado de primera instrucción de ramificación tal, el bit 1 representa el resultado de la segunda instrucción tal, y así sucesivamente.

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

Los bits en la TXBI se establecen mediante instrucciones que son capaces de causar un evento de ruptura, como se ha enumerado anteriormente, excepto por lo siguiente:

- Cualquier instrucción restringida no hace que un bit se establezca en la TXBI.
- Para instrucciones, por ejemplo, de la z/Architecture, cuando el campo M_1 de la instrucción RAMIFICAR BAJO CONDICIÓN, RAMIFICAR RELATIVA BAJO CONDICIÓN, o RAMIFICAR RELATIVA BAJO CONDICIÓN LARGA es cero, o cuando el campo R_2 de las siguientes instrucciones es cero, es dependiente del modelo si la ejecución de la instrucción hace que un bit se establezca en la TXBI.

- RAMIFICAR Y ENLAZAR (BALR); RAMIFICAR Y GUARDAR (BASR); RAMIFICAR Y GUARDAR Y ESTABLECER MODO (BASSM); RAMIFICAR Y ESTABLECER MODO (BSM); RAMIFICAR BAJO CONDICIÓN (BCR); y RAMIFICAR BAJO RECUENTO (BCTR, BCTGR)

- Para las condiciones de abortar que se causaron por una excepción de acceso de ordenador central, la posición del bit 0 del byte 127 se establece en uno. Para todas las otras condiciones de abortar, la posición del bit 0 del byte 127 se establece en cero.

- Para las condiciones de abortar que se detectaron por la unidad de carga/almacenamiento (LSU), los cinco bits de byte 127 de más a la derecha contienen una indicación de la causa. Para las condiciones de abortar que no se detectaron por la LSU, el byte 127 está reservado.

Registros Generales 930: Los bytes 128-255 contienen los contenidos de los registros generales 0-15 en el momento que se abortó la transacción. 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. A menos que se indique de otro modo, los contenidos de los campos reservados son impredecibles.

Como se observa por otras CPU y el subsistema de I/O, el almacenamiento del o de los TDB durante un abortar transacción es una referencia de acceso múltiple que ocurre después de cualquier almacén no transaccional.

Una transacción se puede abortar debido a causas que están fuera del alcance de la configuración inmediata en la que se ejecuta. Por ejemplo, los eventos transitorios reconocidos por un hipervisor (tales como LPAR o z/VM) pueden hacer que una transacción sea abortada.

La información proporcionada en el bloque de diagnóstico de transacciones está destinada con propósitos de diagnóstico y es sustancialmente correcta. No obstante, debido a que un abortar se puede haber causado por un evento fuera del alcance de la configuración inmediata, información tal como el código de abortar o la identificación de interrupción de programa no puede reflejar con precisión las condiciones dentro de la configuración y, de este modo, no se debería usar en la determinación de la acción del programa.

Además de la información de diagnóstico guardada en el TDB, cuando se aborta una transacción debido a cualquier condición de excepción de programa de excepción de datos y tanto el control de registro de AFP, el bit 45 del registro de control 0, como el control de operación de punto flotante (F) permitida eficaz son uno, el código de excepción de datos (DXC) se pone 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 se aborta una transacción, y cualquiera o ambos del control de registro de AFP o del control de operación de punto flotante permitida eficaz son cero, el DXC no se pone en el FPCR.

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

- EXTRAER PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIONES
- ALMACÉN NO TRANSACCIONAL
- ABORTAR TRANSACCION
- COMIENZO DE TRANSACCIÓN
- FIN DE TRANSACCION

Cuando la CPU está en el modo de ejecución transaccional, el intento de ejecución de ciertas instrucciones se restringe y hace que la transacción sea abortada.

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

En un ejemplo de la z/Architecture, las instrucciones restringidas incluyen, como ejemplos, las siguientes instrucciones no privilegiadas: COMPARAR E INTERCAMBIAR Y ALMACENAR; MODIFICAR CONTROLES DE INSTRUMENTACIÓN DE TIEMPO DE EJECUCIÓN; REALIZAR OPERACIÓN BLOQUEADA; OBTENER PREVIAMENTE DATOS (RELATIVA LARGA), cuando el código en el campo M_1 es 6 o 7; ALMACENAR CARACTERES BAJO MÁSCARA ALTA, cuando el campo M_3 es cero y el código en el campo R_1 es 6 o 7; ALMACENAR LISTA DE FACILIDADES EXTENDIDA; ALMACENAR CONTROLES DE INSTRUMENTACIÓN DE TIEMPO DE EJECUCIÓN; LLAMADA DE SUPERVISOR; y PROBAR CONTROLES DE INSTRUMENTACIÓN DE TIEMPO DE EJECUCIÓN.

En la lista anterior, COMPARAR E INTERCAMBIAR Y ALMACENAR y REALIZAR OPERACIÓN BLOQUEADA son instrucciones complejas que se pueden implementar de manera más eficaz mediante el uso de instrucciones básicas en el modo de TX. Los casos de OBTENER PREVIAMENTE DATOS y OBTENER PREVIAMENTE DATOS RELATIVA LARGA están restringidos en la medida que los códigos 6 y 7 liberan una línea de caché, necesitando el compromiso de los datos potencialmente antes de la terminación de una transacción. LLAMADA DE SUPERVISOR está restringida en la medida que causa una interrupción (que hace que una transacción sea abortada).

Bajo las condiciones enumeradas a continuación, las siguientes instrucciones están restringidas:

- RAMIFICAR Y ENLAZAR (BALR), RAMIFICAR Y GUARDAR (BASR), y RAMIFICAR Y GUARDAR Y ESTABLECER MODO, cuando el campo R_2 de la instrucción es distinto de cero y el seguimiento de ramificación está habilitado.
- RAMIFICAR Y GUARDAR Y ESTABLECER MODO y RAMIFICAR Y ESTABLECER MODO, cuando el campo R_2 es distinto de cero y el seguimiento de modo está habilitado; ESTABLECER MODO DE DIRECCIONAMIENTO, cuando el seguimiento de modo está habilitado.
- LLAMADA DE MONITOR, cuando se reconoce una condición de evento de monitor.

La lista anterior incluye instrucciones que pueden formar entradas de seguimiento. Si se permitiese que estas instrucciones se ejecuten de manera transaccional y formasen entradas de seguimiento, y se abortase la transacción posteriormente, el puntero de la tabla de seguimiento en el registro de control 12 se avanzaría, pero los almacenes de la tabla de seguimiento se descartarían. Esto dejaría un hueco incoherente en la tabla de seguimiento; de este modo, las instrucciones están restringidas en los casos donde formarían entradas de seguimiento.

Cuando la CPU está en el modo de ejecución transaccional, es dependiente del modelo si las siguientes instrucciones están restringidas: CIFRAR MENSAJE; CIFRAR MENSAJE CON CFB; CIFRAR MENSAJE CON ENCADENAMIENTO; CIFRAR MENSAJE CON CONTADOR; CIFRAR MENSAJE CON OFB; LLAMADA DE COMPRESIÓN; CALCULAR RESUMEN DE MENSAJE INTERMEDIO; CALCULAR RESUMEN DE ÚLTIMO MENSAJE; CALCULAR CÓDIGO DE AUTENTICACIÓN DE MENSAJE; CONVERTIR UNICODE-16 A UNICODE-32; CONVERTIR UNICODE-16 A UNICODE-8; CONVERTIR UNICODE-32 A UNICODE-16; CONVERTIR UNICODE-32 A UNICODE-8; CONVERTIR UNICODE-8 A UNICODE-16; CONVERTIR UNICODE-8 A UNICODE-32; REALIZAR CÁLCULO CRIPTOGRAFICO; APAGADO DE INSTRUMENTACIÓN DE TIEMPO DE EJECUCIÓN; y ENCENDIDO DE INSTRUMENTACIÓN DE TIEMPO DE EJECUCIÓN.

Cada una de las instrucciones anteriores o bien se implementa actualmente por el coprocesador de hardware, o bien ha estado en máquinas anteriores y, de este modo, se considera restringida.

Cuando el control de modificación de AR (A) permitida eficaz es cero, las siguientes instrucciones están restringidas: COPIAR ACCESO; CARGAR ACCESO MULTIPLE; CARGAR DIRECCIÓN EXTENDIDA; y ESTABLECER ACCESO.

Cada una de las instrucciones anteriores hace que los contenidos de un registro de acceso se modifiquen. Si el control A en la instrucción COMIENZO DE TRANSACCIÓN es cero, entonces el programa ha indicado explícitamente que no ha de ser permitida una modificación del registro de acceso.

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

Bajo ciertas circunstancias, las siguientes instrucciones pueden estar restringidas: EXTRAER TIEMPO DE CPU; EXTRAER PSW; ALMACENAR RELOJ; ALMACENAR RELOJ EXTENDIDO; y ALMACENAR RELOJ RÁPIDO.

Cada una de las instrucciones anteriores está sometida a un control de interceptación en la descripción de estado de ejecución interpretativa. Si el hipervisor ha establecido el control de interceptación para estas instrucciones, entonces su ejecución se puede prolongar debido a la implementación del hipervisor; de este modo, se consideran restringidas si ocurre una interceptación.

Cuando se aborta una transacción no restringida debido a un intento de ejecución de una instrucción restringida, el código de abortar transacción en el bloque de diagnóstico de transacciones se establece en 11 (instrucción restringida), y el código de condición se establece en 3, excepto como sigue: cuando se aborta una transacción no restringida debido al intento de ejecución de una instrucción que, de otro modo, daría como resultado una excepción de operación privilegiada, es impredecible si el código de abortar se establece en 11 (instrucción restringida) o 4 (interrupción de programa no filtrada resultante del reconocimiento de la interrupción de programa de operación privilegiada). Cuando una transacción no restringida se aborta debido al intento de ejecución de OBTENER PREVIAMENTE DATOS (RELATIVA LARGA) cuando el código en el campo M_1 es 6 o 7 o ALMACENAR CARACTERES BAJO MÁSCARA ALTA 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 abortar se establece en 11 (instrucción restringida) o 16 (otra memoria caché). Cuando una transacción no restringida se aborta debido al intento de ejecución de LLAMADA DE MONITOR, y tanto una

condición de evento de monitor como una condición de excepción de especificación están presentes, es impredecible si el código de abortar se establece en 11 o 4, o, si se filtra la interrupción del programa, 12.

Instrucciones adicionales se pueden restringir en una transacción restringida. Aunque no se define actualmente que estas instrucciones estén restringidas en una transacción no restringida, se pueden restringir bajo ciertas circunstancias en una transacción no restringida en procesadores futuros.

Se pueden permitir ciertas instrucciones restringidas en el modo de ejecución transaccional en procesadores futuros. Por lo tanto, el programa no debería depender de que la transacción sea abortada debido al intento de ejecución de una instrucción restringida. La instrucción ABOTAR TRANSACCIÓN se debería usar para hacer de manera fiable que una transacción sea abortada.

En una transacción no restringida, el programa debería proporcionar un camino de código no transaccional alternativo para acomodar una transacción que se aborta debido a una instrucción restringida.

En operación, cuando la profundidad de anidamiento de transacciones es cero, la ejecución de la instrucción COMIENZO DE TRANSACCIÓN (TBEGIN) que da como resultado un código de condición cero hace que la CPU entre en el modo de ejecución transaccional no restringida. Cuando la profundidad de anidamiento de transacciones es cero, la ejecución de la instrucción COMIENZO DE TRANSACCIÓN restringida (TBEGINC) que da como resultado un código de condición cero hace que la CPU entre en el modo de ejecución transaccional restringida.

Excepto donde se señale explícitamente de otro modo, todas las reglas que se aplican para la ejecución no transaccional también se aplican a la ejecución transaccional. A continuación están características adicionales de procesamiento mientras la CPU está en el modo de ejecución transaccional.

Cuando la CPU está en el modo de ejecución transaccional no restringida, la ejecución de la instrucción COMIENZO DE TRANSACCIÓN que da como resultado un código de condición cero hace que la CPU permanezca en el modo de ejecución transaccional no restringida.

Como se observa por la CPU, las obtenciones y los almacenes hechos en el modo de ejecución de transacción no son diferentes de los hechos mientras que no está en el modo de ejecución transaccional. Como se observa por otras CPU y por el subsistema de I/O, todos los accesos de operandos de almacenamiento hechos mientras que una CPU está en el modo de ejecución transaccional parecen ser un único acceso concurrente de bloque. Es decir, los accesos a todos los bytes dentro de media palabra, palabra, palabra doble o palabra cuádruple se especifican para parecer que son concurrentes de bloque como se observa por otras CPU y programas de I/O (por ejemplo, canal). La media palabra, palabra, palabra doble o palabra cuádruple se conocen en esta sección como bloque. Cuando se especifica que una referencia de tipo de búsqueda parezca que es concurrente dentro de un bloque, no se permite ningún acceso de almacén al bloque por otra CPU o programa de I/O durante el tiempo que los bytes contenidos en el bloque están siendo obtenidos. Cuando se especifica que una referencia de tipo almacén parece que es concurrente dentro de un bloque, no se permite ningún acceso al bloque, o bien de obtención o bien de almacén, por otra CPU o programa de I/O durante el tiempo que los bytes dentro del bloque están siendo almacenados.

Los accesos de almacenamiento para las búsquedas de tablas de instrucciones y DAT y ART (Tabla de Registros de Acceso) siguen las reglas no transaccionales.

La CPU abandona el modo de ejecución transaccional normalmente por medio de una instrucción FIN DE TRANSACCIÓN que hace que la profundidad de anidamiento de transacciones pase a cero, en cuyo caso, la transacción se completa.

Cuando la CPU abandona el modo de ejecución transaccional por medio de la terminación de una instrucción FIN DE TRANSACCIÓN, se comprometen todos los almacenes hechos mientras que está en el modo de ejecución transaccional; es decir, los almacenes parecen ocurrir como una única operación concurrente de bloque como se observa por otras CPU y por el subsistema de I/O.

Una transacción se puede abortar implícitamente por una variedad de causas, o se puede abortar explícitamente mediante la instrucción ABORTAR TRANSACCIÓN. Se describen a continuación causas posibles de ejemplo de un abortar transacción, el código de abortar correspondiente y el código de condición que se coloca en la PSW de abortar transacción.

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

Interrupción de Programa (No Filtrada): Una condición de excepción de programa que da como resultado 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 abortar transacción se establece específicamente para el código de interrupción de programa. La PSW de abortar transacción se almacena como la PSW antigua de programa como parte del procesamiento de interrupción de programa.

Una instrucción que de otro modo daría como resultado una transacción que se aborta debido a una excepción de operación puede producir resultados alternativos: para una transacción no restringida, la transacción puede abortarse en su lugar con el código de abortar 11 (instrucción restringida); para una transacción restringida, se puede reconocer una interrupción de programa de restricción de transacción en lugar de la excepción de operación.

Cuando un evento de PER (Grabación de Eventos de Programa) se reconoce junto 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 abortar de transacción se establece en 5, y el código de condición en la PSW de abortar transacción se establece en 2. La PSW de abortar transacción se almacena como la PSW antigua de comprobación de máquina como parte del procesamiento de interrupción de comprobación de máquina.

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

Desbordamiento de Obtención: Se detecta una condición de desbordamiento de obtención cuando la transacción intenta buscar desde más ubicaciones que las que soporta la CPU. El código de abortar transacción se establece en 7 y el código de condición se establece o bien en 2 o bien en 3.

Desbordamiento de Almacén: Se detecta una condición de desbordamiento de almacén cuando la transacción intenta almacenar en más ubicaciones que las que soporta la CPU. El código de abortar transacción se establece en 8, y el código de condición se establece o bien en 2 o bien en 3.

Permitir que el código de condición sea o bien 2 o bien 3 en respuesta a un abortar desbordamiento de obtención o de almacén permite que la CPU indique situaciones que se pueden volver a intentar potencialmente (por ejemplo, el código de condición 2 indica que puede ser productiva la ejecución de nuevo de la transacción; mientras que el código de condición 3 no recomienda la ejecución de nuevo).

Conflicto de Obtención: Se detecta una condición de conflicto de obtención cuando otra CPU o el subsistema de I/O intenta almacenar en una ubicación que ha sido obtenida de manera transaccional por esta CPU. El código de abortar transacción se establece en 9, y el código de condición se establece en 2.

Conflicto de Almacén: Se detecta una condición de conflicto de almacén cuando otra CPU o el subsistema de I/O intenta acceder a una ubicación que ha sido almacenada durante la ejecución transaccional por esta CPU. El código de abortar 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, el intento de ejecución de una instrucción restringida hace que la transacción sea abortada. El código de abortar transacción se establece en 11 y el código de condición se establece en 3.

Cuando la CPU está en el modo de ejecución transaccional restringida, es impredecible si el intento de ejecución de una instrucción restringida da como resultado una interrupción de programa de restricción de transacción o un abortar debido a una instrucción restringida. La transacción todavía se aborta pero el código de abortar puede indicar cualquier causa.

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

Profundidad de Anidamiento Excedida: La condición de profundidad de anidamiento excedida se detecta cuando la profundidad de anidamiento de transacciones está en el valor máximo permisible para la configuración, y se ejecuta una instrucción COMIENZO DE TRANSACCIÓN. La transacción se aborta con un código de abortar transacción de 13, y el código de condición se establece en 3.

Condición de Obtención de Memoria Caché Relacionada: Una condición relacionada con las ubicaciones de almacenamiento obtenidas por la transacción se detecta por la circuitería de memoria caché de la CPU. La transacción se aborta con un código de abortar transacción de 14, y el código de condición se establece o bien en 2 o bien en 3.

Condición de Almacén de Memoria Caché Relacionada: Una condición relacionada con las ubicaciones de almacenamiento almacenadas por la transacción se detecta por la circuitería de memoria caché de la CPU. La transacción se aborta con un código de abortar transacción de 15, y el código de condición se establece o bien en 2 o bien en 3.

Condición de Otra Memoria Caché: Una condición de otra memoria caché se detecta por la circuitería de memoria caché de la CPU. La transacción se aborta con un código de abortar transacción de 16, y el código de condición se establece o bien en 2 o bien en 3.

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

Condición Miscelánea: Una condición miscelánea es cualquier otra condición reconocida por la CPU que hace que la transacción se aborte. El código de abortar transacción se establece en 255 y el código de condición se establece o bien en 2 o bien en 3.

Cuando múltiples configuraciones están ejecutándose en la misma máquina (por ejemplo, particiones lógicas o máquinas virtuales), se puede abortar una transacción debido a una comprobación de máquina externa o una interrupción de I/O que ocurrió en una configuración diferente.

Aunque se proporcionan ejemplos anteriormente, se pueden proporcionar otras causas de un abortar transacción con códigos de abortar y códigos de condición correspondientes. Por ejemplo, una causa puede ser una Interrupción de Reinicio, en la que el código de abortar transacción se establece en 1, y el código de condición en la PSW de abortar transacción se establece en 2. La PSW de abortar transacción se almacena como la PSW antigua de reinicio como parte del procesamiento de reinicio. Como ejemplo adicional, una causa puede ser una condición de LLAMADA DE SUPERVISOR, en la que el código de abortar se establece en 3, y el código de condición en la PSW de abortar transacción se establece en 3. También son posibles otros o diferentes ejemplos.

Notas:

1. La condición miscelánea puede resultar de cualquiera de los siguientes:

- Instrucciones, tales como, en la z/Architecture, COMPARAR Y REEMPLAZAR ENTRADA DE TABLA DE DAT, COMPARAR E INTERCAMBIAR Y PURGAR, INVALIDAR ENTRADA DE TABLA DE DAT, INVALIDAR ENTRADA DE TABLA DE PÁGINAS, REALIZAR FUNCIÓN DE GESTIÓN DE TRAMAS en las que el control NQ es cero y el control SK es uno, ESTABLECER CLAVE DE ALMACENAMIENTO EXTENDIDA en la que el control NQ es cero, realizada por otra CPU en la configuración; el código de condición se establece en 2.

- Una función de operador, tal como reinicio, reiniciar o detener, o la orden SEÑALAR PROCESADOR se realiza en la CPU.

- Cualquier otra condición no enumerada anteriormente; el código de condición se establece en 2 o 3.

2. La ubicación en la que se detectan conflictos de obtención y almacén puede estar en cualquier lugar dentro de la misma línea de caché.

3. Bajo ciertas condiciones, la CPU puede no ser capaz de distinguir entre condiciones de abortar similares. Por ejemplo, un desbordamiento de obtención o almacén puede ser indistinguible de un conflicto de obtención o almacén respectivo.

4. La ejecución especulativa de múltiples caminos de instrucción por la CPU puede dar como resultado una transacción que se aborta debido a condiciones de conflicto o desbordamiento, incluso si tales condiciones no ocurren en la secuencia conceptual. Mientras que está en el modo de ejecución transaccional restringida, la CPU puede inhibir temporalmente la ejecución especulativa, permitiendo que la transacción intente completarse sin detectar tales conflictos o desbordamientos de manera especulativa.

La ejecución de una instrucción ABORTAR TRANSACCIÓN hace que la transacción se aborte. El código de abortar transacción se establece a partir de la dirección del segundo operando. El código de condición se establece o bien en 2 o bien en 3, dependiendo de si el bit 63 de la dirección del segundo operando es cero o uno, respectivamente.

La FIG. 10 resume los códigos de abortar de ejemplo almacenados en el bloque de diagnóstico de transacciones, y el código de condición (CC) correspondiente. La descripción en la FIG. 10 ilustra una implementación particular. Son posibles otras implementaciones y codificaciones de valores.

En una realización, y como se mencionó anteriormente, la facilidad transaccional proporciona tanto transacciones restringidas como transacciones no restringidas, así como el procesamiento asociado con las mismas. Inicialmente, se tratan las transacciones restringidas y luego las transacciones no restringidas.

Una transacción restringida se ejecuta en modo transaccional sin un camino de retroceso. Es un modo de procesamiento útil para funciones compactas. En ausencia de interrupciones o conflictos repetidos con otras CPU o el subsistema de I/O (es decir, causado por condiciones que no permitirán que la transacción se complete con éxito), una transacción restringida se completará finalmente; de este modo, no se requiere ni se especifica una rutina de controlador de abortar. Por ejemplo, en ausencia de violación de una condición que no se puede abordar (por ejemplo, dividir por 0); una condición que no permite que la transacción se complete (por ejemplo, una interrupción de temporizador que no permite que se ejecute una instrucción; una I/O en caliente, etc.); o una violación de una restricción o limitación asociada con una transacción restringida, la transacción se completará finalmente.

Una transacción restringida se inicia por una instrucción de COMIENZO DE TRANSACCIÓN restringida (TBEGINC) cuando la profundidad de anidamiento de transacciones es inicialmente cero. Una transacción restringida se somete a las siguientes restricciones, en una realización.

1. La transacción no ejecuta más de 32 instrucciones, sin incluir las instrucciones COMIENZO DE TRANSACCIÓN restringida (TBEGINC) y FIN DE TRANSACCIÓN.

2. Todas las instrucciones en la transacción han de estar dentro de 256 bytes de almacenamiento contiguos, incluyendo las instrucciones COMIENZO DE TRANSACCIÓN restringida (TBEGINC) y cualquier FIN DE TRANSACCIÓN.

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

a. Las instrucciones se limitan a las conocidas como Instrucciones Generales, incluyendo, por ejemplo, sumar, restar, multiplicar, dividir, desplazar, rotar, etc.

b. Las instrucciones de ramificación se limitan a lo siguiente (las instrucciones enumeradas son de la z/Architecture en un ejemplo):

- RAMIFICAR RELATIVA BAJO CONDICIÓN en la que M_1 es distinto de cero y el campo RI_2 contiene un valor positivo.
- RAMIFICAR RELATIVA BAJO CONDICIÓN LARGA en la que el campo M_1 es distinto de cero, y el campo RI_2 contiene un valor positivo que no causa envolver alrededor de las direcciones.
- COMPARAR Y RAMIFICAR RELATIVA, COMPARAR INMEDIATA Y RAMIFICAR RELATIVA, COMPARAR LÓGICA Y RAMIFICAR RELATIVA, y COMPARAR LÓGICA INMEDIATA Y RAMIFICAR RELATIVA en las cuales el campo M_3 es distinto de cero y el campo RI_4 contiene un valor positivo. (Es decir, solamente las ramas delanteras con máscaras de ramificación distintas de cero).

c. Excepto para FIN DE TRANSACCIÓN e instrucciones que causan una serialización de operandos especificada, las instrucciones que causan una función de serialización están restringidas.

d. Operaciones de almacenamiento y almacenamiento (SS-) y operaciones de almacenamiento y almacenamiento con un código de operación extendido (SSE-) están restringidas.

e. Todas las instrucciones generales siguientes (que son de la z/Architecture en este ejemplo) están restringidas: SUMA DE COMPROBACIÓN; CIFRAR MENSAJE; CIFRAR MENSAJE CON CFB; CIFRAR MENSAJE CON ENCADENAMIENTO; CIFRAR MENSAJE CON CONTADOR; CIFRAR MENSAJE CON OFB; COMPARAR Y FORMAR PALABRA DE CODIGO; COMPARAR LÓGICA LARGA; COMPARAR LÓGICA LARGA EXTENDIDA; COMPARAR LOGICA LARGA UNICODE; COMPARAR CADENA LÓGICA; COMPARAR HASTA SUBCADENA IGUAL; LLAMADA DE COMPRESIÓN; CALCULAR RESUMEN DE MENSAJE INTERMEDIA; CALCULAR ÚLTIMO RESUMEN DE MENSAJE; CALCULAR CÓDIGO DE AUTENTICACIÓN DE MENSAJE; CONVERTIR A BINARIO; CONVERTIR A DECIMAL; CONVERTIR UNICODE-16 A UNICODE-32; CONVERTIR UNICODE-16 A UNICODE-8; CONVERTIR UNICODE-32 A UNICODE-16; CONVERTIR UNICODE-32 A UNICODE-8; CONVERTIR UNICODE-8 A UNICODE-16; CONVERTIR UNICODE-8 A UNICODE-32; DIVIDIR; DIVIDIR LÓGICA; DIVIDIR ÚNICA; EJECUTAR; EJECUTAR RELATIVA LARGA; EXTRAER ATRIBUTO DE MEMORIA CACHÉ; EXTRAER TIEMPO DE CPU; EXTRAER PSW; EXTRAER PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIONES; CARGAR Y AÑADIR; CARGAR Y AÑADIR LÓGICA; CARGAR Y AND; CARGAR Y OR EXCLUSIVA; CARGAR Y OR; CARGAR PAR DISJUNTO; CARGAR PAR DE PALABRA CUÁDRUPLE; LLAMADA DE MONITOR; MOVER LARGA; MOVER LARGA EXTENDIDA; MOVER LARGA UNICODE; MOVER CADENA; ALMACÉN NO TRANSACCIONAL; REALIZAR CÁLCULO CRIPTOGRAFICO; OBTENER PREVIAMENTE DATOS; OBTENER PREVIAMENTE DATOS RELATIVA LARGA; EMITIR INSTRUMENTACIÓN DE TIEMPO DE EJECUCIÓN; INSTRUMENTACIÓN DE TIEMPO DE EJECUCIÓN SIGUIENTE; INSTRUMENTACIÓN DE TIEMPO DE EJECUCIÓN APAGADA; INSTRUMENTACIÓN DE TIEMPO DE EJECUCIÓN ENCENDIDA; BUSCAR CADENA; BUSCAR; CADENA UNICODE; ESTABLECER MODO DE DIRECCIONAMIENTO; ALMACENAR CARACTERES BAJO MÁSCARA ALTA, cuando el campo M_3 es cero, y el código en el campo

R_1 es 6 o 7; ALMACENAR RELOJ; ALMACENAR RELOJ EXTENDIDA; ALMACENAR RELOJ RÁPIDA; ALMACENAR LISTA DE FACILIDADES EXTENDIDA; ALMACENAR PAR A PALABRA CUÁDRUPLE; PROBAR MODO DE DIRECCIONAMIENTO; ABORTAR TRANSACCION; COMIENZO DE TRANSACCIÓN (tanto TBEGIN como TBEGINC); TRADUCIR Y PROBAR EXTENDIDA; TRADUCIR Y PROBAR INVERSA EXTENDIDA; TRADUCIR EXTENDIDA; TRADUCIR UNO A UNO; TRADUCIR UNO A DOS; TRADUCIR DOS A UNO; y TRADUCIR DOS A DOS.

4. Los operandos de almacenamiento de la transacción no acceden a más de cuatro palabras óctuples. Nota: Se consideran CARGAR BAJO CONDICIÓN y ALMACENAR BAJO CONDICIÓN para referenciar el almacenamiento independientemente del código de condición. Una palabra óctuple es, por ejemplo, un grupo de 32 bytes consecutivos en un límite de 32 bytes.

5. La transacción que se ejecuta en esta CPU, o almacena por otras CPU o el subsistema de I/O, no accede a los operandos de almacenamiento en ningún bloque de 4 Kbytes que contenga los 256 bytes de almacenamiento que comienzan con la instrucción COMIENZO DE TRANSACCIÓN restringida (TBEGINC).

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

7. Las referencias de los operandos hechas por la transacción han de estar dentro de una única palabra doble, excepto que para CARGAR ACCESO MÚLTIPLE, CARGAR MÚLTIPLE, CARGAR MÚLTIPLE ALTA, ALMACENAR ACCESO MÚLTIPLE, ALMACENAR MÚLTIPLE, y ALMACENAR MÚLTIPLE ALTA, las referencias de los operandos han de estar dentro de una única palabra óctuple.

Si una transacción restringida viola cualquiera de las restricciones 1-7 enumeradas anteriormente, entonces o bien (a) se reconoce una interrupción de programa de restricción de transacción, o bien (b) la ejecución procede como si la transacción no estuviera restringida, excepto que violaciones de restricción adicionales puedan dar como resultado aún una interrupción de programa restringida de transacción. Es impredecible qué medidas se toman, y las medidas tomadas puede diferir en base a qué restricción se viole.

En ausencia de violaciones de restricción, interrupciones repetidas, o conflictos con otras CPU o el subsistema de I/O, una transacción restringida se completará finalmente, como se ha descrito anteriormente.

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 operando de almacenamiento son menos que el máximo de 4 palabras óctuples.

c. Las referencias de operando de almacenamiento están en la misma línea de caché.

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

2. Una transacción restringida no se asegura necesariamente de que se complete con éxito en su primera ejecución. No obstante, si una transacción restringida que no viola ninguna de las restricciones enumeradas se aborta, la CPU emplea circuitería para asegurar que una ejecución repetida de la transacción tenga éxito posteriormente.

3. Dentro de una transacción restringida, COMIENZO DE TRANSACCIÓN es una instrucción restringida, de este modo, no se puede anidar una transacción restringida.

4. La violación de cualquiera de las restricciones 1-7 anteriores por una transacción restringida puede dar como resultado un bucle de programa.

5. Las limitaciones de una transacción restringida son similares a las de un bucle comparar e intercambiar. Debido a la interferencia potencial de otras CPU y el subsistema de I/O, no hay garantía arquitectónica de que una instrucción COMPARAR E INTERCAMBIAR se complete nunca con el código de condición 0. Una transacción restringida puede sufrir de interferencias similares en la forma de abortar conflictos de obtención y almacén o interrupciones en caliente.

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

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 sometido a la máscara de guardado de registro general. A continuación se muestra un ejemplo.

	LH1	15.0	Contador de reintentos cero
Bucle	TBEGINC	0(0),x ‘FE00’	Conservar GR 0-13
	AHÍ	15.1	Aumentar contador
	…		
	…	Código de ejecución transaccional restringida	
	…		
	TEND		Fin de transacción

* R15 ahora contiene recuento de intentos transaccionales repetidos

Obsérvese que ambos registros 14 y 15 no están restaurados en este ejemplo. También observe que en algunos modelos, el recuento en el registro general 15 puede ser bajo si la CPU detecta la condición de abortar siguiente a la terminación de la instrucción TBEGINC, pero antes de la terminación de la instrucción AHI.

Como se observa por la CPU, las obtenciones y los almacenes hechos en el modo de ejecución transaccional no son diferentes a los realizados mientras que no está en el modo de ejecución de transacciones.

En una realización, el usuario (es decir, el que crea la transacción) selecciona si ha de ser restringida o no una transacción. Una realización de la lógica asociada con el procesamiento de transacciones restringidas, y, en particular, el procesamiento asociado con una instrucción TBEGINC, se describe con referencia a la FIG. 11. La ejecución de la instrucción TBEGINC hace que la CPU entre en el modo de ejecución transaccional restringida o permanezca en el modo de ejecución no restringida. La CPU (es decir, el procesador) que ejecuta TBEGINC realiza la lógica de la FIG. 11.

Con referencia a la FIG. 11, en base a la ejecución de una instrucción TBEGINC, se realiza una función de serialización, PASO 1100. Una función u operación de serialización incluye completar todos los accesos de almacenamiento previos conceptualmente (y, para la z/Architecture, como ejemplo, los ajustes de bit de referencia y bit de cambio relacionados) por la CPU, como se observa por otras CPU y por el subsistema de I/O, antes de que ocurran los accesos de almacenamiento conceptualmente posteriores (y los ajustes de bit de referencia y bit de cambio relacionados). La serialización afecta a la secuencia de todos los accesos de la CPU al almacenamiento y a las claves de almacenamiento, excepto las asociadas con la obtención de la entrada de la tabla de ART y la entrada de la tabla de DAT.

Como se observa por una CPU en el modo de ejecución transaccional, la serialización opera normalmente (como se ha descrito anteriormente). Como se observa por otras CPU y por el subsistema de I/O, una operación de serialización realizada mientras una CPU está en el modo de ejecución transaccional ocurre cuando la CPU abandona el modo de ejecución transaccional, o bien como resultado de una instrucción FIN DE TRANSACCIÓN que disminuye la profundidad de anidamiento de transacciones a cero (finalización normal), o bien como resultado de la transacción que se aborta.

Posterior a realizar la serialización, se hace una determinación en cuanto a si se reconoce una excepción, CONSULTA 1102. Si es así, se maneja la excepción, PASO 1104. Por ejemplo, se reconoce una excepción de operación especial y la operación se suprime si el control de ejecución transaccional, el bit 8 del registro de control 0, es 0. Como ejemplos adicionales, se reconoce una excepción de especificación y se suprime la operación, si el campo B_1, los bits 16-19 de la instrucción, es distinto de cero; se reconoce una excepción de ejecución y se suprime la operación, si el TBEGINC es el objetivo de una instrucción de tipo ejecutar; y se reconoce una excepción de operación y se suprime la operación, si la facilidad de ejecución transaccional no está instalada en la configuración. Si la CPU ya está en el modo de ejecución de transacción restringida, entonces se reconoce una excepción de programa de excepción restringida de transacción y se suprime la operación. Además, si la profundidad de anidamiento de transacciones, cuando se aumenta en 1, excediera un modelo dependiente de la profundidad de anidamiento de transacciones máxima, la transacción se aborta con el código de abortar 13. Se pueden reconocer y manejar otras o diferentes excepciones. No obstante, si no hay una excepción, entonces se hace una determinación en cuanto a si la profundidad de anidamiento de transacciones es cero, CONSULTA 1106. Si la profundidad de anidamiento de transacciones es cero, entonces la dirección de bloque de diagnóstico de transacción se considera que no es válida, PASO 1108; la PSW de abortar transacción se establece a partir de los contenidos de la PSW actual, excepto que la dirección de instrucción de la PSW de abortar transacción designe la instrucción TBEGINC, en lugar de la siguiente instrucción secuencial, PASO 1110; y los contenidos de los pares de registros generales que se designan por la máscara de guardado de registro general se guardan en una ubicación dependiente del modelo que no es accesible directamente por el programa, PASO 1112. Además, la profundidad de anidamiento se establece en 1, PASO 1114. Además, el valor eficaz de la operación de punto flotante (F) permitida y los controles de filtrado de interrupción de programa (PIFC) se establecen en cero, PASO 1316. Además, se determina el valor eficaz del control de modificación de AR (A) permitida, campo de bit 12 del campo I_2 de la instrucción, PASO 1118.

Por ejemplo, el control A eficaz es el AND lógico 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 anidamiento de transacciones es mayor que cero, entonces la profundidad de anidamiento se aumenta en 1, PASO 1120. Además, el valor eficaz de la operación de punto flotante (F) permitida se establece en cero, y el valor eficaz del control de filtrado de interrupción de programa (PIFC) no se cambia, PASO 1122. El procesamiento entonces continúa con el PASO 1118. En una realización, un inicio con éxito de la transacción da como resultado un código de condición 0. Esto concluye una realización de la lógica asociada con la ejecución de una instrucción TBEGINC.

En una realización, la comprobación de excepción proporcionada anteriormente puede ocurrir en un orden variable. Un orden particular para la comprobación de excepción es de la siguiente manera:

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

Excepción de especificación debida al campo B_1 que contiene un valor distinto de cero.

Abortar debido a exceder la profundidad de anidamiento de transacciones.

Código de condición 0 debido a una terminación normal.

Además, lo siguiente se aplica en una o más realizaciones:

1. Los registros designados para ser guardados por la máscara de guardado de registro general solamente se restauran si la transacción se aborta, no cuando la transacción termina normalmente por medio de FIN DE TRANSACCIÓN. Solamente los registros designados por la GRSM de la instrucción de COMIENZO DE TRANSACCIÓN más externa se restauran al abortar.

El campo I_2 debería designar todos los pares de registros que proporcionan valores de entrada que se cambian por una transacción restringida. De este modo, si la transacción se aborta, los valores de registro de entrada se restaurarán a sus contenidos originales cuando se vuelva a ejecutar la transacción restringida.

2. En la mayoría de los modelos, se puede realizar un mejor rendimiento, tanto en el COMIENZO DE TRANSACCIÓN como cuando una transacción se aborta, especificando el número mínimo de registros necesarios a ser guardados y restaurados en la máscara de guardado de registro general.

3. Lo siguiente ilustra los resultados de la instrucción COMIENZO DE TRANSACCIÓN (tanto TBEGIN como TBEGINC) en base a la profundidad de anidamiento de transacciones (TND) actual y, cuando la TND es distinta de cero, si la CPU está en el modo de ejecución transaccional no restringida o restringida:

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

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

Como se describe en la presente memoria, en un aspecto, una transacción restringida se asegura de su terminación, suponiendo que no contiene una condición que la haga incapaz de terminar completarla. Para asegurarse de que se complete, el procesador (por ejemplo, la CPU) que ejecuta la transacción puede tomar ciertas medidas. Por ejemplo, si una transacción restringida tiene una condición de abortar, la CPU puede temporalmente:

(a) inhibir una ejecución fuera de orden;

(b) inhibir que otras CPU accedan a ubicaciones de almacenamiento conflictivas;

(c) inducir retrasos aleatorios en el procesamiento de abortar; y/o

(d) invocar otras medidas para facilitar la terminación con éxito.

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

- Si ya está en el modo de TX restringida, se reconoce una excepción de transacción restringida.
- Si la TND (Profundidad de Anidamiento de Transacciones) actual > 0, la ejecución procede como si fuera una transacción no restringida
    - Control F eficaz establecido en cero
    - PIFC eficaz no se cambia
    - Permite que la TX no restringida externa llame a una función de servicio que puede usar o no una TX restringida.
- Si la TND actual = 0:
    - La dirección de bloque de diagnóstico de transacciones no es válida´
        - Ningún TDB especificado por instrucción almacenado al abortar
    - PSW de abortar transacción establecida en la dirección de TBEGINC
        - No la siguiente instrucción secuencial
    - Pares de registros generales designados por la GRSM guardados en una ubicación dependiente del modelo no accesibles por programa
    - Testigo de transacción formado opcionalmente (a partir del operando D_2). El testigo de transacción es un identificador de la transacción. Puede ser igual a la dirección del operando de almacenamiento u otro valor.
- Eficaz A = TBEGINC A y cualquier A externa
- TND aumentada
    - Si la TND pasa de 0 a 1, la CPU entra en el modo de TX restringida
    - De otro modo, la CPU permanece en el modo de TX no restringida
- Instrucción se completa con CC0
- Excepciones:
    - Excepción de especificación (PIC (Código de Interrupción de Programa) 0006) si el campo B_1 es distinto de cero
    - Excepción de operación especial (PIC 0013 hex) si el control de ejecución de transacción (CR0.8) es cero
    - Excepción de restricción de transacción (PIC 0018 hex) si se emite en el modo de TX restringida
    - Excepción de operación (PIC 0001) si la facilidad de ejecución transaccional restringida no está instalada
    - Excepción de ejecutar (PIC 0003) si la instrucción es el objetivo de una instrucción de tipo ejecutar
    - Código de abortar 13 si se excede la profundidad de anidamiento.
- Condiciones de abortar en transacción restringida:
    - PSW de abortar apunta a la instrucción TBEGINC
        - No la instrucción que la sigue

- Condición de abortar hace que toda la TX se reconduzca
    - Sin camino de fallo
    - La CPU toma medidas especiales para asegurar una terminación con éxito al reconducir
    - Suponiendo que no haya conflicto persistente, interrupción o violación restringida, la transacción se asegura de terminación eventual
- Violación de restricción:
    - PIC 0018 hex - indica una violación de la restricción de transacción
    - O, la transacción se ejecuta como si no estuviera restringida

Como se ha descrito anteriormente, además del procesamiento de transacciones restringida, que es opcional, en una realización, la facilidad transaccional también proporciona procesamiento de transacciones no restringida. Los detalles adicionales con respecto al procesamiento de transacciones no restringida y, en particular, el procesamiento asociado con una instrucción TBEGIN se describen con referencia a la FIG. 12. La ejecución de la instrucción TBEGIN hace que la CPU o bien entre o bien permanezca en el modo de ejecución transaccional no restringida. La CPU (es decir, el procesador) que ejecuta TBEGIN ejecuta la lógica de la FIG. 12.

Con referencia a la FIG. 12, en base a la ejecución de la instrucción TBEGIN, se realiza una función de serialización (descrita anteriormente), PASO 1200. Posterior a realizar la serialización, se hace una determinación en cuanto a si se reconoce una excepción, CONSULTA 1202. Si es así, entonces se maneja la excepción, PASO 1204. Por ejemplo, se reconoce una excepción de operación especial y se suprime la operación si el control de ejecución transaccional, el bit 8 del registro de control 0, es cero. Además, se reconoce una excepción de especificación y se suprime la operación si el control de filtrado de interrupción de programa, los bits 14-15 del campo I_2 de la instrucción, contiene el valor 3; o la dirección del primer operando no designa un límite de palabra doble. Se reconoce una excepción de operación y se suprime la operación, si la facilidad de ejecución transaccional no está instalada en la configuración; y se reconoce una excepción de ejecutar y se suprime la operación si el TBEGIN es el objetivo de una instrucción de tipo ejecutar. Además, si la CPU está en el modo de ejecución transaccional restringida, entonces se reconoce una excepción de programa de excepción de transacción restringida y se suprime la operación. Además, si la profundidad de anidamiento de transacciones, cuando se aumenta en 1, excediese la profundidad de anidamiento de transacciones máxima dependiente de un modelo, la transacción se aborta con el código de abortar 13.

Aún más, cuando el campo B_1 de la instrucción es distinto de cero y la CPU no está en el modo de ejecución transaccional, es decir, la profundidad de anidamiento de transacciones es cero, entonces se determina la accesibilidad de almacén para el primer operando. Si no se puede acceder al primer operando para almacenes, entonces se reconoce una excepción de acceso y la operación o bien se anula, suprime o bien termina, dependiendo de la condición de excepción de acceso específica. Además, se reconoce cualquier evento de alteración de almacenamiento de PER para el primer operando. Cuando el campo B_1 es distinto de cero y la CPU ya está en el modo de ejecución transaccional, es impredecible si se determina la accesibilidad de almacén para el primer operando, y los eventos de alteración de almacenamiento de 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 excepciones, se hace una determinación en cuanto a si la CPU está en el modo de ejecución transaccional (es decir, la profundidad de anidamiento de transacciones 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, PASO 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 accesible directamente por el programa.

Además, se hace una determinación en cuanto a si el campo B_1 de la instrucción es cero, CONSULTA 1210. Si el campo B_1 no es igual a cero, la dirección del primer operando se pone en la dirección de bloque de diagnóstico de transacciones, PASO 1214, y la dirección de bloque de diagnóstico de transacciones es válida. Además, la PSW de abortar transacción se establece a partir de los contenidos de la PSW actual, PASO 1216. La dirección de instrucción de la PSW de abortar transacción designa la siguiente instrucción secuencial (es decir, la instrucción que sigue a TBEGIN más externa).

Además, se hace una determinación del valor eficaz del control (A) de modificación de AR permitido, el bit 12 del campo I_2 de la instrucción, PASO 1218. El control A eficaz es el AND lógico del control A en la instrucción TBEGIN para el nivel actual y para todos los niveles externos. Además, se determina un valor eficaz del control de operación de punto flotante (F) permitida, el bit 13 del campo I_2 de la instrucción, PASO 1220. El control F eficaz es el AND lógico del control F en la instrucción TBEGIN para el nivel actual y para todos los niveles externos. Además, se determina un valor eficaz del control de filtrado de interrupción de programa (PIFC), los bits 14-15 del campo I_2 de la

instrucción, PASO 1222. El valor de PIFC eficaz es el valor más alto en la instrucción TBEGIN para el nivel actual y para todos los niveles externos.

Además, se añade un valor de uno a la profundidad de anidamiento de transacciones, PASO 1224, y la instrucción se completa con ajuste del código de condición 0, PASO 1226. Si la profundidad de anidamiento de transacciones pasa de cero a uno, la CPU entra en el modo de ejecución transaccional no restringida; de otro modo, la CPU permanece en el modo de ejecución transaccional no restringida.

Volviendo a la CONSULTA 1210, si B_1 es igual a cero, entonces la dirección de bloque de diagnóstico de transacciones no es válida, PASO 1211, y el procesamiento continúa con el PASO 1218. De manera similar, si la CPU está en modo de ejecución transaccional, CONSULTA 1206, el procesamiento continúa con el PASO 1218.

El Código de Condición resultante de la ejecución de TBEGIN incluye, por ejemplo:

0 Iniciación de transacción con éxito

1 …

2 …

3 …

Las Excepciones de Programa incluyen, por ejemplo:

- Acceso (almacén, primer operando)
- Operación (facilidad de ejecución transaccional no instalada)
- Operación especial
- Especificación
- Restricción de transacción (debida a instrucción restringida)

En una realización, la comprobación de excepciones proporcionada anteriormente puede ocurrir en un orden variable. Un orden particular para la comprobación de excepciones es de la siguiente manera:

- Excepciones con la misma prioridad como la prioridad de las condiciones de interrupción de programa para el caso general.
- Excepción de especificación debido al valor de PIFC reservado.
- Excepción de especificación debido a que la dirección del primer operando no está en un límite de palabra doble.
- Excepción de acceso (cuando el campo B_1 es distinto de cero).
- Abortar debido a exceder la profundidad de anidamiento de transacciones máxima.
- Código de condición 0 debido a la terminación normal.

Notas:

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

- Ha de ser proporcionado un bloque de diagnóstico de transacciones (TDB) accesible cuando se inicie una transacción más externa - incluso si la transacción nunca se aborta.
- Dado que es impredecible si la accesibilidad del TDB se prueba para transacciones anidadas, se debería proporcionar un TDB accesible para cualquier instrucción TBEGIN anidada.
- El rendimiento de cualquier TBEGIN en el que el campo B_1 es distinto de cero, y el rendimiento de cualquier procesamiento de abortar que ocurre para una transacción que se inició por una TBEGIN más externa en el que el campo B_1 es distinto de cero, puede ser más lento que cuando el campo B_1 es cero.

2. Los registros designados para ser guardados por la máscara de guardado de registro general solamente se restauran, en una realización, si la transacción se aborta, no cuando la transacción termina normalmente por medio de FIN DE TRANSACCIÓN. Solamente los registros designados por la GRSM de la instrucción COMIENZO DE TRANSACCIÓN más externa se restauran al abortar.

El campo I_2 debería designar todos los pares de registros que proporcionan valores de entrada que se cambian por la transacción. De este modo, si la transacción se aborta, los valores de registro de entrada se restaurarán a sus contenidos originales cuando se entre en el controlador de abortar.

3. Se espera que la instrucción COMIENZO DE TRANSACCIÓN (TBEGIN) sea seguida por una instrucción de ramificación condicional que determinará si la transacción se inició con éxito.

4. Si una transacción se aborta debido a condiciones que no dan como resultado una interrupción, la instrucción designada por la PSW de abortar transacción recibe el control (es decir, la instrucción que sigue al COMIENZO DE TRANSACCIÓN (TBEGIN) más externa). Además del código de condición establecido por la instrucción COMIENZO DE TRANSACCIÓN (TBEGIN), los códigos de condición 1-3 también se establecen cuando se aborta una transacción.

Por lo tanto, la secuencia de instrucciones que sigue a la instrucción COMIENZO DE TRANSACCIÓN más externa (TBEGIN) debería ser capaz de acomodar todos de los cuatro códigos de condición, aún cuando la instrucción TBEGIN solamente establece el código 0, en este ejemplo.

5. En la mayoría de los modelos, se puede realizar una mejora del rendimiento, tanto al COMIENZO DE TRANSACCIÓN como cuando se aborta una transacción, especificando el número mínimo de registros que necesitan ser guardados y restaurados en la máscara de guardado de registro general.

6. Mientras que está en el modo de ejecución transaccional no restringida, un programa puede llamar a una función de servicio que puede alterar registros de acceso o registros de punto flotante (incluyendo el registro de control de punto flotante). Aunque tal rutina de servicio puede guardar los registros alterados a la entrada y restaurarlos a la salida, la transacción se puede abortar antes de la salida normal de la rutina. Si el programa que llama no hace ninguna provisión para conservar estos registros mientras que la CPU está en el modo de ejecución transaccional no restringida, pueda no ser capaz de tolerar la alteración de la función de servicio de los registros.

Para evitar la alteración inadvertida de los registros de acceso mientras está en el modo de ejecución transaccional no restringida, el programa puede establecer el control de modificación de AR permitida, el bit 12 del campo I_2 de la instrucción COMIENZO DE TRANSACCIÓN, en cero. De manera similar, para evitar la alteración inadvertida de los registros de punto flotante, el programa puede establecer el control de operación de punto flotante permitida, el 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 COMIENZO DE TRANSACCIÓN (TBEGIN) están sometidas 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 sometidas a filtrado.

8. Con el fin de actualizar múltiples ubicaciones de almacenamiento de 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 almacenamiento, (b) el programa también proporciona un camino de “retroceso” a ser invocado si la transacción se aborta, y (c) el camino de retroceso emplea una palabra de bloqueo, entonces el camino de ejecución transaccional también debería probar la disponibilidad del bloqueo y, si el bloqueo no está disponible, terminar la transacción por medio de la instrucción FIN DE TRANSACCIÓN y ramificar al camino de retroceso. Esto asegura un acceso consistente a los recursos serializados, independientemente de si se actualizan de manera transaccional.

Alternativamente, el programa podría abortarse si el bloqueo no está disponible, no obstante, el procesamiento de abortar puede ser significativamente más lento que simplemente terminar la transacción a través de 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 operación de punto flotante (F) permitida es cero, el código de excepción de datos (DXC) no se establecerá en el registro de control de punto flotante como resultado de un abortar debido a una condición de excepción de programa de excepción de datos. En este escenario (el filtrado se aplica y el control F eficaz es cero), la única ubicación en la que se inspecciona el DXC es en el TDB especificado por TBEGIN. Si el controlador de abortar del programa ha de inspeccionar el DXC en tal situación, el registro general B_1 debería ser distinto de cero, de manera que se establece una dirección de bloque de diagnóstico de transacciones (TDBA) válida.

10. Si existe una alteración de almacenamiento de PER o una condición de detección de dirección cero para el TDB especificado por TBEGIN de la instrucción TBEGIN más externa, y no se aplica la supresión de evento de PER, el evento de PER se reconoce durante la ejecución de la instrucción, causando de este modo que la transacción se aborte inmediatamente, independientemente de si existe cualquier otra condición de abortar.

En una realización, la instrucción TBEGIN establece implícitamente que la dirección de abortar transacción sea la siguiente instrucción secuencial que sigue al TBEGIN. Esta dirección se pretende que sea una instrucción de

ramificación condicional que determina si ramificar o no dependiendo del código de condición (CC). Un TBEGIN con éxito establece CC0, mientras que una transacción abortada establece CC1, CC2 o CC3.

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

Además, proporciona un operando inmediato que incluye lo siguiente:

Una máscara de guardado de registro general (GRSM) que indica qué pares de registros generales han de ser guardados al comienzo de la ejecución transaccional y restaurados si se aborta la transacción;

Un bit (A) para permitir el abortar de la transacción si la transacción modifica los registros de acceso;

Un bit (F) para permitir el abortar 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 deriven la presentación real de una interrupción de programa si se aborta una transacción.

Los controles A, F y PIFC pueden ser diferentes en varios niveles de anidamiento y se pueden restaurar al nivel anterior cuando se terminan los niveles de transacción interna.

Además, el TBEGIN (o en otra realización, TBEGINC) se usa para formar un testigo de transacción. Opcionalmente, el testigo se puede hacer coincidir con un testigo formado por la instrucción TEND. Para cada instrucción TBEGIN (o TBEGINC), como ejemplo, un testigo se forma a partir de la dirección del primer operando. Este testigo se puede formar independiente de si el registro base es cero (a diferencia del ajuste de la dirección de TDB que solamente ocurre cuando el registro base es distinto de cero). Para cada instrucción FIN DE TRANSACCIÓN ejecutada con un registro base distinto de cero, se forma un testigo similar a partir de su operando de almacenamiento. Si los testigos no coinciden, se puede reconocer una excepción de programa para alertar al programa de una instrucción no emparejada.

La coincidencia de testigos proporciona un mecanismo destinado a mejorar la fiabilidad del software asegurando que una declaración TEND esté emparejada correctamente con una TBEGIN (o TBEGINC). Cuando una instrucción TBEGIN se ejecuta en un nivel de anidamiento particular, se forma un testigo a partir de la dirección del operando de almacenamiento que identifica este caso de una transacción. Cuando se ejecuta una instrucción TEND correspondiente, se forma un testigo a partir de la dirección del operando de almacenamiento de la instrucción, y la CPU compara el testigo de inicio para el nivel de anidamiento con el testigo final. Si los testigos no coinciden, se reconoce una condición de excepción. Un modelo puede implementar una coincidencia de testigo solamente para un cierto número de niveles de anidamiento (o para niveles sin anidamiento). El testigo no puede implicar a todos los bits de la dirección del operando de almacenamiento, o los bits se pueden combinar a través de comprobación aleatoria u otros métodos. Un testigo se puede formar 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:
    - Si $B_1 \neq 0$, la dirección de bloque de diagnóstico de transacciones establecida a partir de la dirección del primer operando.
    - La PSW de abortar transacción establecida en la siguiente dirección de instrucción secuencial.
    - Los pares de registros generales designados por el campo I_2 se guardan en una ubicación dependiente del modelo.
        - No directamente accesible por el programa
- Controles de PIFC, A y F eficaces calculados
    - A eficaz = TBEGIN A y cualquier A externa
    - F eficaz = TBEGIN F y cualquier F externa
    - PIFC eficaz = max(TBEGIN PIFC, cualquier PIFC externa)
- Profundidad de anidamiento de transacciones (TND) aumentada
- Si la TND pasa de 0 a 1, la CPU entra en el modo de ejecución transaccional

- Código de condición establecido en cero
    - Cuando la instrucción siguiente a TBEGIN recibe el control:
        - Éxito de TBEGIN indicado por CC0
        - Transacción abortada indicada por CC distinto de cero.
- Excepciones:
    - Código de abortar 13 si se excede la profundidad de anidamiento
    - Excepción de acceso (uno de varios PIC) si el campo B_1 es distinto de cero, y no se puede acceder al operando de almacenamiento para una operación de almacén
    - Ejecutar excepción (PIC 0003) si la instrucción TBEGIN es el objetivo de una instrucción de tipo ejecutar
    - Excepción de operación (PIC 0001) si la facilidad de ejecución transaccional no está instalada
    - PIC 0006 si cualquiera de los dos
        - PIFC no es válido (valor de 3)
        - Dirección del segundo operando no es una palabra doble alineada
    - PIC 0013 hex si el control de ejecución transaccional (CR0.8) es cero
    - PIC 0018 hex si se emite en modo de TX restringida

Como se ha indicado anteriormente, una transacción, o bien restringida o bien no restringida, se puede finalizar mediante una instrucción FIN DE TRANSACCIÓN (TEND). Detalles adicionales con respecto al procesamiento de una instrucción de fin de transacción (TEND) se describen con referencia a la FIG. 13. La CPU (es decir, el procesador) que ejecuta la TEND realiza la lógica de la FIG. 13.

Con referencia a la FIG. 13, inicialmente, en base a la obtención del procesador (por ejemplo, búsqueda, recepción, etc.) de la instrucción TEND, se realizan varias comprobaciones de excepciones, y si hay una excepción, CONSULTA 1300, entonces se maneja la excepción, PASO 1302. Por ejemplo, si la FIN DE TRANSACCIÓN es el objetivo de una instrucción de tipo ejecutar, la operación se suprime y se reconoce una excepción de ejecutar; y se reconoce una excepción de operación especial y la operación se suprime si el control de ejecución transaccional, el bit 8 de CR0, es cero. Aún más, se reconoce una excepción de operación y se suprime la operación, si la facilidad de ejecución transaccional no está instalada en la configuración.

Volviendo a la CONSULTA 1300, si no se reconoce una excepción, entonces la profundidad de anidamiento de transacciones disminuye (por ejemplo, en uno), PASO 1304. Se hace una determinación en cuanto a si la profundidad de anidamiento transaccional es cero siguiendo a la disminución, CONSULTA 1306. Si la profundidad de anidamiento de transacciones es cero, entonces se comprometen todos los accesos de almacén hechos por la transacción (y otras transacciones dentro del nido de transacciones, en su caso, de las cuales es parte esta transacción), PASO 1308. Además, la CPU abandona el modo de ejecución transaccional, PASO 1310, y la instrucción se completa, PASO 1312.

Volviendo a la CONSULTA 1306, si la profundidad de anidamiento de transacciones no es igual a cero, entonces la instrucción FIN DE TRANSACCIÓN sólo termina.

Si la CPU está en el modo de ejecución de transacciones al comienzo de la operación, el código de condición se establece en 0; de otro modo, el código de condición se establece en 2.

Se observa que el control de operación de punto flotante (F) permitida eficaz, el control de modificación de AR (A) permitida y el control de filtrado de interrupción del programa (PIFC) se reinician a sus valores respectivos antes de la instrucción COMIENZO DE TRANSACCIÓN que inició el nivel que se termina. Además, se realiza una función de serialización a la terminación de la operación.

La obtención de la instrucción PER y los eventos de fin de transacción que se reconocen a la terminación de la instrucción FIN DE TRANSACCIÓN más externa no dan como resultado que la transacción se aborte.

En un ejemplo, la instrucción TEND también incluye un campo base B_2 y un campo de desplazamiento D_2, que se combinan (por ejemplo, se añaden) para crear una dirección de segundo operando. En este ejemplo, se puede realizar coincidencia de testigo. Por ejemplo, cuando B_2 es distinto de cero, los bits seleccionados de la dirección de segundo operando se hacen coincidir con un testigo de transacción formado por la TBEGIN correspondiente. Si hay una discrepancia, hay una excepción (por ejemplo, PIC 0006).

Según un aspecto, la ejecución de ciertas instrucciones y, por lo tanto, la modificación y/o acceso de ciertos registros, se controla de manera selectiva en base a uno o más controles de la instrucción COMIENZO DE TRANSACCIÓN. Estos controles se usan para determinar selectivamente si se pueden ejecutar tipos particulares de instrucciones dentro de una transacción dada. Por ejemplo, dado que ciertos tipos de registros, tales como registros de acceso, registros de punto flotante y/o el registro de control de punto flotante, no se restauran en un abortar de la transacción, la rutina de recuperación del controlador de abortar de la transacción no puede ser capaz de acomodar la modificación de tales registros o incluso la ejecución de instrucciones de un contexto de punto flotante. Como se usa en la presente memoria, un contexto de punto flotante incluye cualquier instrucción que pueda inspeccionar o alterar los registros de punto flotante o el registro de control de punto flotante. De este modo, se emplea un mecanismo para indicar si se tolera tal modificación/acceso.

En particular, dado que el controlador de abortar representa el procesamiento en el nivel más externo, y dado que la modificación del registro de acceso o una operación de punto flotante puede haber ocurrido en un nivel de anidamiento interno, se proporciona un mecanismo de indicación de si se permite la modificación de los registros o la ejecución de las instrucciones de contexto. Este mecanismo incluye, por ejemplo, controles sobre la instrucción COMIENZO DE TRANSACCIÓN.

En una realización, los controles incluyen el control “A”, que es el control de modificación de registro de acceso permitida, y el control “F”, que es el control de operación de punto flotante permitida. Las transacciones se pueden anidar y, por lo tanto, hay un conjunto de controles para cada nivel de anidamiento. El control A y F eficaz es un AND lógico de todos los controles del mismo tipo en todos los niveles de anidamiento. De este modo, el control eficaz es el menos permisivo del nivel de anidamiento actual y todos los niveles inferiores. No obstante, en una transacción anidada, el control eficaz se restaura al del nivel de anidamiento inferior cada vez que se ejecuta una instrucción FIN DE TRANSACCIÓN. Por ejemplo, una serie de instrucciones TBEGIN anidadas pueden tener el control A establecido en 1, 1, 0, 1 y 0 en los niveles de anidamiento 1, 2, 3, 4 y 5. De este modo, en las profundidades de anidamiento 1 y 2, se permite la modificación del registro de acceso, pero en el nivel de anidamiento 3 y superior, no se permite la modificación del registro de acceso. A medida que el programa pasa de vuelta del nivel de anidamiento 3 al 2, llega a ser permitida de nuevo la modificación del registro de acceso. Una operación similar se aplica al control F.

Una realización de actualización de los controles a medida que disminuye el nivel de anidamiento se describe con referencia a la FIG. 14. En un ejemplo, un procesador realiza esta lógica. Inicialmente, se inicia una transacción, en base a, por ejemplo, la ejecución de una instrucción COMIENZO DE TRANSACCIÓN, PASO 1400. Esta transacción puede ser o no parte de un nido de transacciones. La instrucción COMIENZO DE TRANSACCIÓN incluye uno o más controles usados para indicar si se pueden actualizar ciertos registros/contexto y, por lo tanto, si se pueden ejecutar ciertas instrucciones. Por ejemplo, una instrucción TBEGIN incluye los controles A y F, cada uno de los cuales se establece respectivamente en un 0 si no se permite la modificación apropiada, o un 1, si se permite la modificación. Además, la instrucción TBEGINC incluye el control A, que se establece de manera similar en 0 o 1. Los valores de estos controles se usan, en base a la ejecución de la instrucción COMIENZO DE TRANSACCIÓN, para determinar el valor eficaz del control A y/o el valor eficaz del control F, como se ha descrito anteriormente, PASO 1402.

En algún punto, una transacción puede finalizar, PASO 1404. Se hace una determinación en cuanto a si este es el final de la transacción más externa (es decir, la transacción más externa de un nido de transacciones o el final de una transacción que no es parte de un nido de transacciones), CONSULTA 1406. Si no es el final de la transacción más externa, entonces se actualizan uno o más controles eficaces, PASO 1408. En un ejemplo, los controles se actualizan volviendo a calcular los controles eficaces sin incluir los controles de la transacción que ha finalizado. Volviendo a CONSULTA 1406, si es el final de la transacción más externa, entonces el procesamiento se completa.

Anteriormente se ha descrito un mecanismo para controlar selectivamente dentro de una transacción la ejecución de ciertas instrucciones. Por ejemplo, se proporcionan controles que indican si se permite ejecutar un tipo particular de instrucción y, por lo tanto, se permite modificar sus registros/contexto respectivos. Por ejemplo, se proporciona un control que indica si se pueden modificar los registros de acceso y, por lo tanto, si se puede ejecutar una instrucción que modifique tales registros. Como ejemplo adicional, se proporciona un control que indica si se pueden realizar operaciones de punto flotante.

Además, se han proporcionado anteriormente unos medios eficaces para actualizar múltiples objetos no contiguos en memoria sin serialización clásica (de grano grueso), tal como bloqueo, que proporciona un potencial para una mejora de rendimiento de multiprocesador significativa. Es decir, múltiples objetos no contiguos se actualizan sin la imposición de un orden de acceso de almacenamiento de grano grueso que se proporciona mediante técnicas clásicas, como tales bloqueos y semáforos. La ejecución especulativa se proporciona sin una configuración de recuperación onerosa, y se ofrecen transacciones restringidas para actualizaciones simples y de huella pequeña.

La ejecución transaccional se puede usar en una variedad de escenarios, incluyendo, pero no limitados a, alineamiento parcial, procesamiento especulativo y la elisión de bloqueo. En alineación parcial, la región parcial a ser incluida en el camino ejecutado se ajusta en TBEGIN/TEND. TABORT se puede incluir en la misma para revertir el estado en una salida lateral. Para especulación, tal como en Java, las comprobaciones nulas en punteros sin

referencia se pueden retrasar hasta el borde del bucle usando una transacción. Si el puntero es nulo, la transacción puede abortarse de manera segura usando TABORT, que se incluye dentro de TBEGIN/TEND.

En cuanto a la elisión de bloqueo, se describe un ejemplo de su uso con referencia a las FIG. 15A-15B y el fragmento de código que se proporciona a continuación.

La FIG. 15A representa una lista doblemente enlazada 1500 de una pluralidad de elementos de cola 1502a-1502d. Ha de ser insertado un nuevo elemento de cola 1502e en la lista doblemente enlazada de elementos de cola 1500. Cada elemento de cola 1502a-1502e incluye un puntero hacia delante 1504a-1504e y un puntero hacia atrás 1506a-1506e. Como se muestra en la FIG. 15B, para añadir el elemento de cola 1502e entre los elementos de cola 1502b y 1502c, (1) el puntero hacia atrás 1506e se establece para apuntar al elemento de cola 1502b, (2) el puntero hacia delante 1504e se establece para apuntar al elemento de cola 1502c, (3) el puntero hacia atrás 1506c se establece para apuntar al elemento de cola 1502e y (4) el puntero hacia adelante 1504b se establece para apuntar al elemento de cola 1502e.

Un fragmento de código de ejemplo correspondiente a las FIG. 15A-15B está a continuación:

* R1 - dirección del nuevo elemento de cola a ser insertado.

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

```
NEW       USING    QEL, R1
CURR      USING    QEL, R2
                   LHI      R15, 10          Recuento de reintentos de carga
LOOP      TBEGIN   TDB, X'C000'             Comenzar transacción (guardar GR 0-3)
                   JNZ      ABORTED          CC distinto de cero significa abortado
          LG       R3, CURR, BWD            Apuntar a elemento previo
PREV      USING    QEL, R3                  Hacerlo direccionable
                   STG      R1, PREV. FWD    Actualizar prev. adelante ptr.
                   STG      R1, CURR. BWD    Actualizar prev. atrás ptr.
                   STG      R2, NEW. FWD     Actualizar nuevo adelante ptr.
                   STG      R3, NEW. BWD     Actualizar nuevo atrás ptr.
                   TEND                      Fin de transacción.
                   …
ABORTED            JO       NO_RETRY         CC3: Abortar no reintentable
                   JCT      R15, LOOP        Transacción reintentar unas pocas veces.
                   J        NO_RETRY         Sin encanto después de 10x, hazlo de la forma difícil
```

En un ejemplo, si la transacción se usa para la elisión de bloqueo, pero el camino de retroceso usa un bloqueo, la transacción ha de ser al menos obtener la palabra de bloqueo para ver que está disponible. El procesador asegura que la transacción se aborta, si otra CPU accede al bloqueo de manera no transaccional.

Como se usa en la presente memoria, almacenamiento, almacenamiento central, almacenamiento principal, memoria y memoria principal se usan de manera intercambiable, a menos que se señale de otro modo, implícitamente mediante su uso o explícitamente. Además, mientras que en una realización, una transacción que se retrasa eficazmente incluye retrasar el compromiso de los almacenes transaccionales con la memoria principal hasta la terminación de una transacción seleccionada; en otra realización, una transacción que retrasa eficazmente incluye permitir actualizaciones transaccionales a la memoria, pero mantener los valores antiguos y restaurar la memoria a los valores antiguos al abortar.

Como apreciará un experto en la técnica, se pueden incorporar uno o más aspectos como sistema, método o producto de programa de ordenador. Por consiguiente, uno o más aspectos pueden tomar la forma de una realización completamente de hardware, una realización completamente de software (incluyendo microprograma, software residente, microcódigo, etc.) o una realización que combina aspectos de software y de hardware a todos los que se puede hacer referencia de manera general en la presente memoria como “circuito”, “módulo” o “sistema”. Además, uno o más aspectos pueden tomar la forma de un producto de programa de ordenador incorporado en uno o más medios legibles por ordenador que tiene un código de programa legible por ordenador incorporado en el mismo.

Se puede utilizar cualquier combinación de uno o más medios legibles por ordenador. El medio legible por ordenador puede ser un medio de almacenamiento legible por ordenador. Un medio de almacenamiento legible por ordenador puede ser, por ejemplo, pero no limitado a, un sistema, aparato o dispositivo electrónico, magnético, óptico, electromagnético, de infrarrojos o semiconductor, o cualquier combinación adecuada de los precedentes. Ejemplos más específicos (una lista no exhaustiva) del medio de almacenamiento legible por ordenador incluyen los siguientes: una conexión eléctrica que tiene uno o más cables, un disquete de ordenador portátil, un disco duro, una memoria de acceso aleatorio (RAM), una memoria de sólo lectura (ROM), una memoria de sólo lectura programable borrable (EPROM o memoria rápida), una fibra óptica, una memoria de sólo lectura de disco compacto (CD-ROM) portátil, un dispositivo de almacenamiento óptico, un dispositivo de almacenamiento magnético, o cualquier combinación adecuada de los precedentes. En el contexto de este documento, un medio de almacenamiento legible por ordenador puede ser cualquier medio tangible que pueda contener o almacenar un programa para su uso por o en conexión con un sistema, aparato o dispositivo de ejecución de instrucciones.

Con referencia ahora a la FIG. 16, en un ejemplo, un producto de programa de ordenador 1600 incluye, por ejemplo, uno o más medios de almacenamiento legibles por ordenador no transitorios 1602 para almacenar medios o lógica de código de programa legible por ordenador 1604 en el mismo para proporcionar y facilitar una o más realizaciones.

El código de programa incorporado en un medio legible por ordenador se puede transmitir usando un medio apropiado, incluyendo, pero no limitado a, inalámbrico, cableado, cable de fibra óptica, RF, etc., o cualquier combinación adecuada de los precedentes.

El código de programa de ordenador para llevar a cabo operaciones para una o más realizaciones puede estar escrito en cualquier combinación de uno o más lenguajes de programación, incluyendo un lenguaje de programación orientado a objetos, tal como Java, Smalltalk, C++ o similares, y lenguajes de programación de procedimientos convencionales, tales como el lenguaje de programación “C”, ensamblador o lenguajes de programación similares. El código del programa puede ejecutarse completamente en el ordenador del usuario, parcialmente en el ordenador del usuario, como un paquete de software autónomo, parcialmente en el ordenador del usuario y parcialmente en un ordenador remoto o completamente en el ordenador o servidor remoto. En este último escenario, el ordenador remoto puede estar conectado al ordenador del usuario a través de cualquier tipo de red, incluyendo una red de área local (LAN) o una red de área extensa (WAN), o la conexión se puede hacer a un ordenador externo (por ejemplo, , a través de Internet usando un Proveedor de Servicios de Internet).

Se describen en la presente memoria una o más realizaciones con referencia a ilustraciones de diagramas de flujo y/o diagramas de bloques de métodos, aparatos (sistemas) y productos de programas de ordenador. Se entenderá que cada bloque de las ilustraciones del diagrama de flujo y/o los diagramas de bloques, y las combinaciones de bloques en las ilustraciones del diagrama de flujo y/o los diagramas de bloques, se pueden implementar mediante instrucciones de programa de ordenador. Estas instrucciones de programa de ordenador se pueden proporcionar 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 manera que las instrucciones, que se ejecutan a través del procesador del ordenador u otro aparato de procesamiento de datos programable, creen medios para implementar las funciones/actos especificados en el diagrama de flujo y/o bloque o bloques del diagrama de bloques.

Estas instrucciones de programa de ordenador también se pueden almacenar en un medio legible por ordenador que puede dirigir un ordenador, otro aparato de procesamiento de datos programable u otros dispositivos a funcionar de una manera particular, de manera que las instrucciones almacenadas en el medio legible por ordenador produzcan un artículo de fabricación que incluye instrucciones que implementan la función/acto especificado en el diagrama de flujo y/o bloque o bloque de diagrama de bloques.

Las instrucciones de programa de ordenador también se pueden cargar en un ordenador, otro aparato de procesamiento de datos programable u otros dispositivos para hacer que una serie de pasos operativos se realicen en el ordenador, otros aparatos programables u otros dispositivos para producir un proceso implementado por ordenador de manera que las instrucciones que se ejecutan en el ordenador u otro aparato programable proporcionen procesos para implementar las funciones/actos especificados en el diagrama de flujo y/o bloque o bloques de diagrama de bloques.

El diagrama de flujo y los diagramas de bloques en las figuras ilustran la arquitectura, la funcionalidad y la operación de posibles implementaciones de sistemas, métodos y productos de programas de ordenador según diversas

realizaciones. En este sentido, cada bloque en el diagrama de flujo o diagramas de bloques puede representar un módulo, segmento o parte de código, que comprende una o más instrucciones ejecutables para implementar la función o las funciones lógicas especificadas. También se debería observar que, en algunas implementaciones alternativas, las funciones señaladas en el bloque pueden ocurrir fuera del orden señalado en las figuras. Por ejemplo, dos bloques mostrados en sucesión, de hecho, se pueden ejecutar de manera sustancialmente concurrente, o los bloques algunas veces se pueden ejecutar en orden inverso, dependiendo de la funcionalidad implicada. También se observará que cada bloque de los diagramas de bloques y/o la ilustración del diagrama de flujo, y las combinaciones de bloques en los diagramas de bloques y/o la ilustración del diagrama de flujo, se pueden implementar por sistemas basados en hardware de propósito especial que realizan las funciones o actos especificados, o combinaciones de hardware de propósito especial e instrucciones de ordenador.

Además de lo anterior, uno o más aspectos se pueden proporcionar, ofrecer, desplegar, gestionar, atender, etc. por un proveedor de servicios que ofrece gestión de entornos de cliente. Por ejemplo, el proveedor de servicios puede crear, mantener, soportar, etc. un código de ordenador y/o una infraestructura de ordenador que realice uno o más aspectos para uno o más clientes. A cambio, el proveedor de servicios puede recibir el pago del cliente bajo una suscripción y/o un acuerdo de pago, como ejemplos. Además o alternativamente, el proveedor de servicios puede recibir el pago de la venta de contenido publicitario a uno o más terceros.

En un aspecto, se puede desplegar una aplicación para realizar una o más realizaciones. Como ejemplo, el despliegue de una aplicación comprende proporcionar una infraestructura informática operable para realizar una o más realizaciones.

Como aspecto adicional, se puede desplegar una infraestructura informática que comprenda integrar código legible por ordenador en un sistema informático, en el que el código en combinación con el sistema informático es capaz de realizar una o más realizaciones.

Como un aspecto adicional más, se puede proporcionar un proceso para integrar una infraestructura informática que comprende integrar código legible por ordenador en un sistema informático. El sistema informático comprende un medio legible por ordenador, en el que el medio de ordenador comprende una o más realizaciones. El código en combinación con el sistema informático es capaz de realizar una o más realizaciones.

Aunque se han descrito anteriormente varias realizaciones, estas son solamente ejemplos. Por ejemplo, los entornos informáticos de otras arquitecturas que se pueden usar incorporan y usan una o más realizaciones. Además, se pueden usar diferentes instrucciones, formatos de instrucción, campos de instrucción y/o valores de instrucción. Además, se pueden proporcionar/usar restricciones/limitaciones diferentes, otras y/o adicionales. Son posibles muchas variaciones.

Además, otros tipos de entornos informáticos pueden beneficiarse y ser usados. Como ejemplo, es utilizable un sistema de procesamiento de datos adecuado para almacenar y/o ejecutar 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, una memoria local empleada durante la ejecución real del código de programa, un almacenamiento masivo y una memoria caché que proporcionan almacenamiento temporal de al menos algún código de programa con el fin de reducir el número de veces que se debe recuperar el código del almacenamiento masivo durante la ejecución

Los dispositivos de Entrada/Salida o I/O (incluyendo, pero no limitados a, teclados, visualizadores, dispositivos de apuntamiento, DASD, cinta, CD, DVD, memorias USB y otros medios de memoria, etc.) se pueden acoplar al sistema o bien directamente o bien a través de controladores de I/O intervinientes. Los adaptadores de red también se pueden acoplar al sistema para permitir que el sistema de procesamiento de datos llegue a ser acoplado a otros sistemas de procesamiento de datos o impresoras remotas o dispositivos de almacenamiento a través de redes privadas o públicas intervinientes. Módems, módems de cable y tarjetas Ethernet son sólo unos pocos de los tipos de adaptadores de red disponibles.

Con referencia a la FIG. 17, se representan componentes representativos de un sistema de Ordenador Central 5000 para implementar una o más realizaciones. El ordenador central 5000 representativo comprende una o más CPU 5001 en comunicación con la memoria de ordenador (es decir, almacenamiento central) 5002, así como interfaces de I/O para dispositivos de medios de almacenamiento 5011 y redes 5010 para comunicarse con otros ordenadores o SAN y similares. La CPU 5001 es compatible con una arquitectura que tiene un conjunto de instrucciones de arquitectura y una funcionalidad de arquitectura. La CPU 5001 puede tener una traducción de registros de acceso (ART) 5012, que incluye un almacenador temporal de ART anticipada (ALB) 5013, para seleccionar un espacio de direcciones a ser usado por la traducción dinámica de direcciones (DAT) 5003 para transformar direcciones de programas (direcciones virtuales) en direcciones reales de memoria. Una DAT típicamente incluye un almacenador temporal de traducción anticipada (TLB) 5007 para almacenar en caché las traducciones, de modo que los accesos posteriores al bloque de la memoria de ordenador 5002 no requieren el retraso de la traducción de direcciones. Típicamente, una memoria caché 5009 se emplea entre la memoria de ordenador 5002 y el procesador 5001. La memoria caché 5009 puede ser jerárquica teniendo una memoria caché grande disponible para más de una CPU y memorias caché más pequeñas y más rápidas (nivel inferior) entre la memoria caché grande y cada CPU. En

algunas implementaciones, las memorias caché de nivel inferior se dividen para proporcionar memorias caché de nivel bajo separadas para la obtención de instrucciones y accesos de datos. En una realización, para la facilidad de TX, un bloque de diagnóstico de transacciones (TDB) 5100 y uno o más almacenadores temporales 5101 se pueden almacenar en una o más de la memoria caché 5009 y la memoria 5002. En un ejemplo, en el modo de TX, los datos se almacenan inicialmente en un almacenador temporal de TX, y cuando termina el modo de TX (por ejemplo, TEND más externa), los datos en el almacenador temporal se almacenan (se comprometen) en la memoria, o si hay un abortar, se descartan los datos en el almacenador temporal.

En una realización, una instrucción se obtiene de la memoria 5002 mediante una unidad de búsqueda de instrucciones 5004 a través de una memoria 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 las unidades de ejecución de instrucciones 5008. Típicamente, se emplean varias unidades de ejecución 5008, por ejemplo, una unidad de ejecución aritmética, una unidad de ejecución de punto flotante y una unidad de ejecución de instrucciones de ramificación. Además, en una realización de la facilidad de TX, se pueden emplear varios controles de TX 5110. La instrucción se ejecuta por la unidad de ejecución, accediendo a los operandos desde registros especificados por instrucción o memoria según sea necesario. Si se ha de acceder (cargar o almacenar) a un operando desde la memoria 5002, una unidad de carga/almacenamiento 5005 típicamente maneja el acceso bajo el control de la instrucción que se ejecuta. Las instrucciones se pueden ejecutar en circuitos de hardware o en microcódigo (microprograma) interno o mediante una combinación de ambos.

Según un aspecto de la facilidad de TX, el procesador 5001 también incluye una PSW 5102 (por ejemplo, PSW de TX y/o de abortar), una profundidad de anidamiento 5104, una TDBA 5106 y uno o más registros de control 5108.

Como se ha señalado, un sistema informático incluye información en el almacenamiento local (o principal), así como direccionamiento, protección, y grabación de referencias y cambios. Algunos aspectos del direccionamiento incluyen el formato de direcciones, el concepto de espacios de direcciones, los diversos tipos de direcciones y la manera en que un tipo de dirección se traduce a otro tipo de dirección. Algo del almacenamiento principal incluye ubicaciones de almacenamiento asignadas permanentemente. El almacenamiento principal dota al sistema con un almacenamiento de datos de acceso rápido directamente direccionable. Tanto los datos como los programas han de ser cargados en el almacenamiento principal (desde los dispositivos de entrada) antes de que se puedan procesar.

El almacenamiento principal puede incluir uno o más almacenamientos de almacenamiento temporal de acceso más rápido y más pequeños, algunas veces llamados memorias caché. Una memoria caché típicamente está asociada físicamente con una CPU o un procesador de I/O. Los efectos, excepto en el rendimiento, de la construcción física y el uso de distintos medios de almacenamiento generalmente no son observables por el programa.

Se pueden mantener memorias caché separadas para instrucciones y para operandos de datos. La información dentro de una memoria caché se mantiene en bytes contiguos en un límite integral llamado bloque de caché o línea de caché (o línea, para abreviar). Un modelo puede proporcionar una instrucción EXTRAER ATRIBUTO DE MEMORIA CACHÉ que devuelve el tamaño de una línea de caché en bytes. Un modelo también puede proporcionar instrucciones OBTENER PREVIAMENTE DATOS y OBTENER PREVIAMENTE DATOS RELATIVA LARGA que efectúan la obtención previa de almacenamiento en la memoria caché de datos o instrucciones o la liberación de datos de la memoria 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 denomina byte, que es el bloque de construcción básico de todos los formatos de información. Cada ubicación de bytes en el almacenamiento se identifica mediante un número entero no negativo único, que es la dirección de esa ubicación de bytes o, simplemente, la dirección de bytes. Las ubicaciones de bytes adyacentes tienen direcciones consecutivas, comenzando con 0 a la izquierda y procediendo en una secuencia de izquierda a derecha. Las direcciones son números enteros binarios sin signo y son de 24, 31 o 64 bits.

La información se transmite entre el almacenamiento y una CPU o un subsistema de canal de un byte, o un grupo de bytes, a la vez. A menos que se especifique de otro modo, por ejemplo, en la z/Architecture, un grupo de bytes en el almacenamiento se direcciona por el byte de más a la izquierda del grupo. El número de bytes en el grupo está o bien implícita o bien explícitamente especificado por la operación a ser realizada. Cuando se usa en una operación de CPU, un grupo de bytes se llama campo. Dentro de cada grupo de bytes, por ejemplo, en la z/Architecture, los bits se numeran en una secuencia de izquierda a derecha. En la z/Architecture, los bits de más a la izquierda algunas veces se conocen como los bits de “orden alto” y los bits de más a la derecha como los bits de “orden bajo”. No obstante, los números de bits no son direcciones de almacenamiento. Solamente se pueden direccionar los bytes. Para operar sobre bits individuales de un byte en el almacenamiento, se accede a todo el byte. Los bits en un byte se numeran de 0 hasta 7, de izquierda a derecha (por ejemplo, en la z/Architecture). Los bits en una dirección se pueden numerar 8-31 o 40-63 para direcciones de 24 bits, o 1-31 o 33-63 para direcciones de 31 bits; se numeran 0-63 para direcciones de 64 bits. En un ejemplo, los bits 8-31 y 1-31 se aplican a las direcciones que están en una ubicación (por ejemplo, un registro) que tiene 32 bits de ancho, mientras que los bits 40-63 y 33-63 se aplican a direcciones que están en una ubicación amplia de 64 bits. Dentro de cualquier otro formato de longitud fija de múltiples bytes, los bits que componen el formato se numeran consecutivamente a partir de 0. Con los propósitos

de detección de errores, y preferiblemente para corrección, se pueden transmitir uno o más bits de comprobación con cada byte o con un grupo de bytes. Tales bits de comprobación se generan automáticamente por la máquina y no se pueden controlar directamente por el programa. Las capacidades de almacenamiento se expresan en número de bytes. Cuando la longitud de un campo de operando de almacenamiento está implícita por el código de operación de una instrucción, se dice que el campo tiene una longitud fija, que puede ser uno, dos, cuatro, ocho o dieciséis bytes. Los campos más grandes pueden estar implícitos para algunas instrucciones. Cuando la longitud de un campo de operando de almacenamiento no está implícita sino que se expresa explícitamente, 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 pone en el almacenamiento, se sustituyen los contenidos solamente de aquellas ubicaciones de bytes que se incluyen en el campo designado, aún cuando el ancho del camino físico al almacenamiento pueda ser mayor que la longitud del campo que se almacena.

Ciertas unidades de información han de estar 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. Se dan nombres especiales a los 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 las instrucciones. Una palabra es un grupo de cuatro bytes consecutivos en un límite de cuatro bytes. Una palabra doble es un grupo de ocho bytes consecutivos en un límite de ocho bytes. Una palabra cuádruple es un grupo de 16 bytes consecutivos en un límite de 16 bytes. Una palabra óctuple 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 palabras óctuples, la representación binaria de la dirección contiene uno, dos, tres, cuatro o cinco bits cero de más a la derecha, respectivamente. Las instrucciones han de estar en los límites integrales de dos bytes. Los operandos de almacenamiento de la mayoría de las instrucciones no tienen requisitos de alineación de límites.

En los dispositivos que implementan memorias caché separadas para instrucciones y operandos de datos, se puede experimentar un retraso significativo si el programa se almacena en una línea de caché desde la cual las instrucciones se obtienen posteriormente, independientemente de si el almacén altera las instrucciones que se obtienen posteriormente.

En un ejemplo, la realización se puede poner en práctica mediante software (algunas veces conocido como código interno bajo licencia, microprograma, microcódigo, milicódigo, picocódigo y similares, cualquiera de los cuales sería coherente con una o más realizaciones). Con referencia a la FIG. 17, se puede acceder al código de programa de software que incorpora uno o más aspectos mediante el procesador 5001 del sistema ordenador central 5000 desde dispositivos de medios de almacenamiento a largo plazo 5011, tales como una unidad de CD-ROM, una unidad de cinta o una unidad de disco duro. El código de programa de software se puede incorporar en cualquiera de una variedad de medios conocidos para su uso con un sistema de procesamiento de datos, tales como un disquete, un disco duro o un CD-ROM. El código se puede distribuir en tales medios, o se puede distribuir a los usuarios desde la memoria de ordenador 5002 o el almacenamiento de un sistema informático sobre una red 5010 a otros sistemas informáticos para su uso por los usuarios de tales otros sistemas.

El código de programa de software incluye un sistema operativo que controla la función y la interacción de los diversos componentes del ordenador y uno o más programas de aplicaciones. El código de programa normalmente se pagina del dispositivo de medios de almacenamiento 5011 al almacenamiento de ordenador de velocidad relativamente más alta 5002 donde está disponible para procesamiento por el procesador 5001. Las técnicas y métodos para incorporar el código de programa de software en la memoria, en medios físicos y/o distribuir código software a través de redes son bien conocidos y no se tratarán más en la presente memoria. El código de programa, cuando se crea y almacena en un medio tangible (incluyendo, pero no limitado a, módulos de memoria electrónica (RAM), memoria rápida, Discos Compactos (CD), DVD, cinta magnética y similares, a menudo se conoce como “producto de programa de ordenador”. El medio de producto de programa de ordenador típicamente es legible por un circuito de procesamiento, preferiblemente en un sistema informático para su ejecución por el circuito de procesamiento.

La FIG. 18 ilustra una estación de trabajo representativa o un sistema de hardware de servidor en el que se pueden poner en práctica una o más realizaciones. El sistema 5020 de la FIG. 18 comprende un sistema informático base 5021 representativo, tal como un ordenador personal, una estación de trabajo o un servidor, incluyendo dispositivos periféricos opcionales. El sistema informático base 5021 incluye uno o más procesadores 5026 y un bus empleado para conectar y habilitar la comunicación entre el procesador o los procesadores 5026 y los otros componentes del sistema 5021 según técnicas conocidas. El bus conecta el procesador 5026 a la memoria 5025 y al almacenamiento a largo plazo 5027 que puede incluir un disco duro (incluyendo cualquiera de medios magnéticos, CD, DVD y memoria rápida, por ejemplo) o una unidad de cinta, por ejemplo. El sistema 5021 también podría incluir un adaptador de interfaz de usuario, que conecta el microprocesador 5026 a través del bus a uno o más dispositivos de interfaz, tales como un teclado 5024, un ratón 5023, una impresora/escáner 5030 y/u otros dispositivos de interfaz, que pueden ser cualquier dispositivo de interfaz de usuario, tal como una pantalla sensible al tacto, almohadilla de entrada digitalizada, etc. El bus también conecta un dispositivo de visualización 5022, tal como una pantalla o monitor LCD, al microprocesador 5026 a través de un adaptador de visualizador.

El sistema 5021 puede comunicarse con otros ordenadores o redes de ordenadores por medio de un adaptador de red capaz de comunicarse 5028 con una red 5029. Adaptadores de red de ejemplo son canales de comunicaciones, token ring, Ethernet o módems. Alternativamente, el sistema 5021 puede comunicarse usando una interfaz inalámbrica, tal como una tarjeta CDPD (datos de paquetes digitales celulares). El sistema 5021 se puede asociar con tales otros ordenadores en una Red de Área Local (LAN) o una Red de Área Extensa (WAN), o el sistema 5021 puede ser un cliente en una disposición cliente/servidor con otro ordenador, etc. Todas estas configuraciones, así como el hardware y software de comunicaciones apropiado, son conocidas en la técnica.

La FIG. 19 ilustra una red de procesamiento de datos 5040 en la que se pueden poner en práctica una o más realizaciones. La red de procesamiento de datos 5040 puede incluir una pluralidad de redes individuales, tales como una red inalámbrica y una red cableada, cada una de las cuales puede incluir una pluralidad de estaciones de trabajo individuales 5041, 5042, 5043, 5044. Además, como apreciarán los expertos en la técnica, se pueden incluir una o más LAN, donde una LAN puede comprender una pluralidad de estaciones de trabajo inteligentes acopladas a un procesador de ordenador central.

Todavía con referencia a la FIG. 19, las redes también pueden incluir grandes ordenadores o servidores, tales como un ordenador pasarela (cliente servidor 5046) o un servidor de aplicaciones (servidor remoto 5048 que puede acceder a un repositorio de datos y también al que se puede acceder directamente desde una estación de trabajo 5045). Un ordenador pasarela 5046 sirve como punto de entrada a cada red individual. Se necesita una pasarela cuando se conecta un protocolo de interconexión de redes con otro. La pasarela 5046 se puede acoplar preferiblemente a otra red (por ejemplo, Internet 5047) por medio de un enlace de comunicaciones. La pasarela 5046 también se puede acoplar directamente a una o más estaciones de trabajo 5041, 5042, 5043, 5044 usando un enlace de comunicaciones. El ordenador pasarela se puede implementar utilizando un servidor IBM eServer System z disponible en International Business Machines Corporation.

Con referencia concurrentemente a la FIG. 18 y la FIG. 19, se puede acceder al código de programación de software 5031, que puede incorporar uno o más aspectos, por el procesador 5026 del sistema 5020, desde medios de almacenamiento a largo plazo 5027, tales como una unidad de CD-ROM o un disco duro. El código de programación de software se puede incorporar en cualquiera de una variedad de medios conocidos para su uso con un sistema de procesamiento de datos, tal como un disquete, un disco duro o un CD-ROM. El código se puede distribuir en tales medios, o se puede distribuir a los usuarios 5050, 5051 desde la memoria o el almacenamiento de un sistema informático sobre una red a otros sistemas informáticos para su uso por los usuarios de tales otros sistemas.

Alternativamente, el código de programación se puede incorporar en la memoria 5025, y acceder por el procesador 5026 usando el bus de procesador. Tal código de programación incluye un sistema operativo que controla la función y la interacción de los diversos componentes del ordenador y uno o más programas de aplicaciones 5032. El código de programa normalmente se pagina del medio de almacenamiento 5027 a la memoria de alta velocidad 5025, donde está disponible para su procesamiento por el procesador 5026. Las técnicas y métodos para incorporar el código de programación de software en la memoria, en medios físicos y/o distribuir código software a través de redes son bien conocidos y no se tratarán aún más en la presente memoria. El código de programa, cuando se crea y almacena en un medio tangible (incluyendo, pero no limitado a, módulos de memoria electrónica (RAM), memoria rápida, Discos Compactos (CD), DVD, cinta magnética y similares, a menudo se conocen como “producto de programa de ordenador”. El medio de producto de programa de ordenador típicamente es legible por un circuito de procesamiento, preferiblemente en un sistema informático para su ejecución por el circuito de procesamiento.

La memoria caché que está más fácilmente disponible para el procesador (normalmente más rápida y más pequeña que otras memorias caché del procesador) es la memoria caché más baja (L1 o nivel uno) y el almacén principal (memoria principal) es la memoria caché de nivel más alto (L3 si hay 3 niveles). La memoria caché de nivel más bajo a menudo se divide en una memoria caché de instrucciones (Caché I) que contiene instrucciones de máquina a ser ejecutadas y una memoria caché de datos (Caché D) que contiene operandos de datos.

Con referencia a la FIG. 20, se representa una realización de procesador ejemplar para el procesador 5026. Típicamente, se emplean uno o más niveles de memoria caché 5053 para almacenar temporalmente bloques de memoria con el fin de mejorar el rendimiento del procesador. La memoria caché 5053 es un almacenador temporal de alta velocidad que contiene líneas de caché de datos de memoria que probablemente han de ser usadas. Las líneas de caché típicas son de 64, 128 o 256 bytes de datos de memoria. A menudo se emplean memorias caché separadas para almacenar en caché instrucciones distintas que para almacenar en caché datos. La coherencia de memoria caché (sincronización de copias de líneas en memoria y las memorias caché) a menudo se proporciona por diversos algoritmos de “fisgoneo” bien conocidos en la técnica. El almacenamiento de memoria principal 5025 de un sistema de procesador a menudo se conoce como memoria caché. En un sistema de procesador que tiene 4 niveles de memoria caché 5053, el almacenamiento principal 5025 algunas veces se conoce como memoria caché de nivel 5 (L5), dado que típicamente es más rápido y solamente contiene una parte del almacenamiento no volátil (DASD, cinta, etc.) que está disponible para un sistema informático. El almacenamiento principal 5025 “almacena en caché” páginas de datos paginados dentro y fuera del almacenamiento principal 5025 por el sistema operativo.

Un contador de programa (contador de instrucciones) 5061 hace un seguimiento de la dirección de la instrucción actual a ser ejecutada. Un contador de programa en un procesador de z/Architecture es de 64 bits y se puede

truncar a 31 o 24 bits para soportar los límites de direccionamiento anteriores. Un contador de programa se incorpora típicamente en una PSW (palabra de estado de programa) de un ordenador, de manera que persista durante la conmutación de contexto. De este modo, un programa en curso, que tiene un valor de contador de programa, se puede interrumpir, por ejemplo, por 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 programa mientras que el programa no está activo, y el contador de programa (en la PSW) del sistema operativo se usa mientras que el sistema operativo está ejecutándose. Típicamente, el contador de programa se aumenta en una cantidad igual al número de bytes de la instrucción actual. Las instrucciones de RISC (Ordenador con Conjunto de Instrucciones Reducido) típicamente son de longitud fija, mientras que las instrucciones CISC (Ordenador con Conjunto de Instrucciones Complejo) típicamente son 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 programa 5061 se modifica o bien por una operación de conmutación de contexto o bien por una operación tomada de ramificación de una instrucción de ramificación, 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 acerca del programa que se ejecuta (tal como códigos de condición), y se carga un nuevo valor de contador de programa que apunta a una instrucción de un nuevo módulo de programa a ser ejecutado. Se realiza una operación tomada de ramificación con el fin de permitir que el programa tome decisiones o realice un bucle dentro del programa cargando el resultado de la instrucción de ramificación en el contador de programa 5061.

Típicamente se emplea una unidad de búsqueda de instrucciones 5055 para obtener instrucciones en nombre del procesador 5026. La unidad de búsqueda o bien obtiene “las siguientes instrucciones secuenciales”, las instrucciones objetivo de las instrucciones tomadas de ramificación, o las primeras instrucciones de un programa que sigue a una conmutación de contexto. Las unidades de búsqueda de instrucciones modernas a menudo emplean técnicas de obtención previa para obtener previamente instrucciones de manera especulativa en base a la probabilidad de que se pudieran usar las instrucciones obtenidas previamente. Por ejemplo, una unidad de búsqueda puede obtener 16 bytes de instrucción que incluye la siguiente instrucción secuencial y bytes adicionales de instrucciones secuenciales adicionales.

Las instrucciones obtenidas se ejecutan entonces por el procesador 5026. En una realización, la instrucción o las instrucciones obtenidas se pasan a una unidad de despacho 5056 de la unidad de búsqueda. La unidad de despacho decodifica la instrucción o las instrucciones y envía información acerca de la instrucción o las instrucciones decodificadas a las unidades 5057, 5058, 5060 apropiadas. Una unidad de ejecución 5057 típicamente recibirá información acerca de las instrucciones aritméticas decodificadas de la unidad de búsqueda de instrucciones 5055 y realizará operaciones aritméticas sobre operandos según el código de operación de la instrucción. Los operandos se proporcionan a la unidad de ejecución 5057, preferiblemente o bien desde la memoria 5025, los registros de arquitectura 5059 o bien desde un campo inmediato de la instrucción que se ejecuta. Los resultados de la ejecución, cuando se almacenan, se almacenan o bien en la memoria 5025, los registros 5059 o bien en otro hardware de máquina (tal como los registros de control, los registros de PSW y similares).

Las direcciones virtuales se transforman en direcciones reales usando la traducción dinámica de direcciones 5062 y, opcionalmente, usando la traducción de registros de acceso 5063.

Un procesador 5026 típicamente tiene una o más unidades 5057, 5058, 5060 para ejecutar la función de la instrucción. Con referencia a la FIG. 21A, una unidad de ejecución 5057 puede comunicarse 5071 con los registros generales de arquitectura 5059, una unidad de decodificación/despacho 5056, una unidad de almacenamiento de carga 5060 y otras unidades de procesador 5065 por medio de la lógica de interfaz 5071. Una unidad de ejecución 5057 puede emplear varios circuitos de registro 5067, 5068, 5069 para contener información sobre la que operará la unidad aritmético lógica (ALU) 5066. La ALU realiza operaciones aritméticas tales como sumar, restar, multiplicar y dividir, así como funciones lógicas tales como AND, OR y OR exclusiva (XOR), rotar y desplazar. Preferiblemente, la ALU soporta operaciones especializadas que son dependientes del diseño. Otros circuitos pueden proporcionar otras facilidades de arquitectura 5072 incluyendo códigos de condición y lógica de soporte de recuperación, por ejemplo. Típicamente, el resultado de una operación ALU se mantiene en un circuito de registro de salida 5070 que puede enviar el resultado a una variedad de otras funciones de procesamiento. Hay muchas disposiciones de unidades de procesador, la presente descripción solamente se pretende que proporcione una comprensión representativa de una realización.

Una instrucción ADD, por ejemplo, se ejecutaría en una unidad de ejecución 5057 que tiene funcionalidad aritmética y lógica, mientras que una instrucción de punto flotante, por ejemplo, se ejecutaría en una ejecución de punto flotante que tiene capacidad de punto flotante especializada. Preferiblemente, una unidad de ejecución opera sobre operandos identificados por una instrucción realizando una función definida de código de operación sobre los operandos. Por ejemplo, una instrucción ADD se puede ejecutar por una unidad de ejecución 5057 sobre los operandos encontrados en dos registros 5059 identificados por los campos de registro de la instrucción.

La unidad de ejecución 5057 realiza la suma aritmética sobre dos operandos y almacena el resultado en un tercer operando donde el tercer operando puede ser un tercer registro o uno de los dos registros fuente. La unidad de ejecución utiliza preferiblemente una Unidad Aritmético Lógica (ALU) 5066 que es capaz de realizar una variedad de funciones lógicas tales como Desplazar, Rotar, AND, OR y XOR, así como una variedad de funciones algebraicas

incluyendo cualquiera de sumar, restar, multiplicar, dividir. Algunas ALU 5066 están diseñadas para operaciones escalares y otras para punto flotante. Los datos pueden ser Big Endian (donde el byte menos significativo está en la dirección de byte más alta) o Little Endian (donde el byte menos significativo está en la dirección de byte más baja) dependiendo de la arquitectura. La z/Architecture de IBM es Big Endian. Los campos de 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 el sentido que la ALU no necesita diseñar una capacidad de resta dado que o bien un valor negativo o bien un valor positivo en el complemento de 2 solamente requiere una adición dentro de la ALU. Los números se describen comúnmente en forma abreviada, donde un campo de 12 bits define una dirección de un bloque de 4096 bytes y se describe comúnmente como un bloque de 4 Kbytes (Kilobytes), por ejemplo.

Con referencia a la FIG. 21B, la información de instrucción de ramificación para ejecutar una instrucción de ramificación se envía típicamente a una unidad de ramificación 5058 que a menudo emplea un algoritmo de predicción de ramificación tal como una tabla de historial de ramificación 5082 para predecir el resultado de la ramificación antes de que se completen otras operaciones condicionales. El objetivo de la instrucción de ramificación actual se buscará y ejecutará especulativamente antes de que se completen las operaciones condicionales. Cuando se completan las operaciones condicionales, las instrucciones de ramificación ejecutadas especulativamente o bien se completan o bien se descartan en base a las condiciones de la operación condicional y al resultado especulado. Una instrucción de ramificación típica puede probar códigos de condición y ramificarse a una dirección objetivo si los códigos de condición cumplen con el requisito de ramificación de la instrucción de ramificación, una dirección objetivo se puede calcular en base a varios números, incluyendo los encontrados en los campos de registro o en un campo inmediato de la instrucción, por ejemplo. La unidad de ramificación 5058 puede emplear una ALU 5074 que tiene una pluralidad de circuitos de registro de entrada 5075, 5076, 5077 y un circuito de registro de salida 5080. La unidad de ramificación 5058 puede comunicarse 5081 con los registros generales 5059, la unidad de despacho de decodificación 5056 u otros circuitos 5073, por ejemplo.

La ejecución de un grupo de instrucciones se puede interrumpir por una variedad de razones, incluyendo una conmutación de contexto iniciada por un sistema operativo, una excepción o error de programa que causa una conmutación de contexto, una señal de interrupción de I/O que causa una conmutación de contexto o actividad de subprocesos múltiples de una pluralidad de programas (en un entorno de subprocesos múltiples), por ejemplo. Preferiblemente, una acción de conmutación de contexto guarda información de estado acerca de un programa que se ejecuta actualmente y entonces carga información de estado acerca de otro programa que se invoca. La información de estado se puede guardar en registros de hardware o en la memoria, por ejemplo. La información de estado comprende preferiblemente un valor de contador de programa que apunta a una siguiente instrucción a ser ejecutada, códigos de condición, información de traducción de memoria y contenido de registro de arquitectura. Una actividad de conmutación de contexto se puede ejercitar por circuitos de hardware, programas de aplicaciones, programas de sistema operativo o código de microprograma (microcódigo, picocódigo o código interno bajo licencia (LIC)) solo o en combinación.

Un procesador accede a los operandos según métodos definidos por instrucción. La instrucción puede proporcionar un operando inmediato que usa el valor de una parte de la instrucción, puede proporcionar uno o más campos de registro que apuntan explícitamente o bien a registros de propósito general o bien a registros de propósito especial (registros de punto flotante, por ejemplo). La instrucción puede utilizar registros implícitos identificados por un campo de código de operación como operandos. La instrucción puede utilizar ubicaciones de memoria para los operandos. Una ubicación de memoria de un operando se puede proporcionar por un registro, un campo inmediato, o una combinación de registros y campo inmediato, como se ejemplifica por la facilidad de desplazamiento larga de 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 proporcionar la dirección del operando en la memoria, por ejemplo. Una ubicación en la presente memoria típicamente implica una ubicación en la memoria principal (almacenamiento principal) a menos que se indique de otro modo.

Con referencia a la FIG. 21C, un procesador accede al almacenamiento usando una unidad de carga/almacenamiento 5060. La unidad de carga/almacenamiento 5060 puede realizar una operación de carga obteniendo la dirección del operando objetivo en la memoria 5053 y cargando el operando en un registro 5059 u otra ubicación de memoria 5053, o puede realizar una operación de almacén obteniendo la dirección del operando objetivo en la memoria 5053 y almacenando los 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 en relación con la secuencia de instrucciones, no obstante, la unidad de carga/almacenamiento 5060 ha de mantener la apariencia para programas en que las instrucciones se ejecutaron en orden. Una unidad de carga/almacenamiento 5060 puede comunicarse 5084 con los registros generales 5059, la unidad de decodificación/despacho 5056, la interfaz de memoria caché/memoria 5053 u otros elementos 5083 y comprende diversos circuitos de registro 5086, 5087, 5088 y 5089, las ALU 5085 y la lógica de control 5090 para calcular direcciones de almacenamiento y proporcionar una secuencia entubada para mantener las operaciones en orden. Algunas operaciones pueden estar fuera de orden, pero la unidad de carga/almacenamiento proporciona funcionalidad para hacer que las operaciones fuera de orden le parezcan al programa como si se hubieran realizado en orden, como es bien conocido en la técnica.

Preferiblemente, las direcciones que un programa de aplicaciones “ve” a menudo se conocen como direcciones virtuales. Las direcciones virtuales algunas veces se conocen como “direcciones lógicas” y “direcciones eficaces”. Estas direcciones virtuales son virtuales en el sentido que se redirigen a una ubicación de memoria física mediante una de una variedad de tecnologías de traducción dinámica de direcciones (DAT) incluyendo, pero no limitadas a, simplemente prefijar una dirección virtual con un valor de desplazamiento, traducir la dirección virtual a través de una o más tablas de traducción, las tablas de traducción que comprenden preferiblemente al menos una tabla de segmentos y una tabla de páginas sola o en combinación, preferiblemente, la tabla de segmentos que tiene una entrada que apunta a la tabla de páginas. En la z/Architecture, se proporciona una jerarquía de traducción que incluye una región de primera tabla, una región de segunda tabla, una región de tercera tabla, una tabla de segmentos y una tabla de páginas opcional. El rendimiento de la traducción de direcciones a menudo se mejora utilizando un almacenador temporal de traducción anticipada (TLB) que comprende entradas que mapean una dirección virtual a una ubicación de memoria física asociada. Las entradas se crean cuando la DAT traduce una dirección virtual usando las tablas de traducción. El uso posterior de la dirección virtual puede utilizar entonces la entrada del TLB rápido en lugar de los accesos de tabla de traducción secuencial lenta. El contenido del TLB se puede gestionar por una variedad de algoritmos de reemplazo, incluyendo LRU (Usado Menos Recientemente).

En el caso donde el procesador es un procesador de un sistema multiprocesador, cada procesador tiene la responsabilidad de mantener los recursos compartidos, tales como I/O, memorias caché, TLB y memoria, entrelazados por coherencia. Típicamente, las tecnologías de “fisgoneo” se utilizarán en el mantenimiento de la coherencia de la memoria caché. En un entorno de fisgoneo, cada línea de caché se puede marcar como que está en uno cualquiera de un estado compartido, un estado exclusivo, un estado cambiado, un estado no válido y similares con el fin de facilitar el intercambio.

Las unidades de I/O 5054 (FIG. 20) dotan al procesador con medios para unirse a dispositivos periféricos incluyendo cinta, discos, impresoras, visualizadores y redes, por ejemplo. Las unidades de I/O a menudo se presentan al programa de ordenador mediante controladores de software. En grandes ordenadores, tales como el System z de IBM®, los adaptadores de canal y los adaptadores de sistema abierto son unidades de I/O del gran ordenador que proporcionan las comunicaciones entre el sistema operativo y los dispositivos periféricos.

Además, otros tipos de entornos informáticos pueden beneficiarse de uno o más aspectos. Como ejemplo, un entorno puede incluir un emulador (por ejemplo, software u otros mecanismos de emulación), en el que una arquitectura particular (incluyendo, por ejemplo, ejecución de instrucciones, funciones de arquitectura, tales como la traducción de direcciones, y registros de arquitectura) o un subconjunto de la misma se emula (por ejemplo, en un sistema informático nativo que tiene un procesador y una memoria). En tal entorno, una o más funciones de emulación del emulador pueden implementar una o más realizaciones, aún cuando un ordenador que ejecuta el emulador pueda tener una arquitectura diferente de las capacidades que se emulan. Como ejemplo, en el modo de emulación, se decodifica la instrucción u operación específica que se emula, y se construye una función de emulación apropiada para implementar la instrucción u operación individual.

En un entorno de emulación, un ordenador central incluye, por ejemplo, una memoria para almacenar instrucciones y datos; una unidad de búsqueda de instrucciones para obtener instrucciones de la memoria y, opcionalmente, para proporcionar almacenamiento temporal local para la instrucción de búsqueda; una unidad de decodificación de instrucciones para recibir las instrucciones de búsqueda y para determinar el tipo de instrucciones que se han obtenido; y una unidad de ejecución de instrucciones para ejecutar las instrucciones. La ejecución puede incluir cargar datos en un registro desde la memoria; almacenar datos de vuelta en la memoria desde un registro; o realizar algún tipo de operación aritmética o lógica, como se determina por la unidad de decodificación. En un ejemplo, cada unidad se implementa en software. Por ejemplo, las operaciones que se realizan por las unidades se implementan como una o más subrutinas dentro del software de emulador.

Más particularmente, en un gran ordenador, las instrucciones de máquina de arquitectura se usan por programadores, normalmente hoy en día programadores en “C”, a menudo por medio de una aplicación de compilador. Estas instrucciones almacenadas en el medio de almacenamiento se pueden ejecutar de manera nativa en un Servidor de IBM® de z/Architecture, o alternativamente en máquinas que ejecutan otras arquitecturas. Se pueden emular en los servidores de grandes ordenadores de IBM® existentes y futuros y en otras máquinas de IBM® (por ejemplo, servidores Power Systems y Servidores System x). Se pueden ejecutar en máquinas que ejecutan Linux en una amplia variedad de máquinas que usan hardware fabricado por IBM®, Intel®, AMD y otros. Además de la ejecución en ese hardware bajo una z/Architecture, se puede usar Linux, así como máquinas que usan emulación de Hercules, UMX o FSI (Fundamental Software, Inc.), donde generalmente la ejecución está 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 típicamente ejecuta un software de emulación que comprende o bien microprograma o bien un sistema operativo nativo para realizar la emulación del procesador emulado. El software de emulación es responsable de obtener y ejecutar las instrucciones de la arquitectura del procesador emulado. El software de emulación mantiene un contador del programa emulado para hacer un seguimiento de los límites de instrucción. El software de emulación puede obtener una o más instrucciones de máquina emuladas a la vez y convertir una o más instrucciones de máquina emuladas en un grupo correspondiente de instrucciones de máquina nativas para su

ejecución por el procesador nativo. Estas instrucciones convertidas se pueden almacenar en caché de manera que se puede lograr una conversión más rápida. Sin embargo, el software de emulación ha de mantener las reglas de arquitectura de la arquitectura del procesador emulado para asegurar que los sistemas operativos y las aplicaciones escritas para el procesador emulado operen correctamente. Además, el software de emulación ha de proporcionar recursos identificados por la arquitectura del procesador emulado incluyendo, pero no limitados a, registros de control, registros de propósito general, registros de punto flotante, función de traducción dinámica de direcciones incluyendo tablas de segmentos y tablas de páginas por ejemplo, mecanismos de interrupción, mecanismos de conmutación de contexto, relojes de Hora del Día (TOD) e interfaces de arquitectura para subsistemas de I/O, de manera que un sistema operativo o un programa de aplicaciones diseñado para ejecutarse en el procesador emulado, se pueda ejecutar en el procesador nativo que tiene el software de emulación.

Se decodifica una instrucción específica que se emula, y se llama a una subrutina para realizar la función de la instrucción individual. Se implementa una función de software de emulación que emula una función de un procesador emulado, por ejemplo, en una subrutina o controlador "C", o algún otro método para proporcionar un controlador para el hardware específico como estará dentro de la habilidad de los expertos en la técnica después de entender la descripción de la realización preferida. Diversas patentes de emulación de software y hardware incluyendo, pero no limitadas a, la Patente de EE.UU. Nº 5551013, titulada "Multiprocessor for Hardware Emulation", de Beausoleil et al.; y la patente de EE.UU. Nº 6009261, titulada "Preprocessing of Stored Target Routines for Emulating Incompatible Instructions on a Target Processor", de Scalzi et al; y la Patente de EE.UU. Nº 5574873, titulada "Decoding Guest Instruction to Directly Access Emulation Routines that Emulate the Guest Instructions", de Davidian et al.; y la Patente de EE.UU. Nº 6308255, titulada "Symmetrical Multiprocessing Bus and Chipset Used for Coprocessor Support Allowing Non-Native Code to Run in a System", de Gorishek et al.; y la Patente de EE.UU. Nº 6463582, titulada "Dynamic Optimizing Object Code Translator for Architecture Emulation and Dynamic Optimizing Object Code Translation Method", de Lethin et al; y la Patente de EE.UU. Nº 5790825, titulada "Method for Emulating Guest Instructions on a Host Computer Through Dynamic Recompilation of Host Instructions", de Eric Traut; y muchas otras, ilustran una variedad de formas conocidas para lograr la emulación de un formato de instrucción diseñado para una máquina diferente para una máquina objetivo disponible para los expertos en la técnica.

En la FIG. 22, se proporciona un ejemplo de un sistema de ordenador central emulado 5092 que emula un sistema de ordenador central 5000' de una arquitectura de ordenador central. En el sistema de ordenador central emulado 5092, el procesador de ordenador central (CPU) 5091 es un procesador de ordenador central emulado (o procesador de ordenador central 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 central 5000'. El sistema de ordenador central emulado 5092 tiene una memoria 5094 accesible por el procesador de emulación 5093. En la realización de ejemplo, la memoria 5094 se divide en una parte de memoria de ordenador central 5096 y una parte de rutinas de emulación 5097. La memoria de ordenador central 5096 está disponible para los programas del ordenador central emulado 5092 según la arquitectura del ordenador central. El procesador de emulación 5093 ejecuta instrucciones nativas de un conjunto de instrucciones de arquitectura de una arquitectura distinta 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 ordenador central para ejecución de un programa en la memoria de ordenador central 5096 empleando una o más instrucciones obtenidas en una secuencia y rutina de acceso/decodificación que puede decodificar la instrucción o las instrucciones del ordenador central a las que se accede para determinar una rutina de ejecución de instrucciones nativas para emular la función de la instrucción de ordenador central a la que se accede. Otras facilidades que se definen para la arquitectura del sistema de ordenador central 5000' se pueden emular por las rutinas de facilidades de arquitectura, incluyendo tales facilidades como registros de propósito general, registros de control, traducción dinámica de direcciones y soporte del subsistema de I/O y memoria caché del procesador, por ejemplo. Las rutinas de emulación también pueden aprovecharse de las funciones disponibles en el procesador de emulación 5093 (tales como registros generales y traducción dinámica de direcciones virtuales) para mejorar el rendimiento de las rutinas de emulación. También se pueden proporcionar hardware especial y motores sin carga para ayudar al procesador 5093 en la emulación de la función del ordenador central 5000'.

La terminología usada en la presente memoria es con el propósito de describir las realizaciones particulares solamente y no se pretende que sea limitante. Como se usa en la presente memoria, las formas singulares "un", "uno", "una", "el" y "la" se pretende que incluyan las formas en plural también, a menos que el contexto lo indique claramente de otro modo. Se entenderá además que los términos "comprende" y/o "que comprende", cuando se usan en esta especificación, especifican la presencia de las características, enteros, pasos, operaciones, elementos y/o componentes expuestos, pero no excluyen la presencia o adición de una o más de otras características, enteros, pasos, operaciones, elementos, componentes y/o grupos de los mismos.

Las estructuras, materiales, actos y equivalentes correspondientes de todos los medios o elementos de función más paso en las reivindicaciones a continuación, en su caso, se pretende que incluyan cualquier estructura, material o acto para realizar la función en combinación con otros elementos reivindicados, como se reivindica específicamente. La descripción de una o más realizaciones se ha presentado con propósitos de ilustración y descripción, pero no se pretende que sea exhaustiva o esté limitada a la forma descrita. Muchas modificaciones y variaciones serán evidentes para los expertos en la técnica. La realización se eligió y describió con el fin de explicar mejor diversos aspectos y la aplicación práctica, y para permitir que otros expertos en la técnica entiendan diversas realizaciones con diversas modificaciones que son adecuadas para el uso particular contemplado.

REIVINDICACIONES

1. Un método de control de ejecución de instrucciones dentro de transacciones en un procesador; en donde el procesador comprende

una unidad de ejecución de punto flotante para ejecutar instrucciones de punto flotante;

un registro de acceso que incluye una especificación indirecta de un elemento de control de espacio de direcciones que designa una tabla de traducción para un espacio de direcciones específico a ser usado en traducción de direcciones;

en donde una transacción incluye una secuencia de instrucciones a ser completada como una única unidad atómica o a ser abortada;

dicho método que comprende

ejecutar, por el procesador, una instrucción de máquina de comienzo de transacción (200), la ejecución que comprende iniciar una transacción;

caracterizado por

ejecutar la instrucción de máquina de comienzo de transacción comprende usar un primer control de al menos un control especificado en la instrucción de máquina de comienzo de transacción para determinar un primer valor, en donde el primer control es uno de

un control de modificación de registro de acceso permitido (212) usado para indicar si la transacción se permite para ejecutar un primer tipo de instrucciones que modifican el registro de acceso; o

un control de operación de punto flotante permitida (214) usado para indicar si la transacción se permite para ejecutar un primer tipo de instrucciones que son instrucciones de punto flotante especificadas;

determinar, en base al primer valor, si se permite que se ejecute una instrucción de la secuencia de instrucciones de la transacción que es del primer tipo; y

ejecutar la instrucción del primer tipo, en base al primer valor que indica que se permite que se ejecute la instrucción.

2. El método de la reivindicación 1, en donde la instrucción de máquina de comienzo de transacción es parte de un nido de instrucciones de máquina de comienzo de transacción (600, 602, 604), el nido de instrucciones de máquina de comienzo de transacción que tiene una pluralidad de niveles de anidamiento de transacciones, y la instrucción de máquina de comienzo de transacción que está en un nivel de anidamiento de transacciones actual, y en donde el primer control es el control de modificación de registro de acceso permitido, y el uso comprende realizar un AND lógico del control de modificación de registro de acceso permitido de la instrucción de máquina de comienzo de transacción para el nivel de anidamiento de transacciones actual y niveles de anidamiento externos, en su caso, para determinar el primer valor, el primer valor que comprende un control de modificación de registro de acceso permitido eficaz que indica si la transacción se permite que ejecute una instrucción que modifica un registro de acceso.

3. El método de la reivindicación 1, en donde la instrucción de máquina de comienzo de transacción es parte de un nido de instrucciones de máquina de comienzo de transacción (600, 602, 604), el nido de instrucciones de máquina de comienzo de transacción que tiene una pluralidad de niveles de anidamiento de transacciones, y la instrucción de máquina de comienzo de transacción que está en un nivel de anidamiento de transacciones actual, y en donde el primer control es el control de operación de punto flotante permitida, y el uso comprende realizar un AND lógico del control de operación de punto flotante permitida de la instrucción de máquina de comienzo de transacción para el nivel de anidamiento de transacciones actual y niveles de anidamiento externos, en su caso, para determinar el primer valor, el primer valor que comprende un control de operación de punto flotante permitida eficaz que indica si la transacción se permite que ejecute las instrucciones de punto flotante especificadas.

4. El método de la reivindicación 1, en donde la transacción es parte de un nido de transacciones, y en donde el método comprende además volver a determinar el primer valor en base a la terminación de una transacción del nido de transacciones.

5. Un sistema que comprende medios adaptados para llevar a cabo todos los pasos del método según cualquier reivindicación del método precedente.

6. Un programa de ordenador que comprende instrucciones para llevar a cabo todos los pasos del método según cualquier reivindicación del método precedente, cuando dicho programa de ordenador se ejecuta en un sistema informático.

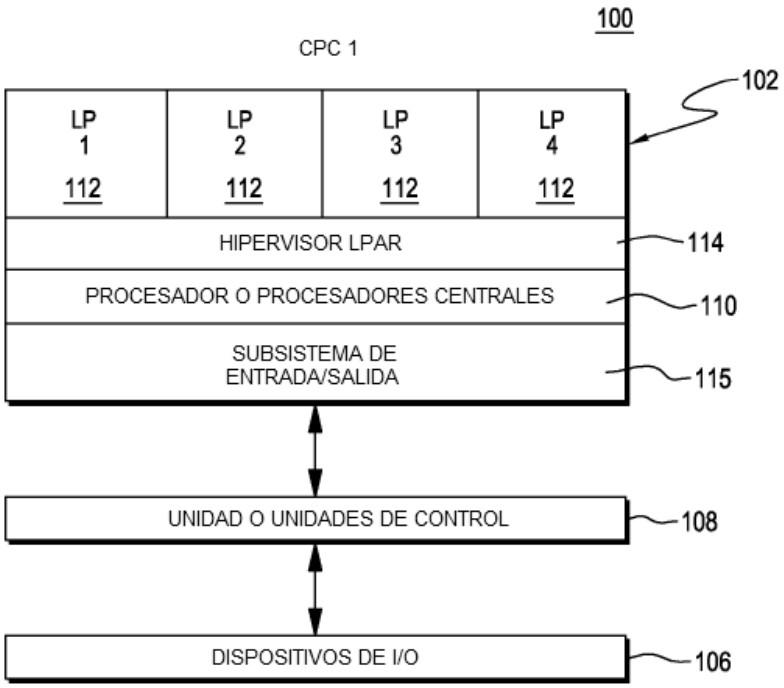
100
CPC 1
102
LP 1 112
LP 2 112
LP 3 112
LP 4 112
HIPERVISOR LPAR
114
PROCESADOR O PROCESADORES CENTRALES
110
SUBSISTEMA DE ENTRADA/SALIDA
115
UNIDAD O UNIDADES DE CONTROL
108
DISPOSITIVOS DE I/O
106

FIG. 1

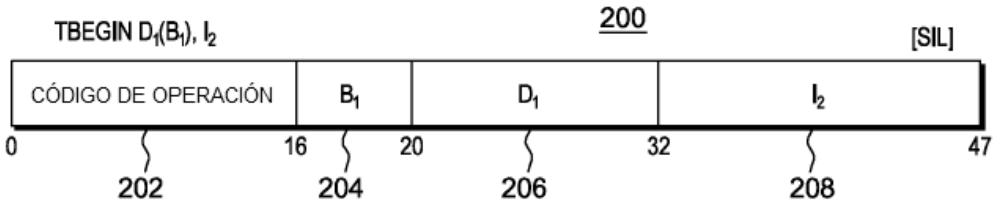

FIG. 2A

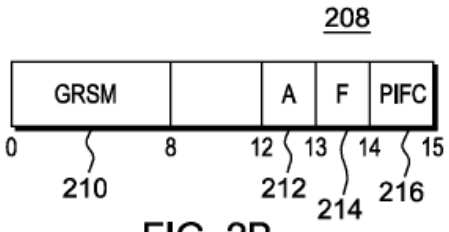

FIG. 2B

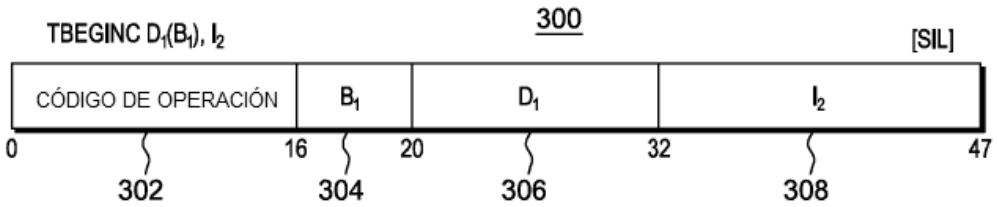

FIG. 3A

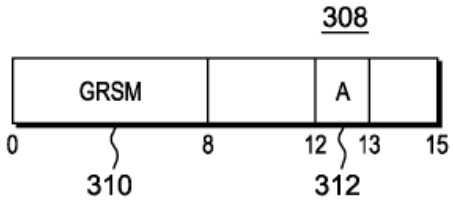

FIG. 3B

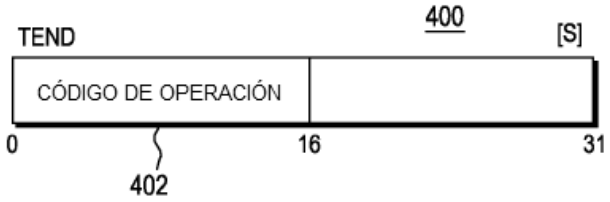
400
TEND
[S]
CÓDIGO DE OPERACIÓN
0
16
31
402

FIG. 4

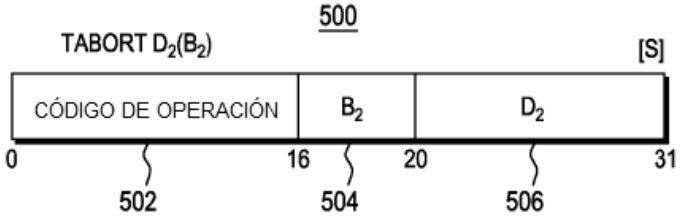
500
TABORT D2(B2)
[S]
CÓDIGO DE OPERACIÓN
B2
D2
0
16
20
31
502
504
506

FIG. 5

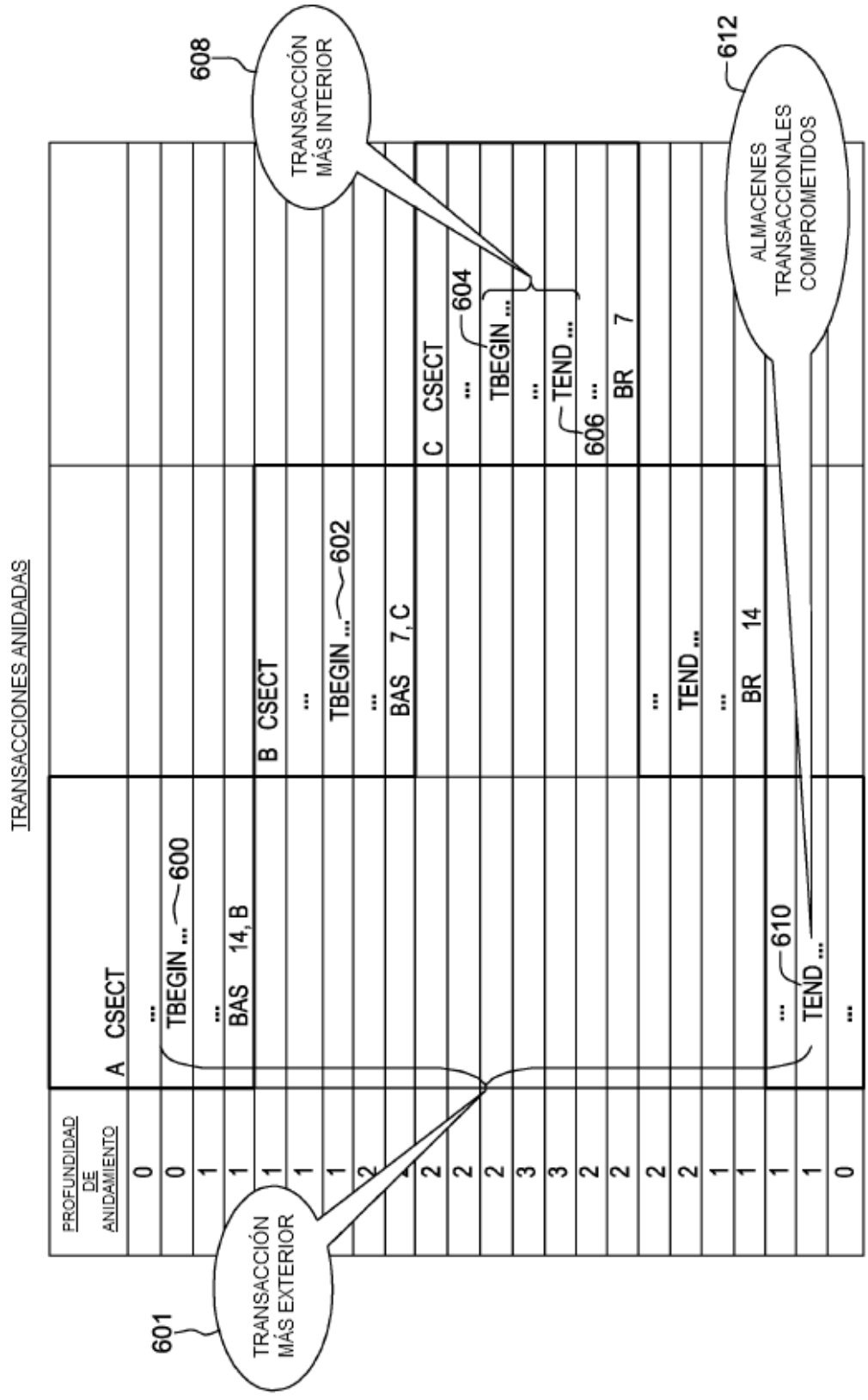
TRANSACCIONES ANIDADAS
PROFUNDIDAD DE ANIDAMIENTO
A CSECT
TBEGIN ...
600
BAS 14, B
B CSECT
TBEGIN ...
602
BAS 7, C
C CSECT
TBEGIN ...
604
TEND ...
606
BR 7
TEND ...
BR 14
TEND ...
610
TRANSACCIÓN MÁS EXTERIOR
601
TRANSACCIÓN MÁS INTERIOR
608
ALMACENES TRANSACCIONALES COMPROMETIDOS
612

FIG. 6

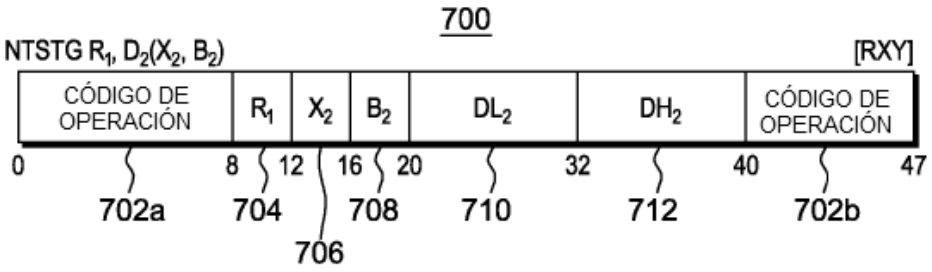
700
NTSTG R1, D2(X2, B2)
[RXY]
CÓDIGO DE OPERACIÓN
R1
X2
B2
DL2
DH2
CÓDIGO DE OPERACIÓN
0
8
12
16
20
32
40
47
702a
704
706
708
710
712
702b

FIG. 7

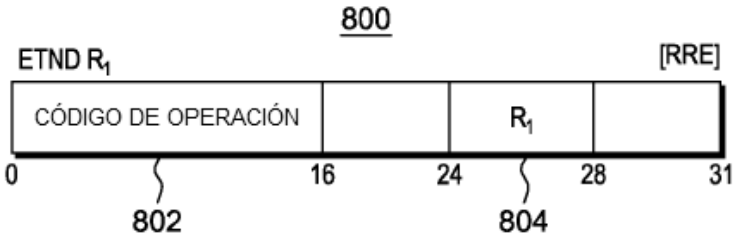
800
ETND R1
[RRE]
CÓDIGO DE OPERACIÓN
R1
0
16
24
28
31
802
804

FIG. 8

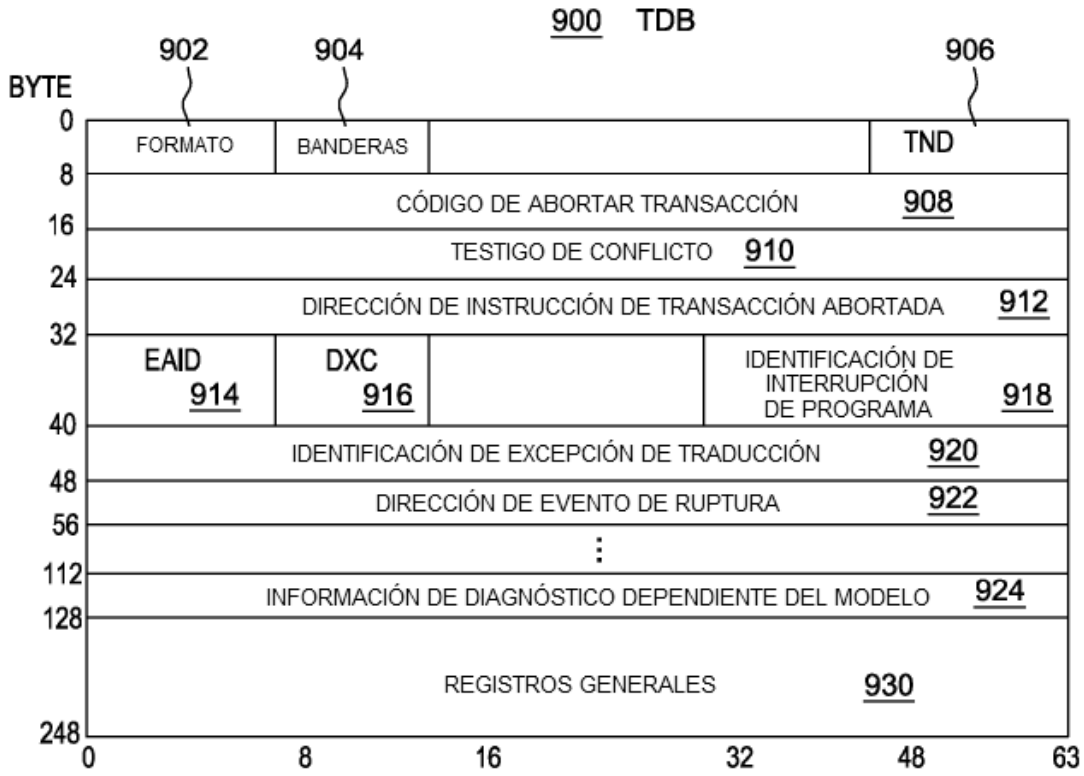
900 TDB
902
904
906
BYTE
0
8
16
24
32
40
48
56
112
128
248
FORMATO
BANDERAS
TND
CÓDIGO DE ABORTAR TRANSACCIÓN 908
TESTIGO DE CONFLICTO 910
DIRECCIÓN DE INSTRUCCIÓN DE TRANSACCIÓN ABORTADA 912
EAID 914
DXC 916
IDENTIFICACIÓN DE INTERRUPCIÓN DE PROGRAMA 918
IDENTIFICACIÓN DE EXCEPCIÓN DE TRADUCCIÓN 920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

FIG. 9

CÓDIGO	RAZÓN PARA ABORTAR	CC ESTABLECIDO
2	INTERRUPCIÓN EXTERNA	2
4	INTERRUPCIÓN DE PROGRAMA (NO FILTRADA)	2 o 3+
5	INTERRUPCIÓN DE COMPROBACIÓN DE MÁQUINA	2
6	INTERRUPCIÓN DE I/O	2
7	DESBORDAMIENTO DE OBTENCIÓN	2 o 3
8	DESBORDAMIENTO DE ALMACÉN	2 o 3
9	CONFLICTO DE OBTENCIÓN	2
10	CONFLICTO DE ALMACÉN	2
11	INSTRUCCIÓN RESTRINGIDA	3
12	CONDICIÓN DE EXCEPCIÓN DE PROGRAMA (FILTRADA)	3
13	PROFUNDIDAD DE ANIDAMIENTO EXCEDIDA	3
14	OBTENCIÓN DE MEMORIA CACHÉ RELACIONADA	2 o 3
15	ALMACÉN DE MEMORIA CACHÉ RELACIONADA	2 o 3
16	OTRA MEMORIA CACHÉ	2 o 3
255	CONDICIÓN MISCELÁNEA	2 o 3
>255	INSTRUCCIÓN TABORT	2 o 3
‡	NO SE PUEDE DETERMINAR; NINGÚN TDB ALMACENADO	1

EXPLICACIÓN:

\+ CÓDIGO DE CONDICIÓN SE BASA EN CÓDIGO DE INTERRUPCIÓN

‡ ESTA SITUACIÓN OCURRE CUANDO UNA TRANSACCIÓN SE ABORTA. PERO EL TDB HA LLEGADO A ESTAR INACCESIBLE POSTERIOR A LA EJECUCIÓN CON ÉXITO DE LA INSTRUCCIÓN TBEGIN MÁS EXTERNA. NINGÚN TDB ESPECIFICADO POR TBEGIN SE ALMACENA, Y LA CONDICIÓN DE CÓDIGO SE ESTABLECE EN 1.

FIG. 10

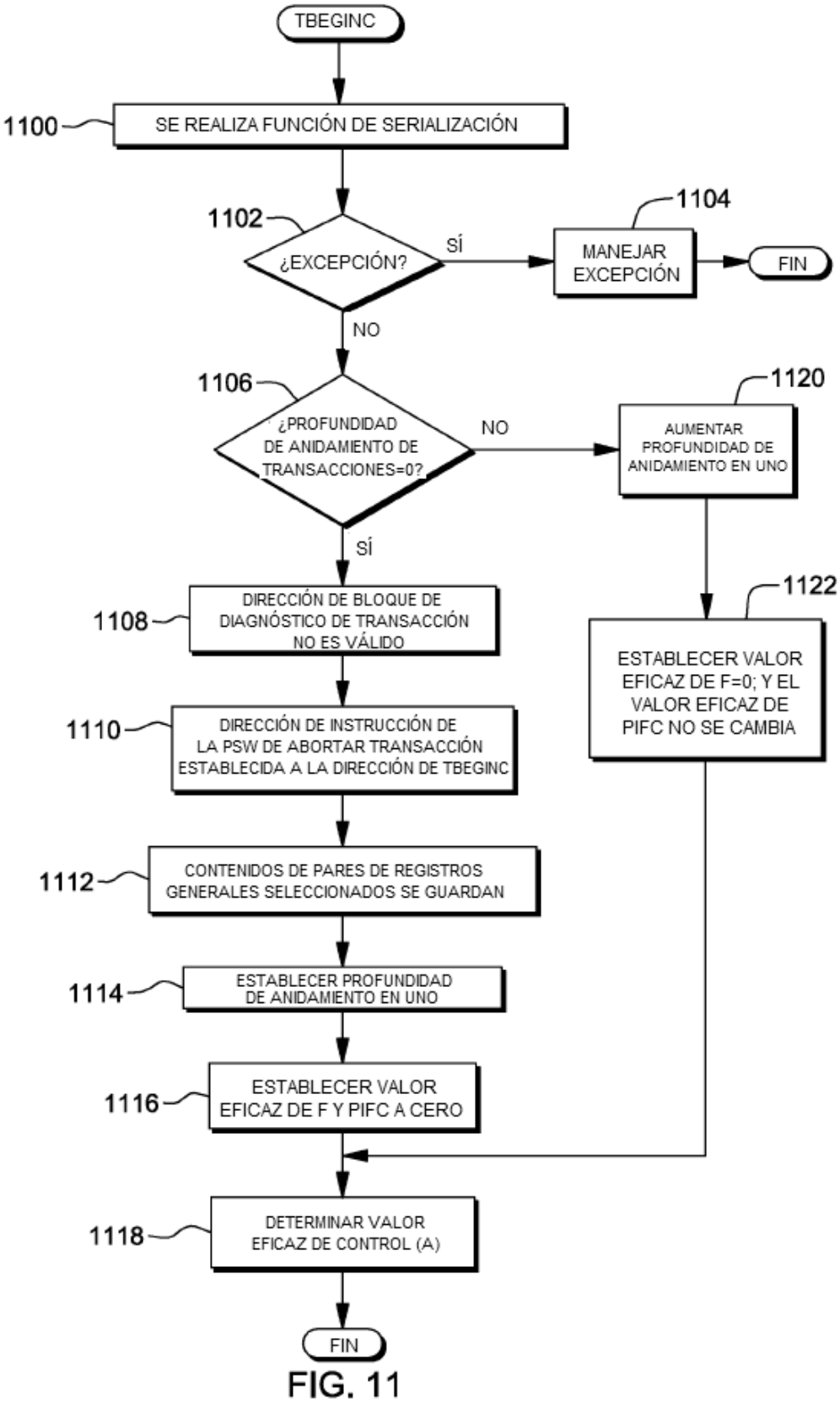
TBEGINC
1100
SE REALIZA FUNCIÓN DE SERIALIZACIÓN
1102
¿EXCEPCIÓN?
SÍ
1104
MANEJAR EXCEPCIÓN
FIN
NO
1106
¿PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIONES=0?
NO
1120
AUMENTAR PROFUNDIDAD DE ANIDAMIENTO EN UNO
SÍ
1108
DIRECCIÓN DE BLOQUE DE DIAGNÓSTICO DE TRANSACCIÓN NO ES VÁLIDO
1122
ESTABLECER VALOR EFICAZ DE F=0; Y EL VALOR EFICAZ DE PIFC NO SE CAMBIA
1110
DIRECCIÓN DE INSTRUCCIÓN DE LA PSW DE ABORTAR TRANSACCIÓN ESTABLECIDA A LA DIRECCIÓN DE TBEGINC
1112
CONTENIDOS DE PARES DE REGISTROS GENERALES SELECCIONADOS SE GUARDAN
1114
ESTABLECER PROFUNDIDAD DE ANIDAMIENTO EN UNO
1116
ESTABLECER VALOR EFICAZ DE F Y PIFC A CERO
1118
DETERMINAR VALOR EFICAZ DE CONTROL (A)
FIN

FIG. 11

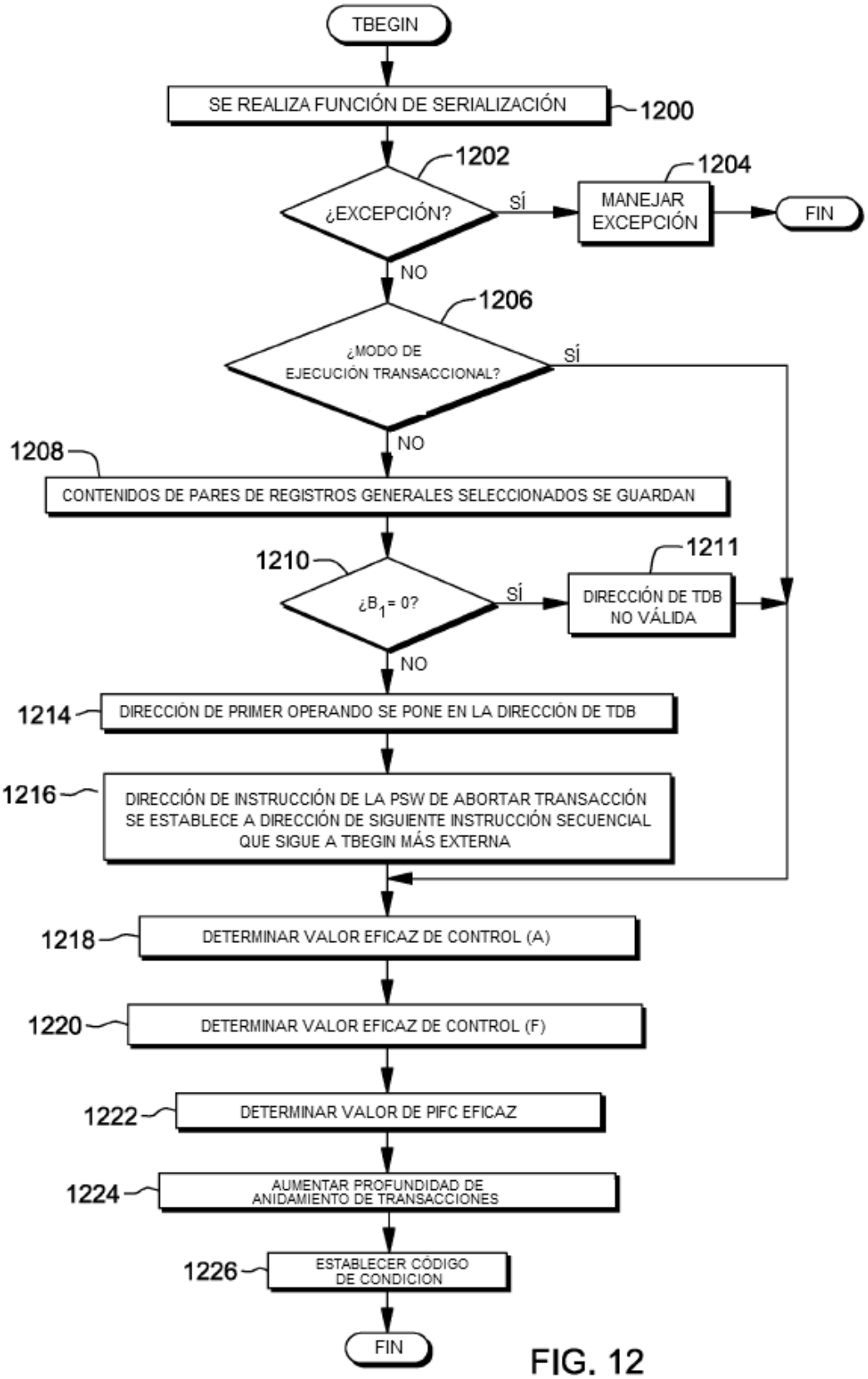
TBEGIN
SE REALIZA FUNCIÓN DE SERIALIZACIÓN
1200
1202
¿EXCEPCIÓN?
SÍ
1204
MANEJAR EXCEPCIÓN
FIN
NO
1206
¿MODO DE EJECUCIÓN TRANSACCIONAL?
SÍ
NO
1208
CONTENIDOS DE PARES DE REGISTROS GENERALES SELECCIONADOS SE GUARDAN
1210
¿B1 = 0?
SÍ
1211
DIRECCIÓN DE TDB NO VÁLIDA
NO
1214
DIRECCIÓN DE PRIMER OPERANDO SE PONE EN LA DIRECCIÓN DE TDB
1216
DIRECCIÓN DE INSTRUCCIÓN DE LA PSW DE ABORTAR TRANSACCIÓN SE ESTABLECE A DIRECCIÓN DE SIGUIENTE INSTRUCCIÓN SECUENCIAL QUE SIGUE A TBEGIN MÁS EXTERNA
1218
DETERMINAR VALOR EFICAZ DE CONTROL (A)
1220
DETERMINAR VALOR EFICAZ DE CONTROL (F)
1222
DETERMINAR VALOR DE PIFC EFICAZ
1224
AUMENTAR PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIONES
1226
ESTABLECER CÓDIGO DE CONDICION
FIN

FIG. 12

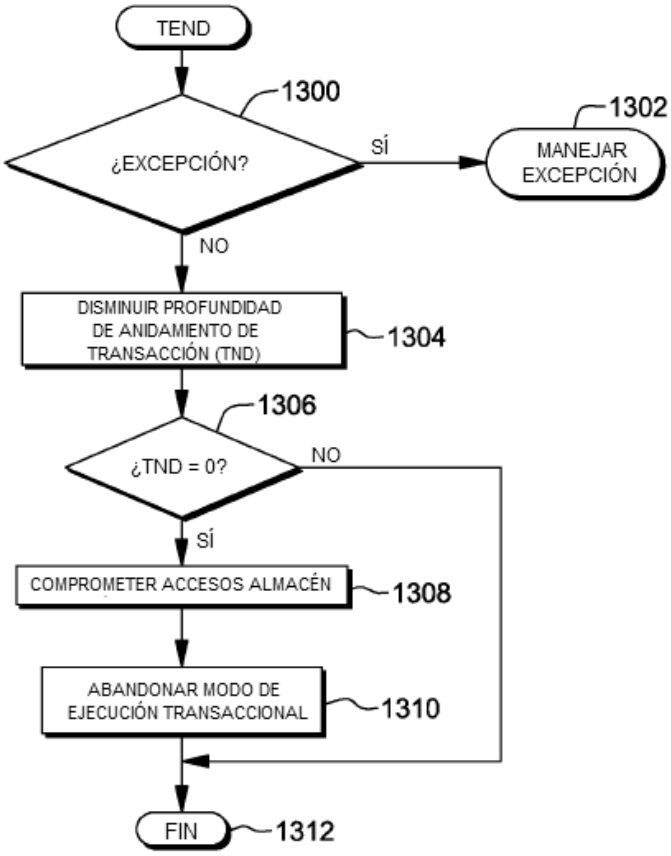
TEND
1300
¿EXCEPCIÓN?
SÍ
1302
MANEJAR EXCEPCIÓN
NO
DISMINUIR PROFUNDIDAD DE ANIDAMIENTO DE TRANSACCIÓN (TND)
1304
1306
¿TND = 0?
NO
SÍ
COMPROMETER ACCESOS ALMACÉN
1308
ABANDONAR MODO DE EJECUCIÓN TRANSACCIONAL
1310
FIN
1312

FIG. 13

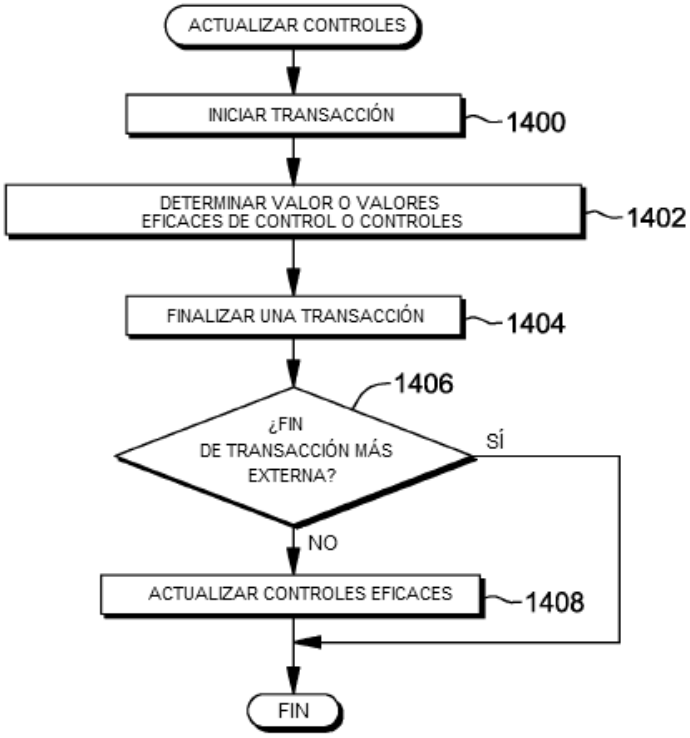
ACTUALIZAR CONTROLES
INICIAR TRANSACCIÓN
1400
DETERMINAR VALOR O VALORES EFICACES DE CONTROL O CONTROLES
1402
FINALIZAR UNA TRANSACCIÓN
1404
1406
¿FIN DE TRANSACCIÓN MÁS EXTERNA?
SÍ
NO
ACTUALIZAR CONTROLES EFICACES
1408
FIN

FIG. 14

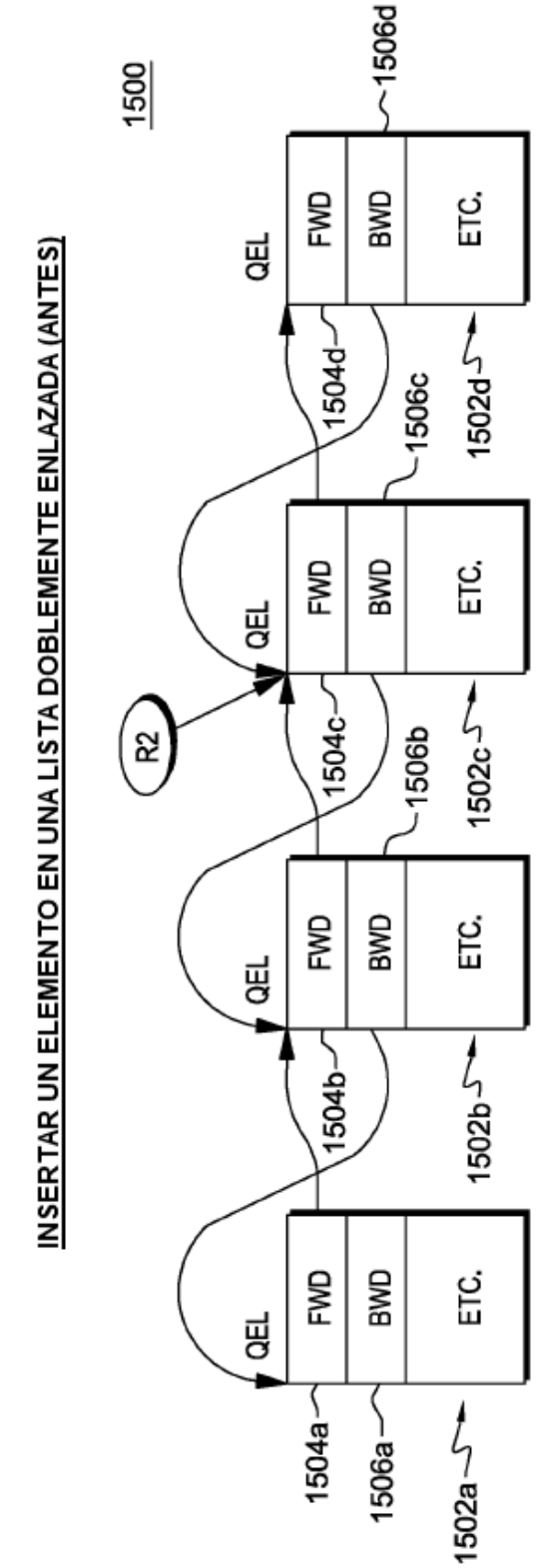
INSERTAR UN ELEMENTO EN UNA LISTA DOBLEMENTE ENLAZADA (ANTES)
1500
QEL
FWD
BWD
ETC.
1504a
1506a
1502a
1504b
1506b
1502b
1504c
1506c
1502c
1504d
1506d
1502d
R2
1504e
1506e
1502e
R1

FIG. 15A

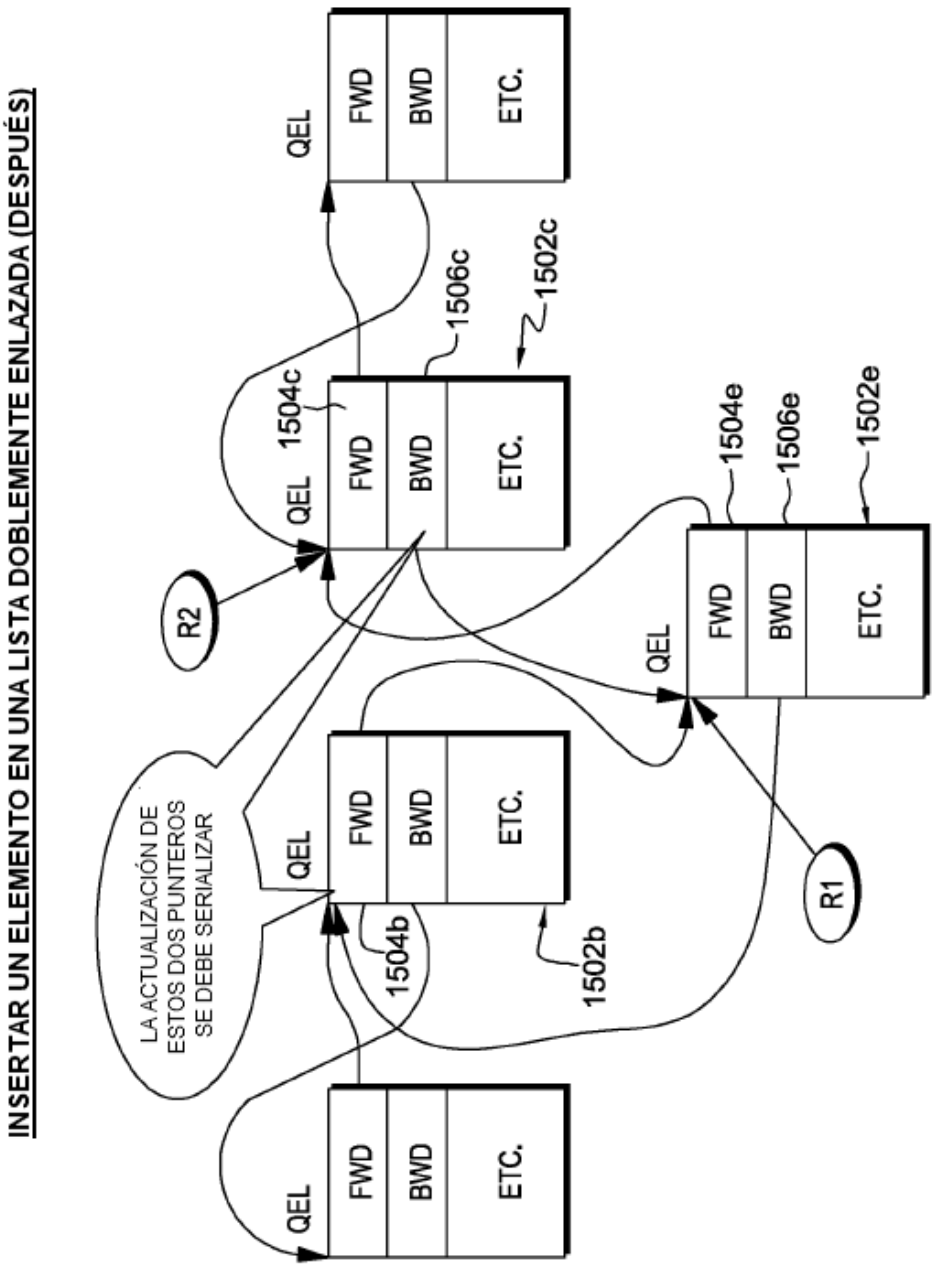
INSERTAR UN ELEMENTO EN UNA LISTA DOBLEMENTE ENLAZADA (DESPUÉS)
QEL
FWD
BWD
ETC.
1504c
1506c
1502c
R2
LA ACTUALIZACIÓN DE ESTOS DOS PUNTEROS SE DEBE SERIALIZAR
1504b
1502b
1504e
1506e
1502e
R1

FIG. 15B

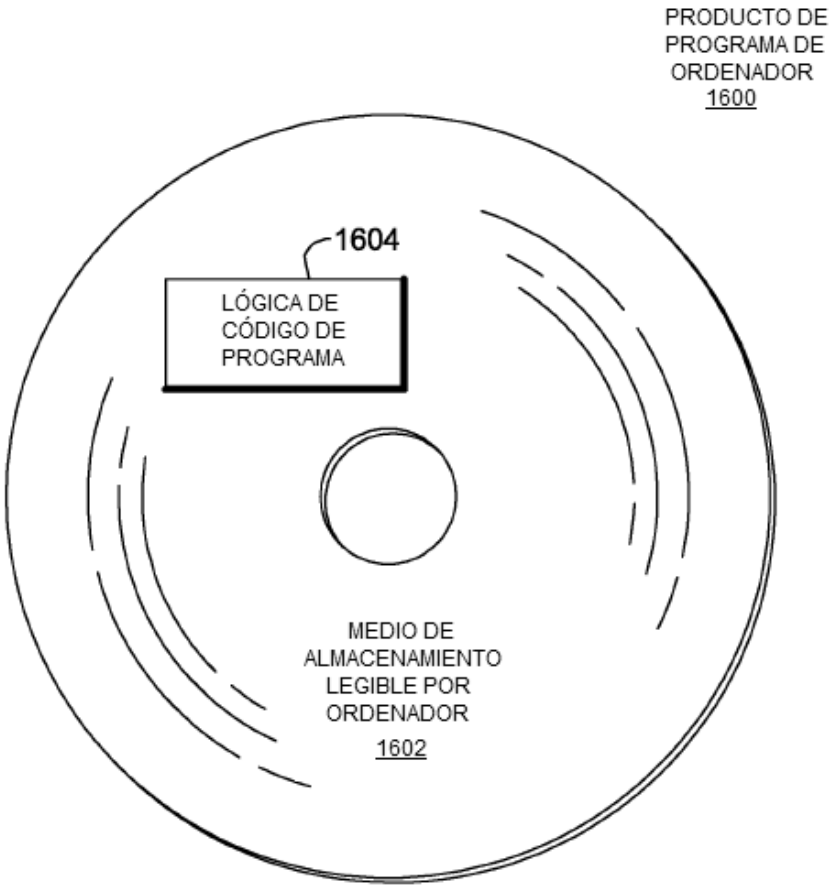
PRODUCTO DE PROGRAMA DE ORDENADOR
1600
1604
LÓGICA DE CÓDIGO DE PROGRAMA
MEDIO DE ALMACENAMIENTO LEGIBLE POR ORDENADOR
1602

FIG. 16

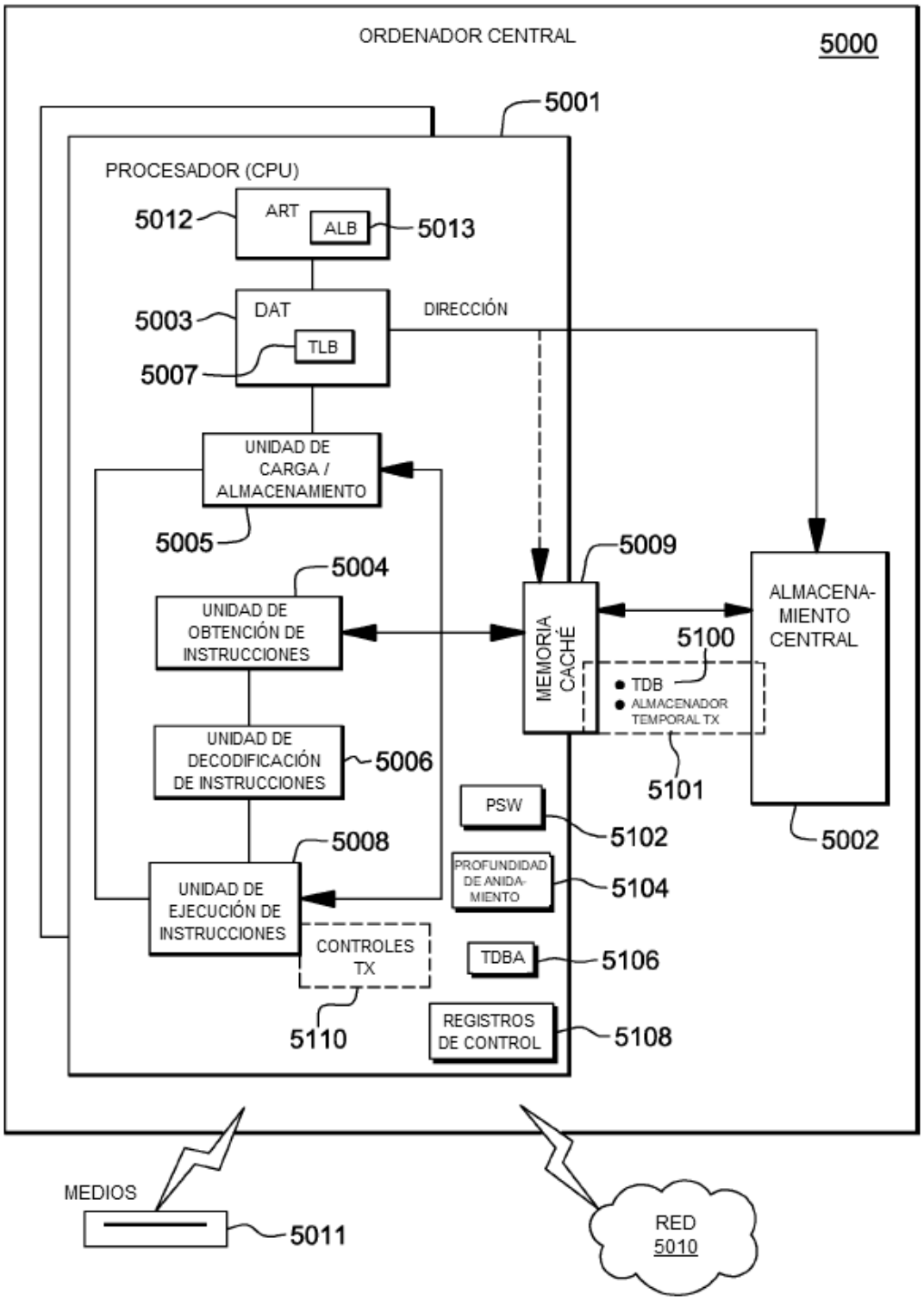
ORDENADOR CENTRAL
5000
5001
PROCESADOR (CPU)
5012
ART
ALB
5013
5003
DAT
TLB
5007
DIRECCIÓN
UNIDAD DE CARGA / ALMACENAMIENTO
5005
5004
UNIDAD DE OBTENCIÓN DE INSTRUCCIONES
UNIDAD DE DECODIFICACIÓN DE INSTRUCCIONES
5006
5008
UNIDAD DE EJECUCIÓN DE INSTRUCCIONES
CONTROLES TX
5110
5009
MEMORIA CACHÉ
5100
TDB
ALMACENADOR TEMPORAL TX
5101
ALMACENA-MIENTO CENTRAL
5002
PSW
5102
PROFUNDIDAD DE ANIDA-MIENTO
5104
TDBA
5106
REGISTROS DE CONTROL
5108
MEDIOS
5011
RED
5010

FIG. 17

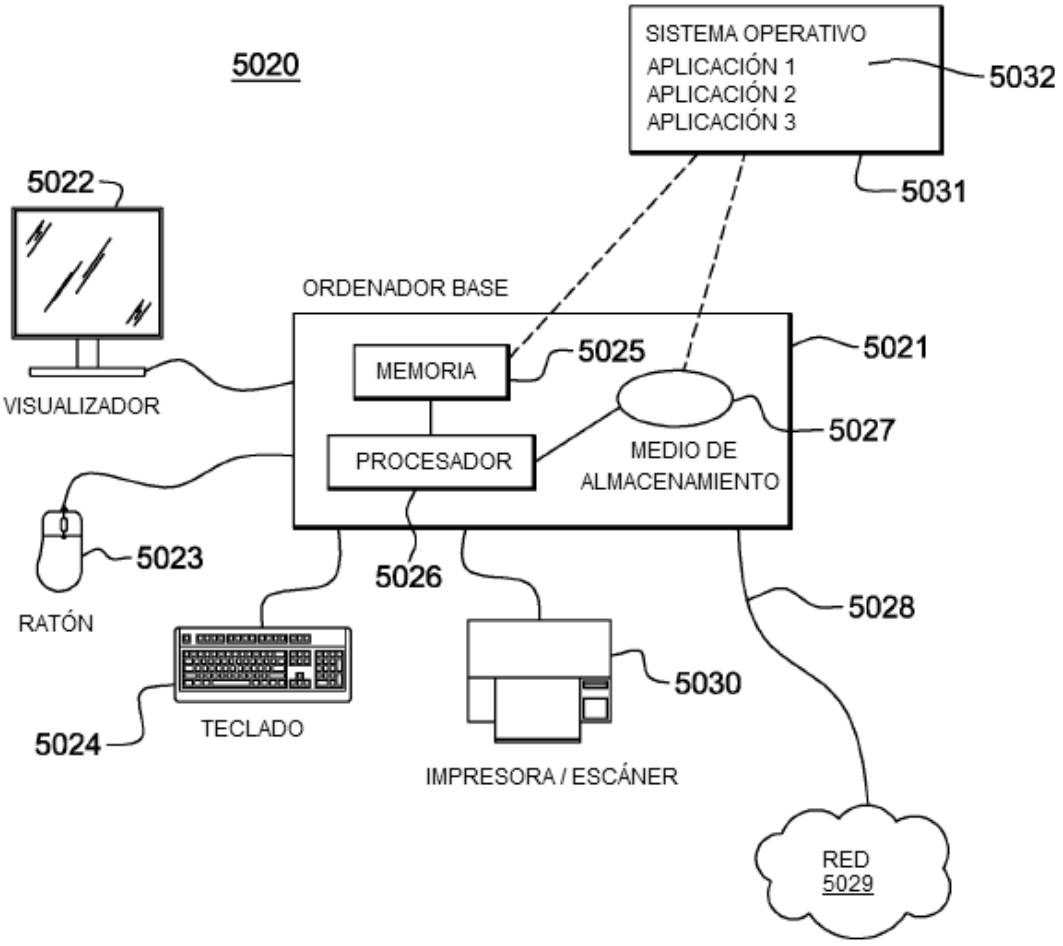
5020
SISTEMA OPERATIVO
APLICACIÓN 1
APLICACIÓN 2
APLICACIÓN 3
5032
5031
5022
ORDENADOR BASE
MEMORIA
5025
5021
PROCESADOR
MEDIO DE ALMACENAMIENTO
5027
VISUALIZADOR
5023
RATÓN
5026
5028
5024
TECLADO
5030
IMPRESORA / ESCÁNER
RED
5029

FIG. 18

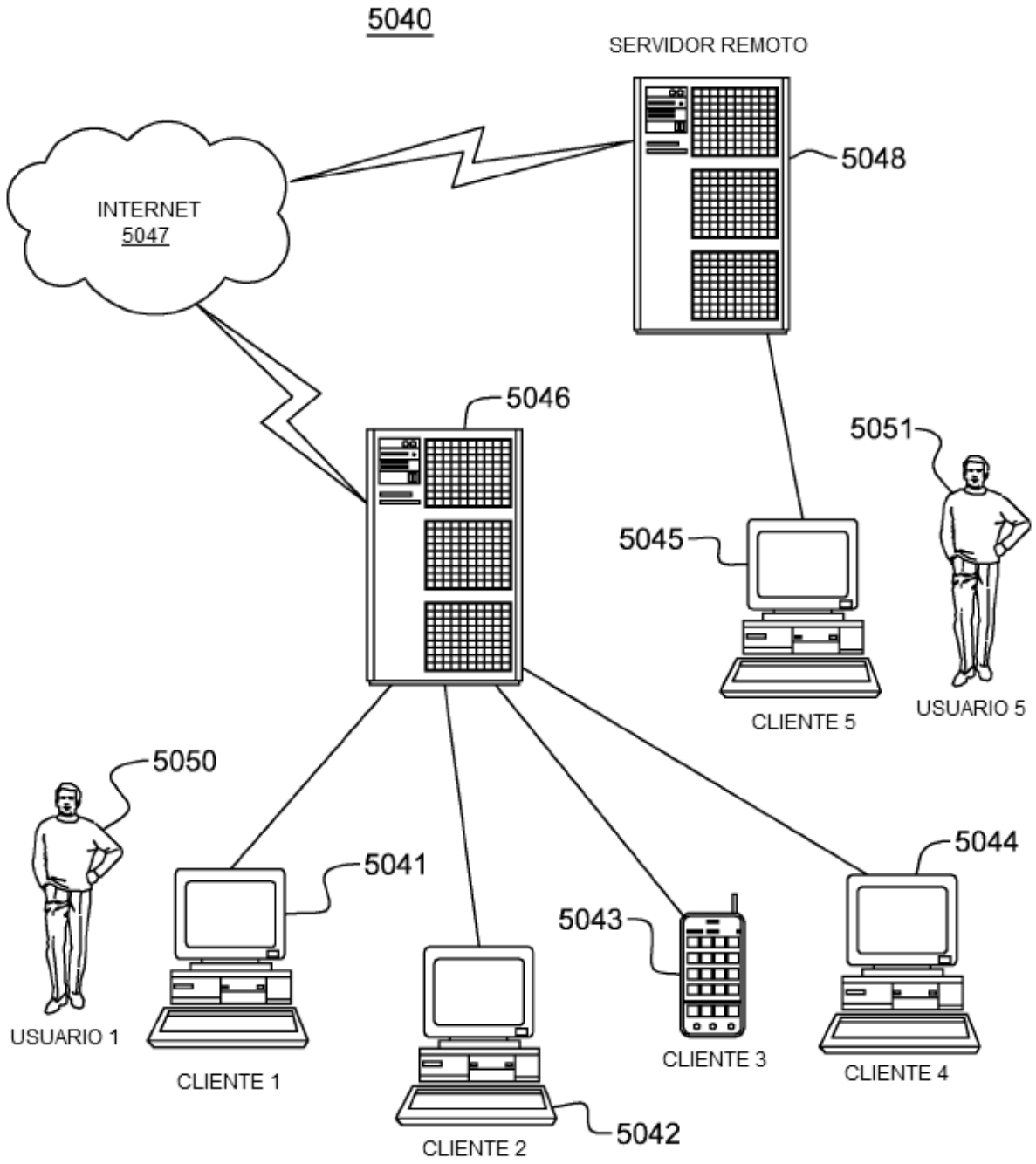
5040
SERVIDOR REMOTO
5048
INTERNET
5047
5046
5051
5045
CLIENTE 5
USUARIO 5
5050
5041
5044
5043
USUARIO 1
CLIENTE 1
CLIENTE 3
CLIENTE 4
CLIENTE 2
5042

FIG. 19

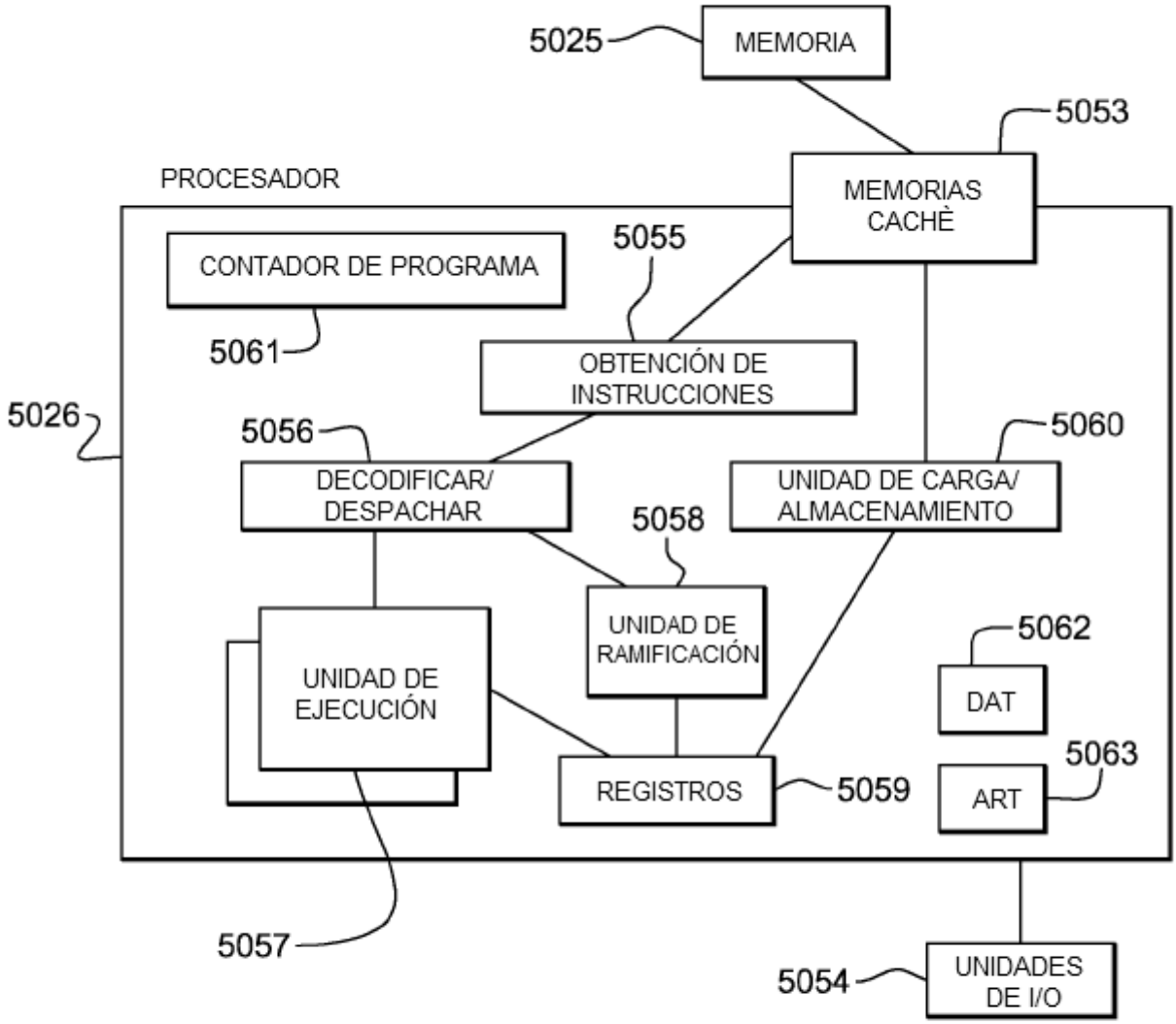
5025
MEMORIA
5053
PROCESADOR
MEMORIAS CACHÈ
5055
CONTADOR DE PROGRAMA
5061
OBTENCIÓN DE INSTRUCCIONES
5026
5056
5060
DECODIFICAR/ DESPACHAR
UNIDAD DE CARGA/ ALMACENAMIENTO
5058
UNIDAD DE RAMIFICACIÓN
5062
UNIDAD DE EJECUCIÓN
DAT
5063
REGISTROS
5059
ART
5057
5054
UNIDADES DE I/O

FIG. 20

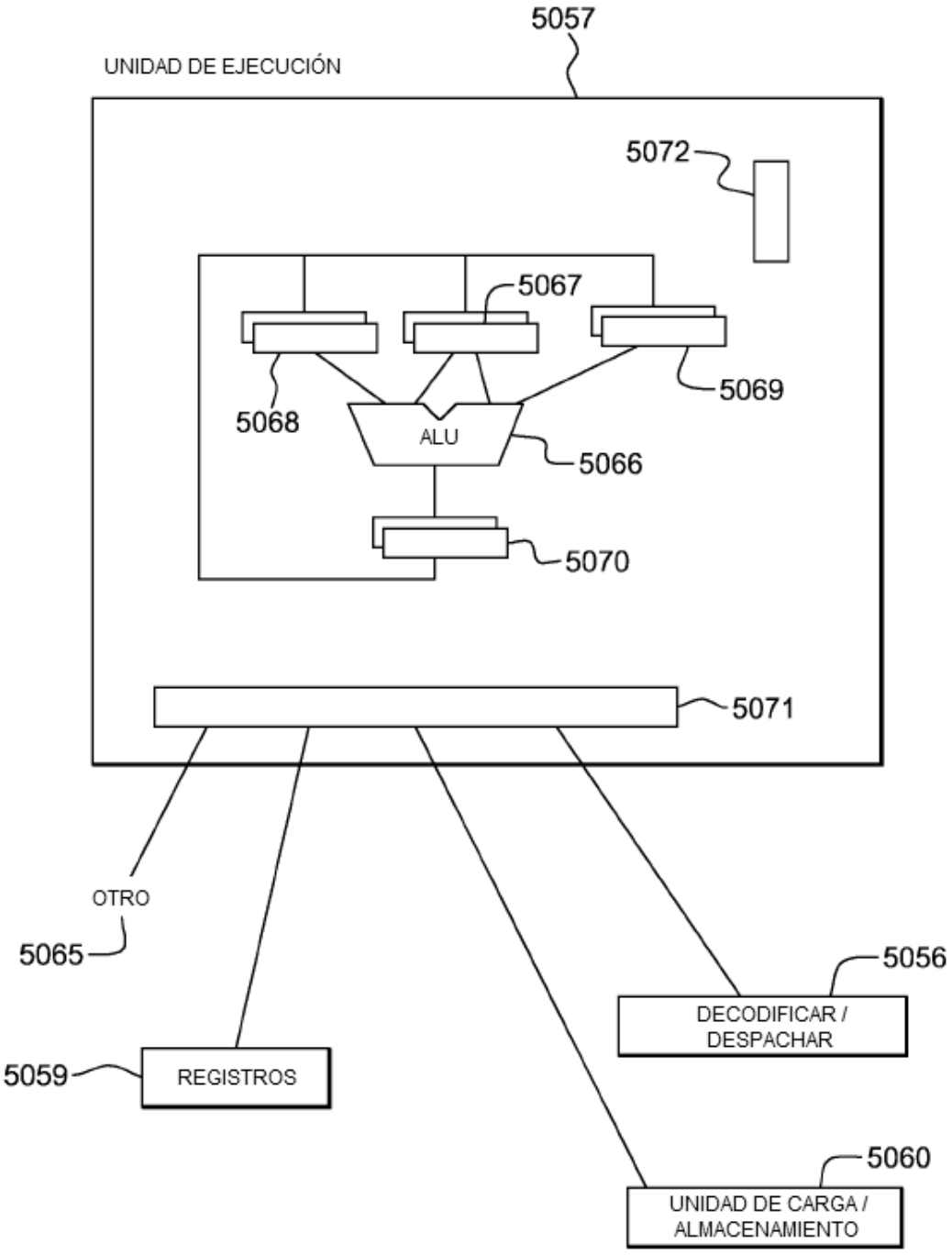
5057
UNIDAD DE EJECUCIÓN
5072
5067
5068
5069
ALU
5066
5070
5071
OTRO
5065
5056
DECODIFICAR / DESPACHAR
5059
REGISTROS
5060
UNIDAD DE CARGA / ALMACENAMIENTO

FIG. 21A

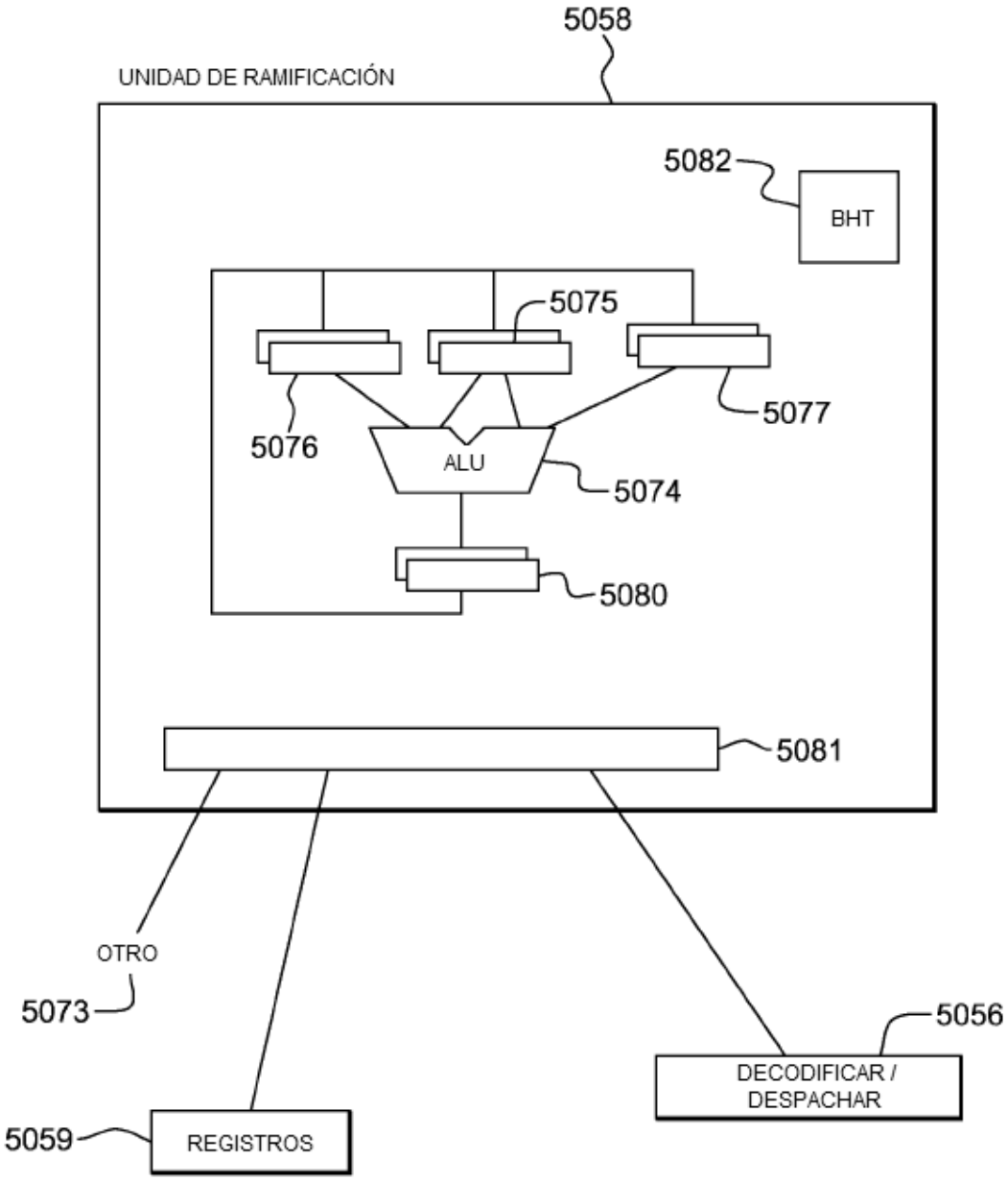
5058
UNIDAD DE RAMIFICACIÓN
5082
BHT
5075
5076
5077
ALU
5074
5080
5081
OTRO
5073
5056
DECODIFICAR / DESPACHAR
5059
REGISTROS

FIG. 21B

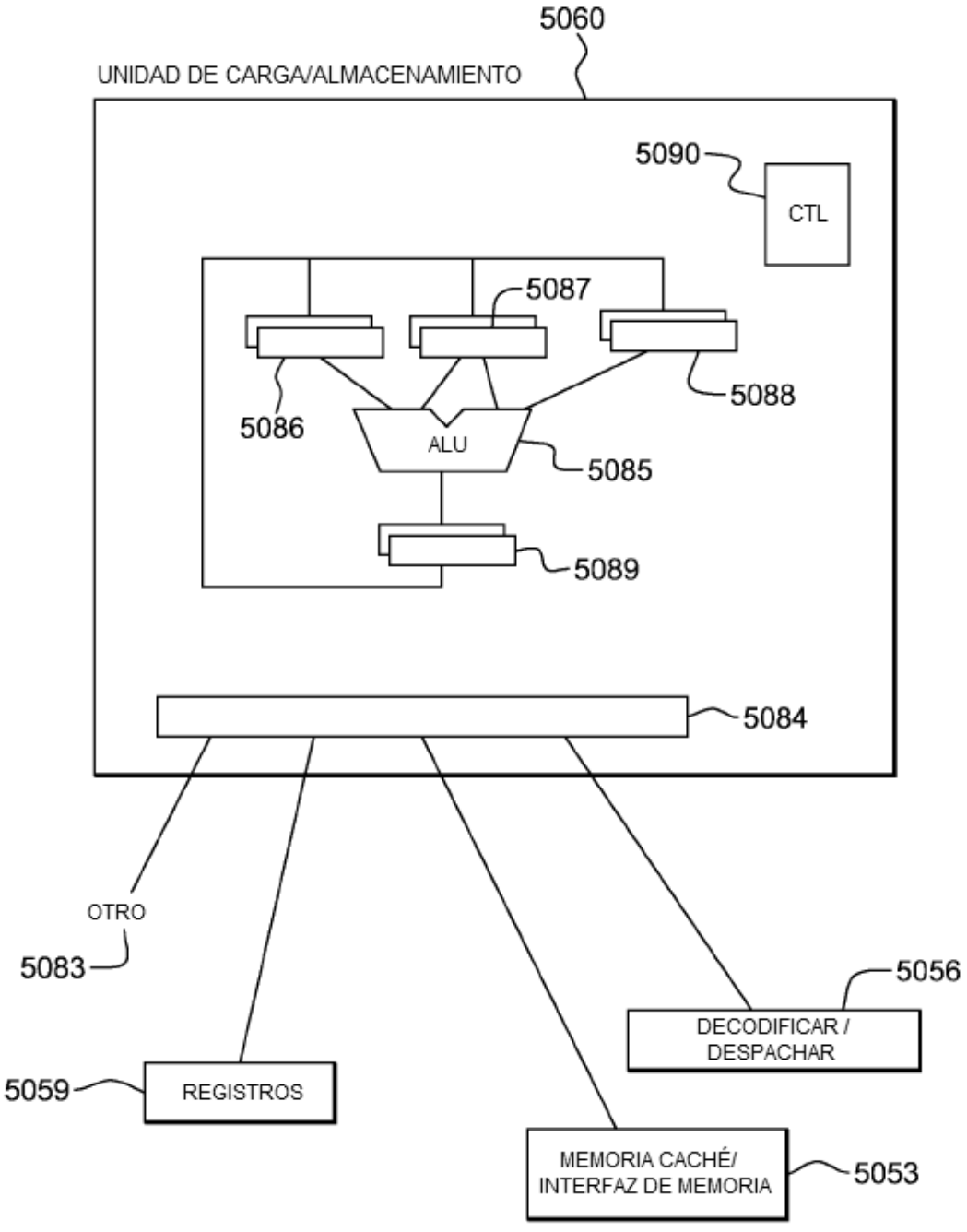
5060
UNIDAD DE CARGA/ALMACENAMIENTO
5090
CTL
5087
5088
5086
ALU
5085
5089
5084
OTRO
5083
5056
DECODIFICAR / DESPACHAR
5059
REGISTROS
MEMORIA CACHÉ/ INTERFAZ DE MEMORIA
5053

FIG. 21C

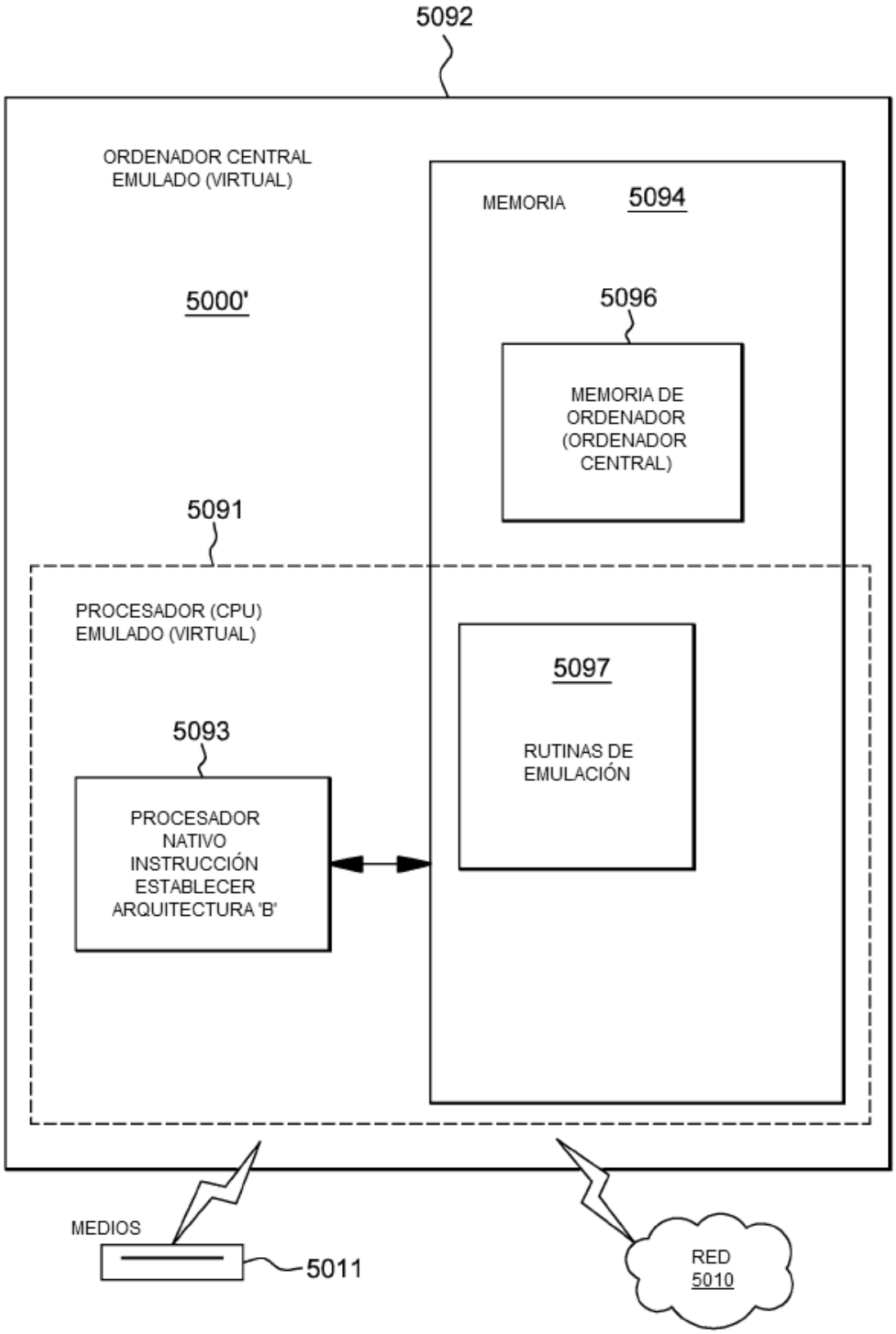
5092
ORDENADOR CENTRAL EMULADO (VIRTUAL)
MEMORIA
5094
5000'
5096
MEMORIA DE ORDENADOR (ORDENADOR CENTRAL)
5091
PROCESADOR (CPU) EMULADO (VIRTUAL)
5097
RUTINAS DE EMULACIÓN
5093
PROCESADOR NATIVO INSTRUCCIÓN ESTABLECER ARQUITECTURA 'B'
MEDIOS
5011
RED
5010

FIG. 22