(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 974 222**

(51) Int. Cl.:

G06F 9/30 (2008.01)

G06F 9/38 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **05.11.2019 PCT/EP2019/080161**

(87) Fecha y número de publicación internacional: **14.05.2020 WO20094601**

(96) Fecha de presentación y número de la solicitud europea: **05.11.2019 E 19798280 (4)**

(97) Fecha y número de publicación de la concesión europea: **21.02.2024 EP 3877841**

(54) Título: **Guardar y restaurar el estado de la máquina entre varias ejecuciones de una instrucción**

(30) Prioridad:

06.11.2018 US 201816182017

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
26.06.2024

(73) Titular/es:

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

(72) Inventor/es:

GIAMEI, BRUCE, CONRAD;
RECKTENWALD, MARTIN;
SCHMIDT, DONALD, WILLIAM;
SLEGEL, TIMOTHY;
PURANIK, ADITYA, NITIN;
FARRELL, MARK;
JACOBI, CHRISTIAN;
BRADBURY, JONATHAN y
ZOELLIN, CHRISTIAN, GERHARD

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Guardar y restaurar el estado de la máquina entre varias ejecuciones de una instrucción

Campo técnico

Uno o más aspectos se refieren, en general, a facilitar el procesamiento dentro de un entorno informático y, en particular, a facilitar dicho procesamiento de instrucciones.

Antecedentes

Una instrucción que se ejecuta en el entorno informático puede requerir un número significativo de ciclos de ejecución para completar una operación. Cuando una instrucción requiere un número significativo de ciclos de ejecución para completarse, la instrucción puede definirse como interrumpible. Por lo tanto, para completar finalmente la instrucción, se realiza un procesamiento adicional. La publicación de solicitud de patente estadounidense número US 2014/0089646 A1, Diewald, H. et al ("Processor with Interruptable Instruction Execution", 27 de marzo de 2014) describe un procesador que incluye una pluralidad de unidades de ejecución. Cada una de las unidades de ejecución incluye un registro de estado configurado para almacenar un valor indicativo del estado de la unidad de ejecución. Al menos una de las unidades de ejecución se configura para ejecutar una instrucción compleja que requiere múltiples ciclos de instrucciones para ejecutarse. La al menos una de las unidades de ejecución también se configura para interrumpir la ejecución de la instrucción compleja para ejecutar una instrucción diferente y reanudar, en función del valor almacenado en el registro de estado, la ejecución de la instrucción compleja en el punto interrumpido después de la ejecución de la instrucción diferente. La patente estadounidense número US 5475822, Sibigtroth, J.M. et al ("Data Processing System for Resuming Instruction Execution after an Interrupt and Method for There'", 12, de diciembre de 1995) divulga un sistema de procesamiento de datos que implementa una instrucción reanudable utilizando dos bytes de instrucciones. Cuando un contador de programa apunta a un primer byte de instrucción, se inicia una primera operación de procesamiento de datos. Si se produce una interrupción durante la ejecución de la primera operación de procesamiento de datos, los cálculos de datos intermedios mantenidos en una pluralidad de registros temporales se guardan en la memoria de pila en una ubicación a la que apunta el registro de punteros de pila. El contador de programa se incrementa para apuntar a un segundo byte de la instrucción. Se ejecuta una operación de reanudación de instrucciones y se accede a los resultados intermedios de la operación de procesamiento de datos desde la memoria de pila y se restauran en las respectivas ubicaciones de almacenamiento dentro del sistema de procesamiento de datos. Una vez restaurados los resultados intermedios, el contador de programa se reduce para apuntar al primer byte de instrucción y la instrucción continúa ejecutando la operación de procesamiento de datos como si no se hubiera producido ninguna interrupción. La patente estadounidense número US 4779192, de Torii, S. et al ("Vector Processor with a Synchronously Controlled Operand Fetch Circuits", 18 de octubre de 1988) divulga un procesador vectorial para leer secuencialmente elementos de una pluralidad de operandos vectoriales y almacenar secuencialmente los resultados de las operaciones en los operandos vectoriales, que comprende: contadores de operandos para indicar los números de los elementos de cada operando; registros de direcciones para cada operando; un primer comparador para comparar cada elemento del vector; registros de números máximos para almacenar los números máximos de elementos de los operandos respectivos; un segundo comparador para comparar el contador de operandos de cada operando con el contenido de los registros de números máximos de cada operando con respecto a cada operando; y un circuito de control para actualizar de forma independiente los contadores de operandos y registros de direcciones de operandos de cada operando en respuesta a todas o partes de las salidas de los comparadores primero y segundo.

Compendio

La presente invención se define en las reivindicaciones independientes adjuntas a las que se debe hacer referencia. Las características ventajosas se exponen en las reivindicaciones dependientes adjuntas.

Según a presente invención, se proporciona un método, un sistema, un producto de programa informático y un programa informático según las reivindicaciones independientes. Se superan los inconvenientes de la técnica anterior y se proporcionan ventajas adicionales mediante la aportación de un producto de programa informático para facilitar el procesamiento dentro de un entorno informático.

El producto de programa informático incluye un soporte de almacenamiento legible por procesador e instrucciones de almacenamiento para realizar un método. El método incluye determinar (1100) que el procesamiento de una operación de una instrucción que se ejecuta en el procesador se ha interrumpido antes de su finalización; extraer (1102) los metadatos del procesador, sobre la base de la determinación de que el procesamiento de la operación se ha interrumpido, los metadatos son metadatos actuales del procesador (1104), en donde la instrucción es una instrucción de clasificación (1110), y los metadatos incluyen información acerca de una o más listas de entrada para la instrucción de clasificación (1112); almacenar (1106) los metadatos en una ubicación asociada con la instrucción, en donde la ubicación es un bloque de parámetros (360) en la memoria designado por la instrucción; extraer los metadatos del bloque de parámetros (1054) y cargar los metadatos en el procesador (1058), sin repetir las tareas para reproducir los metadatos; y usar (1108) los metadatos que se cargan al volver a ejecutar la instrucción para reenviar el procesamiento de la instrucción desde donde se interrumpió, los metadatos son metadatos producidos previamente y almacenados utilizados para preparar el procesador cuando la instrucción se vuelve a ejecutar, en donde la información se almacena en un búfer de estado de continuación en el bloque de parámetros (1054) y la información depende del modelo en el sentido de que se almacena o presenta de manera diferente según el modelo de procesador, y en donde, al volver a

ejecutar la instrucción para reanudar el procesamiento, un número de versión de modelo en el bloque de parámetros indica qué contenidos del búfer de estado de continuación se pueden interpretar.

Al extraer los metadatos para una operación interrumpida y guardar esos datos, cuando la instrucción se vuelve a ejecutar, los metadatos guardados pueden cargarse y usarse, en lugar de repetir tareas para reproducir los metadatos. Esto ahorra tiempo y mejora el rendimiento del procesador.

En un ejemplo, la instrucción es una instrucción de clasificación, y los metadatos incluyen información acerca de una o más listas de entrada para la instrucción de clasificación. Por ejemplo, los metadatos incluyen información sobre las comparaciones anteriores realizadas de los registros de una o más listas de entrada para indicar las próximas comparaciones que se realizarán. Las siguientes comparaciones a realizar se indican, por ejemplo, sin repetir las comparaciones anteriores.

En un ejemplo, la determinación incluye comprobar un código de condición establecido en función de la terminación de la instrucción; el código de condición establecido en un valor seleccionado indica la finalización parcial de la instrucción.

Como ejemplo, la ubicación en la que se almacenan los metadatos es un bloque de parámetros en la memoria designado por la instrucción. La ubicación del bloque de parámetros en la memoria se designa, por ejemplo, mediante el contenido de un registro implicado de la instrucción. En un ejemplo particular, el bloque de parámetros incluye un búfer de estado de continuación para almacenar los metadatos, los metadatos incluyen datos de estado internos del procesador. Además, en un ejemplo, el bloque de parámetros incluye un indicador de continuación para indicar la finalización parcial de la operación.

En esta memoria también se describen y se reivindican métodos implementados en ordenador y sistemas relacionados con uno o más aspectos. Además, también se describen y pueden reivindicarse en esta memoria servicios relacionados con uno o más aspectos.

Se obtienen características y ventajas adicionales a través de las técnicas descritas en esta memoria. Se describen en detalle en esta memoria otras realizaciones y aspectos, y se consideran parte de los aspectos reivindicados.

Breve descripción de los dibujos

Uno o más aspectos se señalan particularmente y se reivindican claramente como ejemplos en las reivindicaciones al final de la memoria descriptiva. Lo anterior y los objetos, características y ventajas de uno o más aspectos son evidentes a partir de la siguiente descripción detallada tomada junto con los dibujos adjuntos, en los que:

la FIG. 1A representa un ejemplo de un entorno informático para incorporar y utilizar uno o más aspectos de la presente invención;

la FIG. 1B representa detalles adicionales de un procesador de la FIG. 1A, según uno o más aspectos de la presente invención;

la FIG. 2 representa otro ejemplo de un entorno informático para incorporar y usar uno o más aspectos de la presente invención;

la FIG. 3A representa un formato de una instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIG. 3B representa un ejemplo de campos de un registro implicado, el registro general 0, utilizado por la instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIG. 3C representa un ejemplo de códigos de función para la instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIG. 3D representa un ejemplo de un campo de un registro implicado, registro general 1, utilizado por la instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIG. 3E representa un ejemplo de contenido de un registro, R_1, especificado por la instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIG. 3F representa un ejemplo de contenido de un registro, R_1+1, usado por la instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIG. 3G representa un ejemplo de contenido de un registro, R_2, especificado por la instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIG. 3H representa un ejemplo de contenido de un registro, R_2+1, usado por la instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIG. 3I representa un ejemplo de contenido de un bloque de parámetros usado por la función SORTL-QAF de la instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIG. 3J representa un ejemplo de un formato de registro de longitud fija utilizado por la instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIG. 3K representa un ejemplo de contenido de un bloque de parámetros usado por la función SORTL-SFLR de la instrucción Clasificar Listas, según un aspecto de la presente invención;

la FIGS. 4A-4B representan ejemplos de SORTL-SFLR, según uno o más aspectos de la presente invención;

la FIG. 5A representa un ejemplo de un resumen de valores para entradas a la función SORTL-SFLR, según un aspecto de la presente invención;

la FIG. 5B representa un ejemplo de restricciones para modificaciones en los campos de longitud y dirección de la lista de entrada para la función SORTL-SFLR, según un aspecto de la presente invención;

la FIG. 6A representa un ejemplo de una primera ubicación de operando/primer operando antes de ejecutar SORTL con una indicación de modo de fusión establecida en cero, según un aspecto de la presente invención;

la FIG. 6B representa un ejemplo de una primera ubicación de operando/primer operando después de ejecutar SORTL con una indicación de modo de fusión establecida en cero, según un aspecto de la presente invención;

la FIG. 6C representa un ejemplo de una segunda ubicación de operando/segundo operando antes de ejecutar SORTL con una indicación de modo de fusión establecida en cero, según un aspecto de la presente invención;

la FIG. 6D representa un ejemplo de una segunda ubicación de operando/segundo operando después de ejecutar SORTL con una indicación de modo de fusión establecida en cero, según un aspecto de la presente invención;

la FIG. 7A representa un ejemplo de una primera ubicación de operando/primer operando antes de ejecutar SORTL con una indicación de modo de fusión establecida en uno, según un aspecto de la presente invención;

la FIG. 7B representa un ejemplo de una primera ubicación de operando/primer operando después de ejecutar SORTL con una indicación de modo de fusión establecida en uno, según un aspecto de la presente invención;

la FIG. 8 representa un ejemplo de ciertos campos de un bloque de parámetros según un aspecto de la presente invención;

la FIG. 9 representa un ejemplo de un formato de registro de longitud variable utilizado por la instrucción Clasificar Listas, según un aspecto de la presente invención;

las FIGS. 10A-10B representan el procesamiento asociado con la interrupción de una operación de una instrucción y la reejecución de la instrucción, según un aspecto de la presente invención;

las FIGS. 11A-11B representan un ejemplo para facilitar el procesamiento en un entorno informático, según un aspecto de la presente invención;

la FIG. 12A representa otro ejemplo de un entorno informático para incorporar y utilizar uno o más aspectos de la presente invención;

la FIG. 12B representa detalles adicionales de la memoria de la FIG. 12A;

la FIG. 13 representa una realización de un entorno informático en la nube; y

la FIG. 14 representa un ejemplo de capas de modelo de abstracción.

Descripción detallada

Según un aspecto, se proporciona una capacidad para facilitar el procesamiento dentro de un entorno informático. Como ejemplo se proporciona una única instrucción (por ejemplo, una única instrucción de máquina de hardware diseñada en la interfaz de hardware/software) para realizar una operación, tal como clasificar y/o fusionar registros de datos. La instrucción se ejecuta, por ejemplo, en un procesador de uso general.

Al ejecutar la instrucción, puede ser necesario un número significativo de ciclos de ejecución para completar la operación. Por lo tanto, en un aspecto, la instrucción se define como interrumpible. Cuando se interrumpe la instrucción, la operación (por ejemplo, clasificar y/o fusionar) solo se completa parcialmente. La ejecución de la instrucción finaliza con el establecimiento de un código de condición en un valor que informa al programa (por ejemplo, al programa que emite la instrucción) de la finalización parcial de la operación. El programa puede entonces volver a ejecutar la instrucción para reanudar el procesamiento.

La instrucción, en una realización, emplea un número (por ejemplo, un número significativo) de ciclos de ejecución para preparar el procesador con metadatos antes de que se produzcan los resultados. La preparación del procesador con los metadatos se realiza cada vez que se ejecuta o se vuelve a ejecutar la instrucción. Por lo tanto, según un aspecto de la presente invención, se almacenan y usan los metadatos producidos previamente. de manera que los metadatos producidos anteriormente no tengan que reproducirse en el momento en que se vuelva a ejecutar la instrucción.

En un ejemplo, la instrucción es una instrucción de clasificación que clasifica y/o fusiona los registros de una o más listas de entrada introducidas en la instrucción. Para tal ejemplo, los metadatos incluyen el estado interno del procesador, incluyendo, por ejemplo, información sobre las listas de entrada, tal como información sobre comparaciones anteriores de registros de las listas de entrada para determinar las próximas comparaciones que se realizarán.

El procesador extrae los metadatos y los almacena en una ubicación proporcionada por el programa. Luego, cuando la instrucción se va a volver a ejecutar después de una interrupción, los metadatos se extraen de la ubicación y se cargan en el procesador, sin utilizar tareas para reproducir los metadatos. Esto ahorra el tiempo que habría sido necesario para generar los metadatos de la operación.

Con referencia a la FIG. 1A se describe una realización de un entorno informático para incorporar y usar uno o más aspectos de la presente invención. Un entorno informático 100 incluye, por ejemplo, un procesador 102 (por ejemplo, una unidad de procesamiento central), una memoria 104 (por ejemplo, memoria principal, también conocida como memoria de sistema, almacenamiento principal, almacenamiento central, almacenamiento) y uno o más dispositivos de entrada/salida (E/S) y/o interfaces 106 acoplados entre sí mediante, por ejemplo, uno o más buses 108 y/u otras conexiones.

En un ejemplo, el procesador 102 se basa en la arquitectura de hardware z/Architecture® ofrecida por International Business Machines Corporation, Armonk, Nueva York, y forma parte de un servidor, tal como un servidor IBM Z®, que también ofrece International Business Machines Corporation e implementa la arquitectura de hardware z/Architecture. Una realización de la arquitectura de hardware z/Architecture se describe en una publicación titulada "z/Architecture Principles of Operation", publicación de IBM n.º SA22-7832-11, 12ª edición, septiembre de 2017. Sin embargo, la arquitectura de hardware z/Architecture es solo un ejemplo de arquitectura; otras arquitecturas y/u otros tipos de entornos informáticos pueden incluir y/o usar uno o más aspectos de la presente invención. En un ejemplo, el procesador ejecuta un sistema operativo, tal como el sistema operativo z/OS®, también ofrecido por International Business Machines Corporation.

El procesador 102 incluye una pluralidad de componentes funcionales utilizados para ejecutar instrucciones. Como se representa en la FIG. 1B, estos componentes funcionales incluyen, por ejemplo, un componente de búsqueda de instrucciones 120 para buscar instrucciones que han de ser ejecutadas; una unidad de descodificación de instrucciones 122 para descodificar las instrucciones buscadas y obtener operandos de las instrucciones descodificadas; un componente de ejecución de instrucciones 124 para ejecutar las instrucciones descodificadas; un componente de acceso a memoria 126 para acceder a la memoria para la ejecución de instrucciones, si es necesario; y un componente de reescritura 130 para proporcionar los resultados de las instrucciones ejecutadas. Uno o más de estos componentes pueden, según uno o más de aspectos de la presente invención, incluir al menos una parte o tener acceso a uno o más de otros componentes que proporcionan procesamiento de clasificación/fusión (u otro procesamiento que pueda usar uno o más aspectos de la presente invención), tal como se describe en la presente memoria. El uno o más de otros componentes incluyen, por ejemplo, un componente de clasificación/fusión (u otro componente) 136. La funcionalidad proporcionada por el componente 136 se describe con más detalle a continuación.

Otro ejemplo de un entorno informático para incorporar y usar uno o más aspectos de la presente invención se describe con referencia a la FIG. 2. En un ejemplo, el entorno informático se basa en la arquitectura de hardware z/Architecture; sin embargo, el entorno informático puede basarse en otras arquitecturas ofrecidas por International Business Machines Corporation u otros.

Con referencia a la FIG. 2, en un ejemplo, el entorno informático incluye un complejo electrónico central (CEC) 200. El CEC 200 incluye una pluralidad de componentes, tales como, por ejemplo, una memoria 202 (también conocida como memoria del sistema, memoria principal, almacenamiento principal, almacenamiento central, almacenamiento) acoplada a uno o más procesadores (también conocidos como unidades centrales de procesamiento (CPU) 204) y a un subsistema de entrada/salida 206.

La memoria 202 incluye, por ejemplo, una o más particiones lógicas 208, un hipervisor 210 que gestiona las particiones lógicas y el firmware de procesador 212. Un ejemplo de hipervisor 210 es el hipervisor Processor Resource/System Manager (PR/SM™), ofrecido por International Business Machines Corporation, Armonk, Nueva York. Tal como se usa en la presente memoria, el firmware incluye, por ejemplo, el microcódigo del procesador. Incluye, por ejemplo, las instrucciones de nivel de hardware y/o las estructuras de datos utilizadas en la implementación del código máquina de nivel superior. En una realización, incluye, por ejemplo, código propietario que normalmente se entrega como microcódigo que incluye software fiable o microcódigo específico para el hardware subyacente y controla el acceso del sistema operativo al hardware de sistema.

Cada partición lógica 208 es capaz de funcionar como un sistema independiente. Es decir, cada partición lógica se puede restablecer de forma independiente, ejecutar un sistema operativo invitado 220, tal como un sistema operativo z/OS, u otro sistema operativo, y operar con diferentes programas 222. Un sistema operativo o un programa de aplicación que se ejecuta en una partición lógica parece tener acceso a un sistema entero y completo, pero en realidad solo una parte de él está disponible.

La memoria 202 se acopla a los procesadores (por ejemplo, las CPU) 204, que son recursos de procesadores físicos que pueden asignarse a las particiones lógicas. Por ejemplo, una partición lógica 208 incluye uno o más procesadores lógicos, cada uno de los cuales representa la totalidad o una parte de un recurso de procesador físico 204 que puede

asignarse dinámicamente a la partición lógica.

Además, la memoria 202 se acopla al subsistema de E/S 206. El subsistema de E/S 206 puede formar parte del complejo electrónico central o estar separado del mismo. Dirige el flujo de información entre el almacenamiento principal 202 y las unidades de control de entrada/salida 230 y los dispositivos de entrada/salida (E/S) 240 acoplados al complejo electrónico central.

Se pueden utilizar muchos tipos de dispositivos de E/S. Un tipo particular consiste en un dispositivo 250 de almacenamiento de datos. El dispositivo 250 de almacenamiento de datos puede almacenar uno o más programas 252, una o más instrucciones 254 de programa legibles por ordenador y/o datos, etc. Las instrucciones de programa legibles por ordenador pueden configurarse para llevar a cabo funciones de realizaciones de aspectos de la invención.

En un ejemplo, el procesador 204 incluye un componente de clasificación/fusión (u otro componente) 260 para realizar una o más operaciones de clasificación y/o fusión (u otras operaciones que puedan usar uno o más aspectos de la presente invención). En diversos ejemplos, puede haber uno o más componentes que realicen estas tareas. Son posibles muchas variaciones.

El complejo electrónico central 200 puede incluir y/o acoplarse a medios de almacenamiento de sistemas informáticos extraíbles/no extraíbles, volátiles/no volátiles. Por ejemplo, puede incluir y/o acoplarse a un medio magnético no extraíble y no volátil (normalmente denominado "disco duro"), una unidad de disco magnético para leer y escribir en un disco magnético extraíble y no volátil (por ejemplo, un "disquete") y/o una unidad de disco óptico para leer o escribir en un disco óptico extraíble y no volátil, tal como un CD-ROM, DVD-ROM u otros medios ópticos. Debe entenderse que podrían usarse otros componentes de hardware y/o software junto con el complejo electrónico central 200. Los ejemplos incluyen, pero no se limitan a: microcódigo, controladores de dispositivo, unidades de procesamiento redundantes, agrupaciones de unidades de disco externo, sistemas RAID, unidades de cinta y sistemas de almacenamiento de archivos de datos, etc.

Además, el complejo electrónico central 200 puede funcionar con muchos otros entornos o configuraciones de sistemas informáticos de uso general o de propósito especial. Los ejemplos de sistemas, entornos y/o configuraciones informáticos bien conocidos que pueden ser adecuados para su uso con el complejo electrónico central 200 incluyen, pero sin limitación a esto, sistemas de ordenador personal (PC), sistemas informáticos de servidor, clientes ligeros, clientes pesados, dispositivos de mano o portátiles, sistemas multiprocesadores, sistemas basados en microprocesadores, cajas de conexión, electrónica de consumo programable, PC de red, sistemas de miniordenadores, sistemas de ordenadores centrales y entornos informáticos en la nube distribuidos que incluyen cualquiera de los anteriores sistemas o dispositivos, y similares.

Aunque en la presente memoria se describen diversos ejemplos de entornos informáticos, uno o más aspectos de la presente invención se pueden usar con muchos tipos de entornos. Los entornos informáticos proporcionados en la presente memoria son solo ejemplos. Además, aunque uno o más aspectos de la presente invención se describen con referencia a una instrucción de clasificación, uno o más de los aspectos son aplicables a otros procesamientos y/o instrucciones que usan un número significativo de ciclos de ejecución y son interrumpibles. La instrucción de clasificación es solo un ejemplo.

Según un aspecto de la presente invención, un procesador, tal como el procesador 102 o 204, emplea una función de clasificación mejorada que proporciona un mecanismo para clasificar múltiples listas de datos de entrada no clasificados en una o más listas de datos de salida clasificados. En un ejemplo, la instalación de clasificación mejorada se instala en el sistema cuando se establece un indicador de instalación, por ejemplo, en uno. Como ejemplo particular de la arquitectura de hardware z/Architecture, el bit 150 de instalación se establece, por ejemplo, en uno cuando la instalación de conversión se instala en el modo de arquitectura z/Architecture. La instalación también proporciona, en una realización, un mecanismo para fusionar múltiples listas de datos de entrada clasificados en una única lista de datos de salida clasificados. La instalación incluye, por ejemplo, la instrucción Clasificar Listas, una realización de la cual se describe más adelante.

Una realización de los detalles relacionados con una instrucción Clasificar Listas se describe con referencia a las FIGS. 3A-3K. Esta instrucción se ejecuta, en un ejemplo, en un procesador de uso general (por ejemplo, el procesador 102 o 204). En la descripción de la presente memoria se indican ubicaciones específicas, campos específicos y/o tamaños específicos de los campos (por ejemplo, bytes y/o bits específicos). Sin embargo, se pueden proporcionar otras ubicaciones, campos y/o tamaños. Además, aunque se especifica el ajuste de un bit a un valor particular, por ejemplo, uno o cero, esto es solo un ejemplo. El bit puede establecerse en un valor diferente, tal como el valor opuesto o en otro valor, en otros ejemplos. Son posibles muchas variaciones.

Con referencia a la FIG. 3A, en un ejemplo, un formato de una instrucción Clasificar Listas (SOFTL) 300 es un formato RRE que indica un registro y una operación de registro con un código de operación extendido (opcode). Como ejemplo, la instrucción incluye un campo de código de operación 302 (por ejemplo, los bits 0-15) que tiene un código de operación que indica una operación de clasificación y/o fusión, un primer campo de registro (R_1) 304 (por ejemplo, los bits 24-27) que designa una primera pareja de registros generales; un segundo campo de registro (R_2) 306 (por ejemplo, los bits 28-31) que designa una segunda pareja de registros generales. El contenido de un registro designado por el campo R_1 304 especifica la ubicación de un primer operando (en almacenamiento), el contenido de un registro designado por el campo R_2 306 especifica la ubicación de un segundo operando (en almacenamiento). El contenido de $R_1 + 1$ especifica la longitud de primer operando y el contenido de $R_2 + 1$ especifica la longitud de segundo operando.

En un ejemplo, los bits 16-23 de la instrucción están reservados y deben contener ceros; de lo contrario, es posible que el programa no funcione de manera compatible en el futuro. Como se usa en la presente memoria, el programa es el que emite la instrucción Clasificar Listas. Puede ser un programa de usuario, un sistema operativo u otro tipo de programa.

En una realización, la ejecución de la instrucción incluye el uso de uno o más registros generales implicados (es decir, registros no designados explícitamente por la instrucción). Por ejemplo, los registros generales 0 y 1 se usan en la ejecución de la instrucción Clasificar Listas, tal como se describe en la presente memoria. El registro general 0 se usa, en un ejemplo, para especificar si se va a realizar la fusión y para especificar una función de clasificación a realizar por la instrucción, y el registro general 1 se usa para proporcionar una ubicación de un bloque de parámetros usado por la instrucción. En otro ejemplo, el registro general 0 no se usa para especificar si se va a realizar la fusión; en cambio, la fusión la establece o no la máquina (por ejemplo, el procesador) y no se puede cambiar mediante un indicador de modo. Son posibles otras variaciones.

Como ejemplo, con referencia a la FIG. 3B, un registro general 0 (308) incluye un campo de modo de fusión 310 (descrito más adelante) y un campo de código de función 312. En un ejemplo particular, las posiciones de bits 57-63 del registro general 0 contienen un código de función; pero en otras realizaciones se pueden usar otros bits para contener el código de función. En un ejemplo, cuando los bits 57-63 del registro general 0 designan un código de función no asignado o desinstalado, se reconoce una excepción de especificación.

Los ejemplos de códigos de función asignados para la instrucción Clasificar Listas se muestran en la FIG. 3C e incluyen, por ejemplo: el código de función 0 (313) indica una función SORTL-QAF (consulta de funciones disponibles); el código de función 1 (315) indica una función SORTL-SFLR (clasificar registros de longitud fija); y el código de función 2 (317) que indica una función SORTL-SVLR (clasificar registros de longitud variable). Cada código usa un bloque de parámetros y el tamaño del bloque de parámetros depende, en un ejemplo, de la función. Por ejemplo, para la función SORTL-QAF, el bloque de parámetros es de 32 bytes; y para SORTL-SFLR y SORTL-SVLR, el bloque de parámetros es $576 + 16 \times N_{IS}$, donde N_{IS} es un número de listas de entrada, según lo especificado por el tamaño de interfaz. Otros códigos de función no están asignados en este ejemplo. Aunque se describen funciones y códigos de función ejemplares, se pueden usar otras funciones y/o códigos de función.

Como se ha indicado anteriormente, el registro general 0 también incluye un campo de modo de fusión 310. En un ejemplo, el bit 56 del registro general 0 especifica un modo de operación (modo de fusión) que se aplica, por ejemplo, a las funciones SORTL-SFLR y SORLT-SVLR. El bit 56 del registro general 0 se ignora, en un ejemplo, cuando la función especificada es SORTL-QAF. Además, en un ejemplo, se ignoran las posiciones de bits 0-55 del registro general 0.

Con referencia a la FIG. 3D se describen detalles adicionales con respecto a otro registro implicado, registro general 1, utilizado por la instrucción Clasificar Listas. El contenido del registro general 1 (314) especifica, por ejemplo, una dirección lógica 316 del byte situado más a la izquierda de un bloque de parámetros almacenado. En un ejemplo, el bloque de parámetros debe designarse en un límite de dos palabras; de lo contrario, se reconoce una excepción de especificación. Más abajo se describen más detalles con respecto al bloque de parámetros.

Para las funciones especificadas (p. ej., SORTL-QAF, SORTL-SFLR, SORTL-SVLR), el contenido de los registros generales 0 y 1 no se modifica. Además, en un ejemplo, el campo R_1 304 designa una pareja par-impar de registros generales. Es para designar un registro par y no para designar el registro general 0; de lo contrario, se reconoce una excepción de especificación. Cuando la función especificada es SORTL-SFLR o SORTL-SVLR, como se muestra en las FIGS. 3E-3F, el contenido de un registro general R_1 318 especifica, por ejemplo, una dirección lógica 320 del byte más a la izquierda del primer operando, y el contenido de un registro general $R_1 + 1$ (322) especifica una longitud 324 del primer operando en, por ejemplo, bytes. Cuando la función especificada es SORTL-SFLR o SORTL-SVLR, el primer operando, por ejemplo, debe designarse en un límite de dos palabras; de lo contrario, se reconoce una excepción de especificación. Los datos, en forma de registros, se seleccionan de un conjunto de listas de entrada y se almacenan en la ubicación de primer operando (por ejemplo, comenzando en la dirección especificada R_1). Cuando se especifica la función SORTL-QAF, se ignora el contenido de los registros generales R_1 y $R_1 + 1$.

Además, para las funciones especificadas (por ejemplo, SORTL-QAF, SORTL-SFLR, DFLTCC-CMPR y SORTL-SVLR), en un ejemplo, el campo R_2 306 designa un pareja par-impar de registros generales. Es para designar un registro par y no para designar el registro general 0; de lo contrario, se reconoce una excepción de especificación. Cuando la función especificada es SORTL-SFLR o SORTL-SVLR, y el modo de fusión (MM) es cero, como se muestra en las FIGS. 3G-3H, el contenido de un registro general R_2 326 especifica, por ejemplo, una dirección lógica 328 del byte más a la izquierda del segundo operando, y el contenido de un registro general $R_2 + 1$ (330) especifica una longitud 332 del segundo operando en, por ejemplo, bytes. Cuando la función especificada es SORTL-SFLR o SORTL-SVLR y el modo de fusión (MM) es cero, el segundo operando se designará en un límite de dos palabras; de lo contrario, se reconocerá una excepción de especificación, en un ejemplo. La dirección inicial y la longitud de cada lista de salida, denominadas delineaciones de listas de salida (OLD), se almacenan en la ubicación de segundo operando (por ejemplo, comenzando en la dirección especificada mediante R_2) cuando MM es cero. Cuando se especifica la función SORTL-QAF, o MM es uno, se ignora el contenido de los registros generales R_2 y $R_2 + 1$.

En la ejecución, en una realización, se realiza una función especificada por el código de función en el registro general 0. Como parte de la operación cuando la función especificada es SORTL-SFLR o SORTL-SVLR, ocurre lo siguiente, en una realización:

La dirección en el registro general R_1 se incrementa en el número de bytes almacenados en la ubicación de primer operando, y la longitud en el registro general $R_1 + 1$ se reduce en el mismo número.

Cuando MM es cero, la dirección en el registro general R_2 se incrementa en el número de bytes almacenados en la ubicación de segundo operando, y la longitud en el registro general $R_2 + 1$ se reduce en el mismo número.

En un ejemplo, la formación y actualización de las direcciones y longitudes dependen del modo de direccionamiento.

En una realización, en el modo de direccionamiento de 24 bits, se aplica lo siguiente:

El contenido de las posiciones de bits 40-63 de los registros generales 1, R_1 y R_2 constituye las direcciones del bloque de parámetros, el primer operando y el segundo operando, respectivamente, y se ignora el contenido de las posiciones de bits 0-39.

Los bits 40-63 de las direcciones actualizadas del primer operando y del segundo operando sustituyen a los bits correspondientes en los registros generales R_1 y R_2, respectivamente. Las salidas de la posición de bits 40 de las direcciones actualizadas se ignoran, y el contenido de las posiciones de bits 32-39 de los registros generales R_1 y R_2 se establece en ceros. El contenido de las posiciones de bits 0-31 de los registros generales R_1 y R_2 permanece sin cambios.

El contenido de las posiciones de bits 32-63 de los registros generales $R_1 + 1$ y $R_2 + 1$ forma números enteros binarios sin signo de 32 bits que especifican el número de bytes en los operandos primero y segundo, respectivamente. El contenido de las posiciones de bits 0-31 de los registros generales $R_1 + 1$ y $R_2 + 1$ se ignora.

Los bits 32-63 de las longitudes actualizadas del primer operando y del segundo operando sustituyen a los bits correspondientes en los registros generales $R_1 + 1$ y $R_2 + 1$, respectivamente. El contenido de las posiciones de bits 0-31 de los registros generales $R_1 + 1$ y $R_2 + 1$ permanece sin cambios.

En una realización se aplica lo siguiente en el modo de direccionamiento de 31 bits:

El contenido de las posiciones de bits 33-63 de los registros generales 1, R_1 y R_2 constituye las direcciones del bloque de parámetros, el primer operando y el segundo operando, respectivamente, y se ignora el contenido de las posiciones de bits 0-32.

Los bits 33-63 de las direcciones actualizadas de primer operando y segundo operando sustituyen a los bits correspondientes en los registros generales R_1 y R_2, respectivamente. Las salidas de la posición de bits 33 de las direcciones actualizadas se ignoran, y el contenido de la posición de bits 32 de los registros generales R_1 y R_2 se establece en cero. El contenido de las posiciones de bits 0-31 de los registros generales R_1 y R_2 permanece sin cambios.

El contenido de las posiciones de bits 32-63 de los registros generales $R_1 + 1$ y $R_2 + 1$ forma números enteros binarios sin signo de 32 bits que especifican el número de bytes en los operandos primero y segundo, respectivamente. El contenido de las posiciones de bits 0-31 de los registros generales $R_1 + 1$ y $R_2 + 1$ se ignora.

Los bits 32-63 de las longitudes actualizadas del primer operando y del segundo operando sustituyen a los bits correspondientes en los registros generales $R_1 + 1$ y $R_2 + 1$, respectivamente. El contenido de las posiciones de bits 0-31 de los registros generales $R_1 + 1$ y $R_2 + 1$ permanece sin cambios.

En una realización, en el modo de direccionamiento de 64 bits, se aplica lo siguiente:

El contenido de las posiciones de bits 0-63 de los registros generales 1, R_1 y R_2 constituye las direcciones del bloque de parámetros, el primer operando, el segundo operando, respectivamente.

Los bits 0-63 de las direcciones actualizadas del primer operando y del segundo operando sustituyen a los bits correspondientes en los registros generales R_1 y R_2, respectivamente. Las salidas de bits de la posición 0 de las direcciones actualizadas se ignoran. El contenido de las posiciones de bits 0-63 de los registros generales $R_1 + 1$ y $R_2 + 1$ forma números enteros binarios sin signo de 64 bits que especifican el número de bytes en los operandos primero y segundo, respectivamente.

Los bits 0-63 de las longitudes actualizadas del primer operando y del segundo operando sustituyen a los bits correspondientes en los registros generales $R_1 + 1$ y $R_2 + 1$, respectivamente.

En el modo de registro de acceso, los registros de acceso 1, R_1 y R_2 especifican los espacios de direcciones que contienen el bloque de parámetros, el primer operando y el segundo operando, respectivamente.

A continuación se describen más detalles sobre las diversas funciones:

Código de función 0: SORTL-QAF (Consultar Funciones Disponibles)

La función SORTL-QAF (consultar) proporciona un mecanismo para indicar la disponibilidad de todas las funciones instaladas y los formatos de bloques de parámetros instalados y los tamaños de interfaz disponibles. El tamaño de interfaz es el número de listas de entrada disponibles para el programa. El tamaño del bloque de parámetros de las

funciones SORT-SFLR y SORT-SVLR es proporcional al tamaño de la interfaz especificado por el programa.

Un formato ejemplar del bloque de parámetros para la función SORTL-QAF se describe con referencia a la FIG. 3I. En un ejemplo, un bloque de parámetros 340 para la función SORTL-QAF (por ejemplo, el código de función 0) incluye un vector de funciones instaladas 342, un vector de tamaños de interfaz instalados 344 y un vector de formatos de bloques de parámetros instalados 346. En un ejemplo particular, estos vectores se almacenan en los bytes 0-15, byte 16, y los bytes 24-25, respectivamente, del bloque de parámetros. Más abajo se describe con mayor detalle cada uno de los vectores.

Como ejemplo, los bits 0-127 del vector 342 de funciones instalado corresponden a los códigos de función 0-127, respectivamente, de la instrucción Clasificar Listas. Cuando un bit es, por ejemplo, uno, se instala la función correspondiente; de lo contrario, la función no se instala.

Además, en un ejemplo, los bits 0-7 del vector de tamaños de interfaz instalado 344 indican los tamaños de interfaz disponibles para el programa. El tamaño de interfaz es el número de listas de entrada que debe especificar el programa para las funciones SORT-SFLR y SORTL-SVLR. Los bits 0-7 del vector de tamaños de interfaz instalado 344 corresponden a los siguientes tamaños de interfaz, en un ejemplo: Bits 0, 1, 5-7 reservados; bit 2 — 32 listas de entrada; bit 3 — 64 listas de entrada; y bit 4 — 128 listas de entrada. También son posibles otros ejemplos.

Cuando un bit del vector de tamaños de interfaz instalado 344 es, por ejemplo, uno, el tamaño de interfaz correspondiente está disponible para el programa. Uno o más bits pueden almacenarse como unos. Por ejemplo, un valor de 00101000 binario indica que hay listas de entrada de 32 y 128 tamaños de interfaces disponibles. En un ejemplo, los bits 0-1 y 5-7 se reservan y se almacenan como ceros. Además, en un ejemplo, el tamaño de interfaz de 32 listas de entrada está disponible cuando se instala la función de clasificación mejorada. Por lo tanto, el bit 2 se almacena como uno. También son posibles otros ejemplos.

Además de lo anterior, en un ejemplo, los bits 0-15 del vector 346 de formatos de bloques de parámetros instalados corresponden a los formatos de bloques de parámetros 0-15, respectivamente. Cuando un bit es, por ejemplo, uno, se instala el formato de bloque de parámetros correspondiente; de lo contrario, el formato de bloque de parámetros no se instala. En un ejemplo, se almacenan ceros en los bytes reservados 17-23 y 26-31 del bloque de parámetros.

La función SORT-QAF ignora el contenido de los registros generales R_1, R_2, $R_1 + 1$ y $R_2 + 1$.

Se reconoce un evento de alteración de almacenamiento PER (grabación de eventos de programa), en caso aplicable, para el bloque de parámetros. Se reconoce un evento de detección de dirección cero PER, en caso aplicable, para el bloque de parámetros.

El código de condición 0 se establece cuando se completa la ejecución de la función SORTL-QAF; en un ejemplo, los códigos de condición 1, 2 y 3 no son aplicables a la función de consulta.

Código de función 1: SORTL-SFLR (Clasificar registros de longitud fija)

En un ejemplo, un conjunto de listas de entrada se clasifica y almacena como un conjunto de listas de salida en la ubicación de primer operando. Cada lista es un conjunto de registros y, con referencia a la FIG. 3J, cada registro 350 incluye una clave 352 (por ejemplo, una clave de longitud fija) y una carga útil 354 (por ejemplo, una carga útil de longitud fija).

Los registros de las listas de entrada se clasifican en función de los valores de las claves. Los registros se pueden clasificar en orden ascendente o descendente, como se especifica en un campo de orden de clasificación (SO) del bloque de parámetros asociado con el código de función 1, que se describe a continuación. Los registros de una lista de entrada pueden o no estar listados en orden clasificado.

Los registros de una lista de salida pueden provenir de múltiples listas de entrada y se almacenan en orden clasificado. El número de listas de salida almacenadas en la ubicación de primer operando depende de los datos de entrada. En un ejemplo, cuando cada lista de entrada activa contiene registros listados en el mismo orden que se especifica en el campo SO, solo se genera una lista de salida.

Como se ha indicado anteriormente, el bit 56 del registro general 0 especifica un modo de operación, denominado modo de fusión (MM), que se aplica a la función SORTL-SFLR. Cuando el modo de fusión es, por ejemplo, cero, para cada lista de salida almacenada en la ubicación de primer operando, la correspondiente delineación de la lista de salida (OLD) se almacena en la ubicación de segundo operando. Cada OLD incluye, por ejemplo, una dirección OLD de 8 bytes, que designa la ubicación del primer registro en la lista de salida correspondiente, y una longitud de OLD de 8 bytes, que especifica la longitud, por ejemplo, en bytes, de la lista de salida correspondiente. Cuando el modo de fusión es uno, las listas de entrada se consideran previamente clasificadas. Es decir, se considera que cada lista de entrada activa contiene registros en el mismo orden que el especificado en el campo SO del bloque de parámetros.

Cuando MM es uno y cada lista de entrada está preseleccionada, el resultado almacenado en la ubicación de primer operando es una única lista de salida de registros en orden clasificado. Cuando MM es uno y cada lista de entrada no está previamente clasificada, los resultados son impredecibles.

Cuando MM es, por ejemplo, uno, el contenido de los registros generales R_2 y R_2 +1 se ignora y no se almacena

información en la ubicación de segundo operando. Cuando MM es uno, es posible que no se realicen los procedimientos utilizados para distinguir las separaciones entre las listas de salida, mejorando así potencialmente el rendimiento de la operación. Cuando MM es uno, los datos no se almacenan en un búfer de recuperación de registros de continuación, que se describe a continuación.

Para generar una lista única de registros en orden clasificado a partir de un conjunto de registros en orden aleatorio, un programa puede realizar el siguiente procedimiento, en un ejemplo:

1. Partición uniformemente el conjunto de registros entre un conjunto inicial de listas, donde cada lista contiene registros en orden aleatorio. Ejecutar la instrucción Clasificar Listas con el conjunto inicial de listas como listas de entrada y el modo de fusión igual a cero, para generar un conjunto intermedio de listas (cada una de las cuales contiene registros clasificados) y las ubicaciones y longitudes de almacenamiento de cada lista del conjunto intermedio de listas.

2. Ejecutar la instrucción Clasificar Listas con el conjunto intermedio de listas como listas de entrada y el modo de fusión igual a uno, para generar la lista final y única, que contiene los registros en orden clasificado.

Un ejemplo del SORTL-SFLR con modo de fusión igual a cero se ilustra en la FIG. 4A. Las entradas y las salidas resultantes se incluyen en el ejemplo. Como se muestra, hay tres listas de entrada 400: lista de entrada0, lista de entrada1 y lista de entrada2. Además, se representa un ejemplo de un primer operando 402 y un segundo operando 404 resultantes. En un ejemplo, hay tres listas en el primer operando 402 (FIG. 4A) y, como se muestra en el segundo operando 404, una comienza en la dirección 1000 y tiene una longitud de 18; otra comienza en la dirección 1018 y tiene una longitud de 28; y una tercera comienza en la dirección 1040 y tiene una longitud de 20.

En un ejemplo, cuando dos operaciones realizan la misma función SORTL-SFLR con el modo de fusión igual a cero en el mismo conjunto de registros de entrada sin clasificar y la única diferencia entre las dos operaciones es el número de listas de entrada utilizadas para especificar los datos de entrada, la operación con el mayor número de listas de entrada da como resultado un número menor de listas de salida. La FIG. 4B ilustra un ejemplo del uso de seis listas de entrada 450 para operar con los mismos datos de entrada que el ejemplo de la FIG. 4A, que usa tres listas de entrada. También se representa un primer operando 452 resultante con dos listas de salida, en lugar de tres, y un segundo operando 454 que proporciona delineaciones de las dos listas de salida.

Como se indica, la función SORTL-SFLR usa un bloque de parámetros, un ejemplo del cual se describe con referencia a la FIG. 3K. En el bloque de parámetros ejemplares descrito en la presente memoria se indican ubicaciones específicas dentro del bloque de parámetros para campos específicos y tamaños específicos de los campos (por ejemplo, bytes y/o bits específicos). Sin embargo, se pueden prever otras ubicaciones y/o tamaños para uno o más de los campos. Además, aunque se especifica el ajuste de un bit a un valor particular, por ejemplo, uno o cero, esto es solo un ejemplo. El bit puede establecerse en un valor diferente, tal como el valor opuesto o en otro valor, en otros ejemplos. Son posibles muchas variaciones.

En un ejemplo, un bloque de parámetros 360 para la función SORTL-SFLR incluye lo siguiente:

Número de versión de bloque de parámetros (PBVN) 362: Los bytes 0-1 del bloque de parámetros especifican la versión y el tamaño del bloque de parámetros. Los bits 0-7 del PBVN tienen el mismo formato y definición que los bits 0-7 del vector de lista de tamaños de interfaz instalado (byte 16) del bloque de parámetros de la función SORTL-QAF (consulta). Los bits 0-7 especifican el número de listas de entrada descritas en el bloque de parámetros, N_{IS}. El tamaño del bloque de parámetros, en bytes, se determina evaluando la fórmula ($576 + 16 \times N_{IS}$). Un bit de los bits 0-7 debe tener un valor de uno; de lo contrario, se reconoce una excepción general de datos de operandos. Los bits 8-11 del PBVN están reservados y deben contener ceros; de lo contrario, es posible que el programa no funcione de manera compatible en el futuro. Los bits 12-15 del PBVN contienen un entero binario sin signo que especifica el formato del bloque de parámetros. La función SORTL-QAF proporciona un mecanismo para indicar los formatos de bloques de parámetros disponibles. Cuando el modelo no soporta el tamaño o el formato del bloque de parámetros especificado, se reconoce una excepción general de datos de operando. El PBVN lo especifica el programa y no se modifica durante la ejecución de la instrucción.

Número de versión de modelo (MVN) 364: El byte 2 del bloque de parámetros es un entero binario sin signo que identifica el modelo que ejecutó la instrucción. El MVN se actualiza durante la ejecución de la instrucción, p. ej. el procesador. El valor almacenado en el MVN depende del modelo.

Cuando el indicador de continuación (CF) 368, descrito a continuación, es uno, el MVN es una entrada a la operación. Cuando CF es uno y el MVN identifica el mismo modelo que el modelo que ejecuta actualmente la instrucción, los datos del búfer de estado de continuación (CSB) 390, que se describen a continuación, pueden usarse para reanudar la operación. Cuando CF es uno y el MVN identifica un modelo diferente al modelo que ejecuta actualmente la instrucción, se puede ignorar parte o la totalidad del campo CSB.

En un ejemplo, el programa inicializa el MVN a ceros. Se espera que el programa no modifique el MVN en caso de que la instrucción se haya de volver a ejecutar con el fin de reanudar la operación; de lo contrario, los resultados son impredecibles.

Orden de clasificación (SO) 366: El bit 56 del bloque de parámetros, cuando es cero, especifica un orden de clasificación ascendente, y cuando es uno, especifica un orden de clasificación descendente. Cuando se especifica

un orden de clasificación ascendente, cada registro de una lista de salida contiene una clave que es mayor o igual que la clave del registro adyacente, por ejemplo, a la izquierda, en la misma lista de salida. Cuando se especifica un orden de clasificación descendente, cada registro de una lista de salida contiene una clave que es menor o igual que la clave del registro adyacente, por ejemplo, a la izquierda, en la misma lista de salida. El SO no se actualiza durante la ejecución de la instrucción.

Indicador de continuación (CF) 368: El bit 63 del bloque de parámetros, cuando es uno, indica que la operación está parcialmente completa y el contenido del búfer de estado de continuación 390, y cuando un modo de fusión (MM) es cero, el contenido de un búfer de recuperación de registros de continuación puede usarse para reanudar la operación. El programa debe inicializar el indicador de continuación (CF) a cero y no modificar el CF en caso de que la instrucción se haya de volver a ejecutar con el fin de reanudar la operación; de lo contrario, los resultados son impredecibles. El procesador, en un ejemplo, modifica la CF en caso de que la instrucción se vuelva a ejecutar.

Longitud de clave de registro 370: Los bytes 10-11 del bloque de parámetros contienen un entero binario sin signo que especifica el tamaño, en bytes, de las claves de los registros procesados durante la operación. En un ejemplo se reconoce una excepción general de datos de operando para cualquiera de las siguientes condiciones:

Se especifica un tamaño de clave de cero bytes.

Se especifica un tamaño de clave que no sea múltiplo de 8.

Se especifica un tamaño de clave superior a 4096 bytes.

La longitud de clave de registro no se actualiza durante la ejecución de la instrucción.

Longitud de carga útil de registro 372: Cuando se especifica la función SORTL-SFLR, los bytes 14-15 del bloque de parámetros contienen un entero binario sin signo que especifica el tamaño, en bytes, de las cargas útiles, en los registros procesados durante la operación. En un ejemplo se reconoce una excepción general de datos de operando para cualquiera de las siguientes condiciones:

Se especifica un tamaño de carga útil que no sea múltiplo de 8.

La suma de los tamaños de clave y de carga útil especificada supera los 4096 bytes.

Un tamaño de carga útil igual a cero es válido.

Cuando se especifica la función SORTL-SVLR, se ignora el campo de longitud de carga útil de registro del bloque de parámetros. La longitud de carga útil de registro no se actualiza durante la ejecución de la instrucción.

Intención de acceso al operando (OAI) 374: Los bits 0-1 del byte 32 del bloque de parámetros indican la intención de acceso futuro a la CPU para las listas de entrada y el primer operando. Las intenciones de acceso proporcionadas pueden usarse para modificar las políticas de instalación y reemplazo de líneas de memoria caché para las ubicaciones de almacenamiento correspondientes en diversos niveles de memoria caché de la jerarquía de almacenamiento.

Cuando el bit 0 del campo OAI es uno, las ubicaciones de almacenamiento designadas para contener datos para cualquier lista de entrada activa se referenciarán como uno o más operandos de instrucciones posteriores. Cuando el bit 0 del campo OAI es cero, las ubicaciones de almacenamiento designadas para contener datos para cualquier lista de entrada activa no se referenciarán como uno o más operandos de instrucciones posteriores.

Cuando el bit 1 del campo OAI es uno, las ubicaciones de almacenamiento designadas para contener el primer operando se referenciarán como uno o más operandos de instrucciones posteriores. Cuando el bit 1 del campo OAI es cero, las ubicaciones de almacenamiento designadas para contener el primer operando no se referenciarán como uno o más operandos de instrucciones posteriores.

No se garantiza que la CPU utilice esta información. La duración en la que se puede usar esta información no está definida, pero es finita.

Cuando la siguiente instrucción secuencial después del intento de acceso a la siguiente instrucción (NIAI) es Clasificar Listas (SORTL), NIAI no efectúa la ejecución de SORTL.

El OAI no se actualiza durante la ejecución de la instrucción.

Código de recuento de listas de entrada activas (AILCC) 376: Los bits 1-7 del byte 33 del bloque de parámetros son un entero sin signo de 7 bits que especifica el número de la lista de entrada que indica el límite entre las listas de entrada activas e inactivas. Las listas de entrada con números de lista, por ejemplo, inferiores o iguales al valor del campo AILCC, están en estado activo. Las listas de entrada con números de lista, por ejemplo, superiores al valor del campo AILCC, están en estado inactivo. El número de listas de entrada en estado activo es uno más que el valor del campo AILCC.

Las listas de entrada en estado activo participan en la operación. Las listas de entrada en estado inactivo no participan en la operación.

El Bit 0 de Byte 33 del bloque de parámetros se reservado y debe contener cero; de lo contrario, es posible que el programa no funcione de manera compatible en el futuro.

Cuando el valor del campo AILCC más uno es mayor que el número de listas de entrada descritas en el bloque de parámetros, según lo especificado por los bits 0-7 del campo PBVN, se reconoce una excepción general de datos de operandos, en un ejemplo.

El valor especificado en el campo AILCC no afecta al tamaño del bloque de parámetros. Las excepciones de acceso se aplican a las referencias a campos del bloque de parámetros que especifican una dirección o longitud de lista de entrada correspondiente a una lista de entrada en estado inactivo.

El AILCC no se actualiza durante la ejecución de la instrucción.

Control de listas de entrada vacías (EILCL) 378: Cuando el bit 0 del byte 40 del bloque de parámetros es uno, la operación finaliza cuando la longitud de la lista de entrada 0 pasa a ser cero durante la operación. Cuando el bit 0 del byte 40 del bloque de parámetros es cero, la operación continúa procediendo cuando la longitud de la lista de entrada 0 pasa a ser cero durante la operación. Cuando el bit 1 del byte 40 del bloque de parámetros es uno, la operación finaliza cuando la longitud de una lista de entrada activa, distinta a la lista de entrada 0, pasa a ser cero durante la operación. Cuando el bit 1 del byte 40 del bloque de parámetros es cero, la operación continúa procediendo cuando la longitud de una lista de entrada activa, distinta a la lista de entrada 0, pasa a ser cero durante la operación.

Cuando la longitud de una lista de entrada activa es inicialmente cero antes de la ejecución de la instrucción, no se aplica el bit correspondiente de la EILCL.

El EILCL no se actualiza durante la ejecución de la instrucción.

Se espera que el programa no modifique el EILCL en caso de que la instrucción se haya de volver a ejecutar con el fin de reanudar la operación; de lo contrario, los resultados son impredecibles.

Indicador de lista de entrada vacía (EILF) 380: Cuando la EILCL es 11 binario, y la operación finaliza debido a que la longitud actualizada de una lista de entrada activa es igual a cero, y se establece el código de condición 2, el procesador almacena el valor de uno, por ejemplo, en el bit 2, del byte 40, del bloque de parámetros; de lo contrario, el valor de cero se almacena en el bit 2, del byte 40, del bloque de parámetros. Cuando la EILF contiene un valor de uno, el número de lista de entrada de la lista de entrada que quedó vacía durante la operación se coloca en el campo EILN del bloque de parámetros. En un ejemplo, el programa inicializa el EILF a ceros.

Se puede hacer referencia a la EILF al comienzo de la ejecución de la instrucción cuando se reanuda la operación. Se espera que el programa no modifique el EILF en caso de que la instrucción se haya de volver a ejecutar con el fin de reanudar la operación; de lo contrario, los resultados son impredecibles.

Número de lista de entrada vacía (EILN) 382: Cuando las condiciones hacen que se almacene un valor de uno en el campo EILF, el número de lista de entrada de la lista de entrada que quedó vacía durante la operación es almacenado, por ejemplo, por el procesador, en el byte 41 del bloque de parámetros; de lo contrario, el valor de cero se almacena en el byte 41 del bloque de parámetros.

El EILN se ignora al principio de la operación. En un ejemplo, el programa inicializa el EILN a ceros.

Indicador de lista de entrada incompleta (IILF) 384: Cuando la operación finaliza como resultado de intentar procesar una lista de entrada incompleta, el procesador almacena el valor de uno, por ejemplo, en el bit 0, del byte 46, del bloque de parámetros; de lo contrario, el valor de cero se almacena en el bit 0, del byte 46, del bloque de parámetros. Una lista de entrada activa se considera incompleta cuando la longitud de la lista de entrada correspondiente es mayor que cero e inferior al número de bytes del registro designado por la dirección de la lista de entrada. Esta condición puede existir al comienzo de la operación o puede presentarse durante la operación. Cuando el IILF contiene un valor de uno, el número de lista de entrada, de la lista de entrada incompleta encontrada, se coloca en el campo IILN del bloque de parámetros. En un ejemplo, el programa inicializa el IILF a cero.

Cuando la operación finaliza con el código de condición de configuración 2 y el valor resultante en el campo IILF es cero, la operación finaliza debido a que la lista de entrada está vacía. Cuando la operación finaliza con el código de condición de configuración 2 y el valor resultante en el campo IILF es uno, la operación finaliza debido a que la lista de entrada está incompleta.

Se puede hacer referencia a la IILF al comienzo de la ejecución de la instrucción cuando se reanuda la operación. Se espera que el programa no modifique el IILF en caso de que la instrucción se haya de volver a ejecutar con el fin de reanudar la operación; de lo contrario, los resultados son impredecibles.

Número de lista de entrada incompleta (IILN) 386: Cuando las condiciones hacen que se almacene un valor de uno en el campo IILF, el número de lista de entrada, de la lista de entrada incompleta encontrada, se almacena, por ejemplo, por el procesador, en el byte 47 del bloque de parámetros; de lo contrario, el valor de cero se almacena en el byte 47 del bloque de parámetros. Cuando hay varias listas de entrada incompletas, depende del modelo qué número de lista de entrada incompleta se almacene en el campo IILN. En un ejemplo, el programa inicializa el IILN a cero.

El IILN se ignora al principio de la operación.

Origen de búfer de recuperación de registros de continuación 388: El programa proporciona un búfer de almacenamiento de 4 K bytes, denominado búfer de recuperación de registros de continuación, para que la CPU almacene y haga referencia a los datos entre dos ejecuciones de la misma instrucción Clasificar Listas, en caso de que una operación finalice y pueda reanudarse más adelante. Cincuenta y dos bits, empezando por el bit 0 del byte 56 y hasta el bit 3 del byte 62, del bloque de parámetros contienen un entero binario sin signo utilizado en la formación de la dirección de recuperación del registro de continuación, que se alinea en un límite de 4 K-bytes. La dirección de recuperación de registro de continuación es, por ejemplo, la dirección lógica del byte más a la izquierda del búfer de recuperación de registro de continuación.

En el modo de direccionamiento de 24 bits, los bits 40-51 del origen del búfer de recuperación de registros de continuación con 12 ceros adjuntos a la derecha forman la dirección de recuperación de registros de continuación. En el modo de direccionamiento de 31 bits, los bits 33-51 del origen del búfer de recuperación de registros de continuación con 12 ceros adjuntos a la derecha forman la dirección de recuperación de registros de continuación. En el modo de direccionamiento de 64 bits, los bits 0-51 del origen del búfer de recuperación de registros de continuación con 12 ceros adjuntos a la derecha forman la dirección de recuperación de registros de continuación.

En el modo de registro de acceso, el registro de acceso 1 especifica el espacio de direcciones que contiene el búfer de recuperación de registros de continuación en el almacenamiento.

Cuando el modo de fusión (MM) es cero, la operación finaliza después de almacenar uno o más registros y no se produce una finalización normal, la clave del último registro almacenado para el primer operando también se almacena en el búfer de recuperación de registros de continuación. Cuando MM es uno, se ignora el origen del búfer de recuperación de registros de continuación.

El origen del búfer de recuperación de registros de continuación no se modifica durante la ejecución de la instrucción.

Se espera que el programa no modifique el origen de búfer de recuperación de registros de continuación en caso de que la instrucción se haya de volver a ejecutar con el fin de reanudar la operación; de lo contrario, los resultados son impredecibles.

Búfer de estado de continuación (CSB) 390: Cuando las condiciones hacen que se almacene un valor de uno en el campo CF, los datos de estado interno se almacenan, p. ej. por el procesador, en los bytes 64-575 del bloque de parámetros; de lo contrario, los bytes 64-575 del bloque de parámetros no están definidos y pueden modificarse. Los datos de estado interno almacenados dependen del modelo y pueden usarse posteriormente para reanudar la operación cuando se vuelve a ejecutar la instrucción. En un ejemplo, el programa inicializa el búfer de estado de continuación en ceros. Se espera que el programa no modifique el búfer de estado de continuación en caso de que la instrucción se haya de volver a ejecutar con el fin de reanudar la operación; de lo contrario, los resultados son impredecibles.

Como ejemplo, los datos de estado internos incluyen información relacionada con las listas de entrada, tal como información sobre comparaciones anteriores de registros de las listas de entrada para determinar las siguientes comparaciones que se realizarán. En la invención reivindicada, los datos de estado internos dependen del modelo en el sentido de que se almacenan o presentan de manera diferente según el modelo de procesador. Son posibles otras variaciones.

En una realización, la instrucción puede completarse parcialmente con un modelo en una configuración y la ejecución puede reanudarse en un modelo diferente de la configuración. Aunque diferentes modelos, en una realización, pueden mantener diferentes estados internos, en un ejemplo, cada modelo debe ser capaz de interpretar los contenidos del CSB, si los hay, que se emplean para reanudar la operación. Cuando se reanuda una operación, el MVN indica qué contenido del CSB, si lo hay, es capaz de interpretar la máquina.

Dirección de lista de entradaN 392, 394, 396: El bloque de parámetros define varias listas de entrada. El número de listas de entrada definidas en el bloque de parámetros, N_{IS}, se especifica mediante los bits 0-7 del PBVN 362. Las listas de entrada se numeran de cero a (N_{IS}-1). Para cada lista de entrada, el bloque de parámetros especifica, por ejemplo, una dirección de lista de entrada de 8 bytes. Para el número de lista de entrada N, el contenido de los bytes 576 + 16 x N a 583 + 16 x N, del bloque de parámetros, especifica, por ejemplo, la dirección lógica del byte más a la izquierda del número de lista de entrada N almacenado.

Cada dirección de lista de entrada correspondiente a una lista de entrada en el estado activo, según lo especificado por el campo AILCC, es una entrada para la operación y la operación la actualiza. La operación ignora cada dirección de lista de entrada correspondiente a una lista de entrada en estado inactivo, tal como se especifica en el campo AILCC.

Cuando una dirección de lista de entrada es una entrada para la operación, se aplica lo siguiente, en una realización:

En el modo de direccionamiento de 24 bits, los bits 40-63 de la dirección de la lista de entrada designan la ubicación del byte más a la izquierda de la lista de entrada en el almacenamiento, y el contenido de los bits 0-39 de la dirección de la lista de entrada se trata como ceros.

En el modo de direccionamiento de 31 bits, los bits 33-63 de la dirección de la lista de entrada designan la ubicación del byte más a la izquierda de la lista de entrada en el almacenamiento, y el contenido de los bits 0-32 de la dirección de la lista de entrada se trata como ceros.

En el modo de direccionamiento de 64 bits, los bits 0-63 de la dirección de la lista de entrada designan la ubicación del byte más a la izquierda de la lista de entrada en el almacenamiento.

En el modo de registro de acceso, el registro de acceso 1 especifica el espacio de direcciones que contiene las listas de entrada activas almacenadas.

Para las listas de entrada en estado activo, la dirección de lista de entrada correspondiente se designará en un límite de dos palabras; de lo contrario, se reconoce una excepción general de datos de operandos, en un ejemplo.

Cuando una dirección de lista de entrada se actualiza por la operación, se aplica lo siguiente, en una realización:

Cuando uno o más registros de la lista de entrada se han procesado como parte de la operación, la dirección de lista de entrada correspondiente se incrementa en el número de bytes que los registros procesados ocupan en el almacenamiento. La formación y actualización de la dirección de lista de entrada dependen del modo de direccionamiento.

En el modo de direccionamiento de 24 bits, los bits 40-63 de la dirección de lista de entrada actualizada sustituyen a los bits correspondientes en el campo de direcciones de lista de entrada del bloque de parámetros, se ignora la posición de bit 40 de la dirección de lista de entrada actualizada y el contenido de las posiciones de bits 0-39 del campo de direcciones de lista de entrada del bloque de parámetros se establece en ceros.

En el modo de direccionamiento de 31 bits, los bits 33-63 de la dirección de lista de entrada actualizada sustituyen a los bits correspondientes en el campo de direcciones de lista de entrada del bloque de parámetros, se ignora la posición de bit 33 de la dirección de lista de entrada actualizada y el contenido de las posiciones de bits 0-32 del campo de direcciones de lista de entrada del bloque de parámetros se establece en ceros.

En el modo de direccionamiento de 64 bits, los bits 0-63 de la dirección de lista de entrada actualizada sustituyen a los bits correspondientes en el campo de direcciones de lista de entrada del bloque de parámetros, y se ignora la posición de bit 0 de la dirección de lista de entrada actualizada.

En los modos de direccionamiento de 24 y 31 bits, cuando finaliza la ejecución de la instrucción y la instrucción no se suprime, anula o termina, cada dirección de lista de entrada de 64 bits correspondiente a una lista de entrada activa se actualiza, incluso cuando la dirección no se incrementa.

Longitud de lista de entrada N 393, 395, 397: Para cada lista de entrada, el bloque de parámetros especifica una longitud de lista de entrada de 8 bytes. Para el número de lista de entrada N, los bytes 584 + 16 x N a 591 + 16 x N del bloque de parámetros contienen un entero sin signo que especifica el número de bytes del número de lista de entrada N.

Cada longitud de lista de entrada correspondiente a una lista de entrada en el estado activo, según lo especificado por el campo AILCC, es una entrada para la operación y la operación la actualiza. La operación ignora cada longitud de lista de entrada correspondiente a una lista de entrada en estado inactivo, tal como se especifica en el campo AILCC.

En los diversos modos de direccionamiento, el contenido de las posiciones de bits 0-63 de un campo de longitud de lista de entrada especifica la longitud de la lista de entrada correspondiente.

Cuando uno o más registros de una lista de entrada se han procesado como parte de la operación, la longitud de lista de entrada correspondiente se decrementa en el número de bytes que los registros procesados ocupan en el almacenamiento. En los diversos modos de direccionamiento, los bits 0-63 de una longitud de lista de entrada actualizada sustituyen a los bits 0-63 en el campo de longitud de lista de entrada correspondiente del bloque de parámetros.

Reservado: Hay varios campos reservados en el bloque de parámetros (por ejemplo, los campos que no incluyen otra información). Como entrada a la operación, los campos reservados deben contener ceros; de lo contrario, es posible que el programa no funcione de manera compatible en el futuro. Cuando finaliza una operación, los campos reservados pueden almacenarse como ceros o pueden permanecer sin cambios.

Las FIGS. 5A-5B resumen un ejemplo de los valores originales y finales para las entradas a la función SORTL-SFLR. que incluyen campos en el bloque de parámetros.

En un realización, no es necesario, ni se espera, que el programa modifique el bloque de parámetros entre la finalización de la operación con el código de condición 3 establecido y la bifurcación de nuevo a la instrucción, para volver a ejecutar la instrucción, con el fin de reanudar la operación.

En una realización, la función SORTL-SFLR incluye múltiples comparaciones entre claves de registros de diferentes listas de entrada. Al comparar claves, se aplica lo siguiente, en un ejemplo:

Las claves se tratan como enteros binarios sin signo, también denominados datos no estructurados.

Puede que no sea necesario acceder a todos los bytes de cada clave que se está comparando para determinar qué clave contiene el valor más bajo o más alto. El número de bytes de cada clave comparados a la vez, denominado unidad de comparación de claves, depende del modelo. El número de bytes de una clave a los que se accede es un número entero de unidades de comparación de claves.

Al comparar claves de igual valor, en un ejemplo, se selecciona la clave de la lista de entrada con el número más alto de la lista de entrada para que esté en orden de clasificación antes que otras claves con el mismo valor. En este caso, el registro correspondiente de la lista de entrada con el número de lista de entrada más alto se almacena en el primer operando antes que otros registros con el mismo valor de clave. Esto se aplica a los órdenes de clasificación ascendentes y descendentes.

Una implementación puede mantener un historial de comparaciones previas entre registros de las listas de entrada activas. Cuando el historial esté disponible y sea aplicable, en lugar de acceder y comparar registros que se compararon anteriormente, se puede hacer referencia al historial. Las referencias al historial reducen el tiempo de ejecución necesario para generar resultados, lo que mejora el procesamiento en el entorno informático.

La función SORTL-SFLR incluye seleccionar registros de un conjunto de listas de entrada, en el orden de clasificación especificado, y colocar los registros seleccionados en la ubicación de primer operando. A medida que avanza la operación, se mantienen los valores actuales de la dirección de primer operando y las direcciones de las listas de entrada activas. La función procede en unidades de operación. Durante cada unidad de operación, para cada lista de entrada activa, se examina la clave designada por la dirección de la lista de entrada actual correspondiente y se coloca un registro en la ubicación de primer operando.

Cuando el modo de fusión (MM) es cero, las listas de entrada activas designan listas, cada una de las cuales se trata como si contuviera registros, de, por ejemplo, izquierda a derecha, en orden aleatorio. Cuando MM es cero, los registros almacenados en la ubicación de primer operando constituyen una o más listas de salida, y la dirección inicial y la longitud de cada lista de salida se almacenan en la ubicación de segundo operando. Cuando MM es cero, cada unidad de operación incluye las siguientes etapas, en el orden especificado, a modo de ejemplo:

1. Determinar si el siguiente registro que se almacenará en la ubicación de primer operando puede incluirse en la lista de salida más reciente (la lista de salida que incluye el registro almacenado más recientemente en la ubicación de primer operando), de la siguiente manera:

Cuando el indicador de continuación (CF) es cero y se está procesando la primera unidad de operación, no se ha almacenado ningún registro en la ubicación de primer operando, y el siguiente registro a almacenar será el primer registro de una lista de salida.

Cuando CF es uno, la ejecución anterior de la instrucción finalizó con el código de condición 1 y la primera unidad de operación se está procesando para la ejecución actual de la instrucción, el siguiente registro a almacenar será el primer registro de una lista de salida.

Cuando CF es uno, IILF es cero, EILF es cero, la ejecución anterior de la instrucción finalizó con el código de condición 2 y la primera unidad de operación se está procesando para la ejecución actual de la instrucción, el siguiente registro a almacenar será el primer registro de una lista de salida.

Cuando CF es uno, IILF o EILF es uno, la ejecución anterior de la instrucción finalizó con el código de condición 2 y la primera unidad de operación se está procesando para la ejecución actual de la instrucción, el siguiente registro a almacenar puede incluirse en la lista de salida más reciente.

Cuando CF es uno, la ejecución anterior de la instrucción finalizó con el código de condición 3 y la primera unidad de operación se está procesando para la ejecución actual de la instrucción, el siguiente registro a almacenar puede incluirse en la lista de salida más reciente.

Cuando la unidad de operación que se está procesando no es la primera unidad de operación para la ejecución actual de la instrucción, el siguiente registro a almacenar puede incluirse en la lista de salida más reciente.

2. Cuando el siguiente registro a almacenar pueda incluirse en la lista de salida más reciente, determinar el conjunto de registros que cumplen los requisitos para incluirse en la lista de salida más reciente. Para cada lista de entrada que esté activa, no vacía ni incompleta, comparar la clave del registro designado por la dirección de la lista de entrada actual (clave de entrada actual) con la clave del registro almacenado más recientemente en la ubicación de primer operando (clave almacenada anteriormente). Para este propósito, la referencia a la clave previamente almacenada no es una referencia a la ubicación de primer operando. En cambio, es una referencia a la lista de entrada de la que se seleccionó la clave o es una referencia al búfer de recuperación de registros de continuación. Es una referencia al búfer de recuperación de registros de continuación cuando se reanuda la operación y la ejecución actual de la instrucción aún no ha colocado ningún registro en la ubicación de primer operando.

Cuando el orden de clasificación es ascendente y el valor de la clave de entrada actual es mayor o igual al valor de la clave almacenada anteriormente, considerar que la clave de entrada actual pertenece a un conjunto de claves que pueden incluirse en la lista de salida más reciente. Cuando el orden de clasificación es descendente y el valor de la clave de entrada actual es menor o igual al valor de la clave almacenada anteriormente, considerar que la clave de entrada actual pertenece a un conjunto de claves que pueden incluirse en la lista de salida más reciente. Cuando el

número de claves del conjunto de claves que pueden incluirse en la lista de resultados más reciente sea cero, el siguiente registro a almacenar será el primer registro de una lista de resultados. Cuando el número de claves del conjunto de claves que pueden incluirse en la lista de resultados más reciente no sea cero, el siguiente registro a almacenar se incluirá en la lista de salida más reciente.

3. Cuando el siguiente registro a almacenar se incluirá en la lista de salida más reciente, comparar las claves en el conjunto de claves que cualifican para la inclusión en la lista de salida más reciente. Cuando el orden de clasificación es ascendente, seleccionar el valor clave más pequeño y el registro correspondiente. Cuando el orden de clasificación es descendente, seleccionar el valor clave más grande y el registro correspondiente.

4. Cuando el siguiente registro a almacenar es el primer registro de una lista de salida, comparar las claves de los registros designados por las direcciones de la lista de entrada actual correspondientes a las listas de entrada que están activas, no vacías ni incompletas. Cuando el orden de clasificación es ascendente, seleccionar el valor clave más pequeño y el registro correspondiente. Cuando el orden de clasificación es descendente, seleccionar el valor clave más grande y el registro correspondiente.

5. El registro seleccionado se coloca en la ubicación actual de primer operando.

6. La dirección actual de primer operando se incrementa en un número de bytes igual a la longitud del registro seleccionado.

7. La dirección de lista de entrada actual, correspondiente a la lista de entrada que contiene el registro seleccionado, se incrementa en un número de bytes igual a la longitud del registro seleccionado.

Como parte de la operación cuando el modo de fusión es cero, para cada lista de salida almacenada en la ubicación de primer operando, la correspondiente delineación de la lista de salida (OLD) se almacena en la ubicación de segundo operando. Cada OLD incluye, por ejemplo, una dirección de OLD de 8 bytes, que designa la ubicación del primer registro en la lista de salida correspondiente, y, por ejemplo, una longitud de OLD de 8 bytes, que especifica la longitud en bytes, de la lista de salida correspondiente. Cuando la operación finaliza con el código de condición 3, el código de condición 2 y el EILF iguales a uno, o el código de condición 2 y el IILF iguales a uno, la lista de salida más reciente que se está procesando al final de la operación puede procesarse parcialmente y no procesarse completamente. Es decir, el número de registros en la lista de salida parcialmente procesada es un valor intermedio y se puede aumentar cuando se reanude la operación. En este caso, una delineación de lista de salida (OLD), correspondiente a la lista de salida parcialmente procesada, no se coloca en la ubicación de segundo operando, hasta después de que se reanude la operación y la lista de salida se procese por completo.

Cuando el modo de fusión es cero y la operación finaliza después de almacenar uno o más registros y no se produce una finalización normal, la clave del último registro almacenado para la ubicación de primer operando también se almacena en el búfer de recuperación de registros de continuación.

Cuando el modo de fusión es cero y la operación finaliza debido a la finalización normal, se ha colocado una o más listas de salida en la ubicación de primer operando y las delineaciones de la lista de salida se han colocado en la ubicación de segundo operando. El programa puede usar delineaciones de listas de salida como valores de longitud y dirección de lista de entrada en un bloque de parámetros para una operación SORTL posterior.

Las FIGS. 6A-6D ilustran los operandos primero y segundo, antes y después de ejecutar SORTL-SFLR con el modo de fusión igual a cero. Haciendo referencia a las FIGS. 6A-6B, la FOSA 600 es la dirección de inicio de primer operando: ubicación especificada por R_1; FOEA 602 es la dirección de finalización de primer operando: ubicación especificada por $R_1 + (R_1 + 1) — 1$; y OL 604 es la lista de salida (p. ej., lista de salida 1... lista de salida N). Además, haciendo referencia a las FIGS. 6C-6D, SOSA 610 es la dirección de inicio de segundo operando: ubicación especificada por el R_2; SOEA 612 es la dirección final de segundo operando: ubicación especificada por $R_2 + (R_2 + 1) — 1$; y OLD 614 es la designación de lista de salida (por ejemplo, designación de lista de salida 1... designación de lista de salida N).

Cuando el modo de fusión (MM) es uno, las listas de entrada activas designan listas, cada una de las cuales se considera que contiene registros, de izquierda a derecha, en el orden clasificado, según lo especificado en el campo SO del bloque de parámetros. Cuando MM es uno, los registros almacenados en la ubicación de primer operando constituyen una lista de salida única. Cuando MM es uno, cada unidad de operación incluye, por ejemplo, las siguientes etapas, en el orden especificado, como un ejemplo:

1. Comparar las claves de los registros designados por las direcciones de la lista de entrada actual correspondientes a las listas de entrada que están activas, no vacías ni incompletas. Cuando el orden de clasificación es ascendente, seleccionar el valor clave más pequeño y el registro correspondiente. Cuando el orden de clasificación es descendente, seleccionar el valor clave más grande y el registro correspondiente.

2. El registro seleccionado se coloca en la ubicación actual de primer operando.

3. La dirección actual de primer operando se incrementa en un número de bytes igual a la longitud del registro seleccionado.

4. La dirección de lista de entrada actual, correspondiente a la lista de entrada que contiene el registro seleccionado, se incrementa en un número de bytes igual a la longitud del registro seleccionado.

Las FIGS. 7A-7B ilustran el primer operando, antes y después de ejecutar SORTL-SFLR con un modo de fusión igual a uno. Haciendo referencia a las FIGS. 7A-7B, FOSA 700 es la dirección de inicio de primer operando: ubicación especificada por R_1; FOEA 702 es la dirección de finalización de primer operando: ubicación especificada por $R_1 + (R_1 + 1) — 1$; y OL 704 es la lista de salida (p. ej., lista de salida 1).

Como parte de la operación, cuando el modo de fusión es cero o uno, se actualizan las direcciones y longitudes de las listas de entrada en estado activo. Para cada lista de entrada en estado activo, la dirección de lista de entrada se incrementa en el número de bytes de los registros de la lista de entrada que se seleccionaron y colocaron en la ubicación de primer operando durante la operación, y la longitud de la lista de entrada se reduce en el mismo número. La formación y actualización de las direcciones de listas de entrada dependen del modo de direccionamiento.

A medida que avanza la operación, puede encontrarse una lista de entrada incompleta. Se reconoce una lista de entrada incompleta durante una unidad de operación que intenta hacer referencia a un registro de una lista de entrada que está incompleta. Se pueden completar varias unidades de operación antes de reconocer una lista de entrada incompleta. Esto se aplica cuando el modo de fusión es cero o uno.

A medida que avanza la operación, se puede encontrar una excepción de acceso para acceder a una lista de entrada, al primer operando o al segundo operando, cuando sea aplicable. Se reconoce una excepción de acceso durante una unidad de operación que intenta acceder a una ubicación de almacenamiento y existe una excepción de acceso para esa ubicación. Se pueden completar varias unidades de operación antes de reconocer una excepción de acceso. Esto se aplica cuando el modo de fusión es cero o uno.

Cuando la operación finaliza con una finalización parcial, los datos de estado internos, que pueden incluir un historial de comparaciones anteriores entre registros, se almacenan en el campo de búfer de estado de continuación (CSB) del bloque de parámetros. Posteriormente, cuando se vuelve a ejecutar la instrucción, con el fin de reanudar la operación, el contenido del CSB se puede cargar en la implementación y se puede hacer referencia al historial cuando se reanude la operación. Esto se aplica cuando el modo de fusión es cero o uno.

La finalización normal se produce cuando los registros de las listas de entrada activas se clasifican y almacenan en el primer operando.

En una realización, cuando la operación termina debido a una finalización normal, ocurre lo siguiente:

La dirección y la longitud en los registros generales R_1 y $R_1 + 1$, respectivamente, se actualizan.

La dirección y la longitud en los registros generales R_2 y $R_2 + 1$, respectivamente, se actualizan cuando MM es cero.

Los campos de dirección lista de entradaN y longitud de lista de entradaN se actualizan para las listas de entrada en estado activo.

Se establece el número de versión del modelo.

El indicador de continuación se establece en cero.

El indicador de lista de entrada vacía se establece en cero.

El número de la lista de entrada vacía se establece en cero.

El indicador de lista de entrada incompleta se establece en cero.

El número de lista de entrada incompleta se establece en cero.

Se establece el código de condición 0.

La formación y actualización de las direcciones y longitudes dependen del modo de direccionamiento.

Cuando se produce una finalización normal, el campo CSB del bloque de parámetros no está definido una vez finalizada la operación.

En una realización, cuando se ha procesado un número de bytes determinado por la CPU, la operación finaliza y ocurre lo siguiente:

La dirección y la longitud en los registros generales R_1 y $R_1 + 1$, respectivamente, se actualizan.

La dirección y la longitud en los registros generales R_2 y $R_2 + 1$, respectivamente, se actualizan cuando MM es cero.

Los campos de dirección lista de entradaN y longitud de lista de entradaN se actualizan para las listas de entrada en estado activo.

Se establece el número de versión del modelo.

El indicador de continuación se establece en uno.

Un valor clave se almacena en el búfer de recuperación de registros de continuación cuando MM es cero y se han colocado uno o más registros en la ubicación de primer operando durante la ejecución de la instrucción.

Se actualiza el búfer de estado de continuación.

El indicador de lista de entrada vacía se establece en cero.

El número de la lista de entrada vacía se establece en cero.

El indicador de lista de entrada incompleta se establece en cero.

El número de lista de entrada incompleta se establece en cero.

Se establece el código de condición 3.

La formación y actualización de las direcciones y longitudes dependen del modo de direccionamiento.

El número de bytes determinado por la CPU depende del modelo y puede ser un número diferente cada vez que se ejecuta la instrucción. El número de bytes determinado por la CPU es normalmente distinto de cero. Aunque este número puede ser cero y aparecer como un caso sin progreso, la CPU protege contra la repetición sin fin de este caso sin progreso.

Después de que la instrucción termine con, p. ej., el código de condición 3 establecido, se espera que el programa no modifique ninguna especificación de entrada o salida de la instrucción y se ramifique para volver a ejecutar la instrucción para reanudar la operación.

Cuando el bit 0 del control de listas de entrada vacías (EILCL) es uno y la longitud de la lista de entrada 0 pasa a ser cero durante la operación y no se aplica la finalización normal, la operación finaliza y ocurre lo siguiente, en una realización:

La dirección y la longitud en los registros generales R_1 y $R_1 + 1$, respectivamente, se actualizan.

La dirección y la longitud en los registros generales R_2 y $R_2 + 1$, respectivamente, se actualizan cuando MM es cero.

Los campos de dirección lista de entradaN y longitud de lista de entradaN se actualizan para las listas de entrada en estado activo.

Se establece el número de versión del modelo.

El indicador de continuación se establece en uno.

Se puede almacenar un valor clave en el búfer de recuperación de registros de continuación cuando EILCL es 10 binarios y MM es cero. Se almacena un valor clave en el búfer de recuperación de registros de continuación cuando EILCL es 11 binario y MM es cero. En cualquier caso, se han colocado uno o más registros en la ubicación de primer operando durante la ejecución de la instrucción.

Se actualiza el búfer de estado de continuación.

Se establece el indicador de lista de entrada vacía (consúltese la FIG. 8, que representa diversos campos de bloques de parámetros cuando finaliza una operación).

Se establece el número de lista de entrada vacía (consúltese la FIG. 8).

El indicador de lista de entrada incompleta se establece en cero.

El número de lista de entrada incompleta se establece en cero.

Se establece el código de condición 2.

La formación y actualización de las direcciones y longitudes dependen del modo de direccionamiento.

Cuando el bit 1 del control de listas de entrada vacías (EILCL) es uno y la longitud de una lista de entrada activa, distinta a la lista de entrada 0, pasa a ser cero durante la operación y no se aplica la finalización normal, la operación finaliza y ocurre lo siguiente, en una realización:

La dirección y la longitud en los registros generales R_1 y $R_1 + 1$, respectivamente, se actualizan.

La dirección y la longitud en los registros generales R_2 y $R_2 + 1$, respectivamente, se actualizan cuando MM es cero.

Los campos de dirección lista de entradaN y longitud de lista de entradaN se actualizan para las listas de entrada en estado activo.

Se establece el número de versión del modelo.

El indicador de continuación se establece en uno.

Se puede almacenar un valor clave en el búfer de recuperación de registros de continuación cuando EILCL es 01 binarios y MM es cero. Se almacena un valor clave en el búfer de recuperación de registros de continuación cuando EILCL es 11 binario y MM es cero. En cualquier caso, se han colocado uno o más registros en la ubicación de primer operando durante la ejecución de la instrucción.

Se actualiza el búfer de estado de continuación.

Se establece el indicador de lista de entrada vacía (consúltese la FIG. 8).

Se establece el número de lista de entrada vacía (consúltese la FIG. 8).

El indicador de lista de entrada incompleta se establece en cero.

El número de lista de entrada incompleta se establece en cero.

Se establece el código de condición 2.

La formación y actualización de las direcciones y longitudes dependen del modo de direccionamiento.

Cuando se encuentra una lista de entrada incompleta en el estado activo, la operación finaliza y ocurre lo siguiente, en una realización:

La dirección y la longitud en los registros generales R_1 y $R_1 + 1$, respectivamente, se actualizan.

La dirección y la longitud en los registros generales R_2 y $R_2 + 1$, respectivamente, se actualizan cuando MM es cero.

Los campos de dirección lista de entradaN y longitud de lista de entradaN se actualizan para las listas de entrada en estado activo.

Se establece el número de versión del modelo.

El indicador de continuación se establece en uno.

Un valor clave se almacena en el búfer de recuperación de registros de continuación cuando MM es cero y se han colocado uno o más registros en la ubicación de primer operando durante la ejecución de la instrucción.

Se actualiza el búfer de estado de continuación.

El indicador de lista de entrada vacía se establece en cero.

El número de la lista de entrada vacía se establece en cero.

El indicador de lista de entrada incompleta (IILF) se establece en uno.

El número de lista de entrada de la lista de entrada incompleta encontrada se coloca en el campo número de lista de entrada incompleta (IILN) del bloque de parámetros.

Se establece el código de condición 2.

La formación y actualización de las direcciones y longitudes dependen del modo de direccionamiento.

Cuando la longitud de primer operando es insuficiente para almacenar otro registro, la operación finaliza y ocurre lo siguiente, en una realización:

La dirección y la longitud en los registros generales R_1 y $R_1 + 1$, respectivamente, se actualizan.

La dirección y la longitud en los registros generales R_2 y $R_2 + 1$, respectivamente, se actualizan cuando MM es cero.

Los campos de dirección lista de entradaN y longitud de lista de entradaN se actualizan para las listas de entrada en estado activo.

Se establece el número de versión del modelo.

El indicador de continuación se establece en uno.

Un valor clave se puede almacenar en el búfer de recuperación de registros de continuación cuando MM es cero y se ha colocado uno o más registros en la ubicación de primer operando durante la ejecución de la instrucción.

Se actualiza el búfer de estado de continuación.

El indicador de lista de entrada vacía se establece en cero.

El número de la lista de entrada vacía se establece en cero.

El indicador de lista de entrada incompleta se establece en cero.

El número de lista de entrada incompleta se establece en cero

Se establece el código de condición 1.

La formación y actualización de las direcciones y longitudes dependen del modo de direccionamiento.

Cuando el modo de fusión (MM) es cero y la longitud de segundo operando es inferior a 16, la operación finaliza y ocurre lo siguiente, en una realización:

La dirección y la longitud en los registros generales R_1 y $R_1 + 1$, respectivamente, se actualizan.

La dirección y la longitud en los registros generales R_2 y $R_2 + 1$, respectivamente, se actualizan.

Los campos de dirección lista de entradaN y longitud de lista de entradaN se actualizan para las listas de entrada en estado activo.

Se establece el número de versión del modelo.

El indicador de continuación se establece en uno.

Un valor clave se puede almacenar en el búfer de recuperación de registros de continuación cuando uno o más registros se han colocado en la ubicación de primer operando durante la ejecución de la instrucción.

Se actualiza el búfer de estado de continuación.

El indicador de lista de entrada vacía se establece en cero.

El número de la lista de entrada vacía se establece en cero.

El indicador de lista de entrada incompleta se establece en cero.

El número de lista de entrada incompleta se establece en cero.

Se establece el código de condición 1.

La formación y actualización de las direcciones y longitudes dependen del modo de direccionamiento.

La condición de finalización de operación se denomina finalización parcial cuando la ejecución de la instrucción finaliza (no termina en supresión, anulación o terminación) y no se produce una finalización normal.

Se reconoce un evento de alteración del almacenamiento PER, cuando es aplicable, para la ubicación de primer operando, la ubicación de segundo operando, el búfer de recuperación de registros de continuación y la parte del bloque de parámetros que se almacena. Cuando se reconoce un evento de alteración de almacenamiento PER, se almacenan menos de 4k bytes adicionales en la ubicación del operando que se cruza con el área de almacenamiento PER designada, antes de que se informe del evento.

Se reconoce un evento de detección de dirección PER cero, cuando es aplicable, para la ubicación de segundo operando y para el bloque de parámetros. La detección de cero direcciones no se aplica a las direcciones de lista de entrada ni al origen de búfer de recuperación de registros de continuación, que se especifican en el bloque de parámetros.

Consúltese Otras condiciones a continuación para obtener descripciones de ejemplos de otras condiciones que se aplican a la función SORTL-SFLR.

Cuando la instrucción finaliza con el código de condición 1, el programa puede modificar la dirección de primer operando, la longitud de primer operando, la dirección de segundo operando, la longitud de segundo operando, cualquier dirección de lista de entrada activa y cualquier longitud de lista de entrada activa, según corresponda, y posteriormente, reanudar la operación.

Cuando la instrucción finaliza con el código de condición 2, IILF igual a cero y EILF igual a cero, el programa puede modificar la dirección de primer operando, la longitud de primer operando, la dirección de segundo operando, la longitud de segundo operando, cualquier dirección de la lista de entrada activa y cualquier longitud de la lista de entrada activa, según corresponda, y posteriormente, reanudar la operación.

Cuando la instrucción finaliza con el código de condición 2 y el EILF igual a uno, el programa puede modificar la dirección y la longitud de la lista de entrada para la lista de entrada especificada por el EILN, según corresponda, y, posteriormente, reanudar la operación. En este caso, el programa también puede modificar la dirección de primer operando y la longitud de primer operando cuando el modo de fusión (MM) es uno.

Cuando la instrucción finaliza con el código de condición 2 y el IILF igual a uno, el programa puede modificar la dirección y la longitud de la lista de entrada para la lista de entrada especificada por el IILF, según corresponda, y, posteriormente, reanudar la operación. En este caso, el programa también puede modificar la dirección de primer operando y la longitud de primer operando cuando el modo de fusión (MM) es uno.

Cuando la instrucción finaliza con el código de condición 3, y antes de volver a ejecutar la instrucción para reanudar la operación, el programa modifica cualquier dirección o longitud de la lista de entrada activa, la dirección o longitud de primer operando o la dirección o longitud de segundo operando, los resultados son impredecibles.

Código de función 2: SORTL-SVLR (Clasificar registros de longitud variable)

La función SORTL-SVLR funciona igual que la función SORTL-SFLR, excepto en lo siguiente:

Los registros incluyen, por ejemplo, como se muestra en la FIG. 9, una clave de longitud fija 900, una longitud de carga útil (PL) 902 de 8 bytes y una carga útil de longitud variable 904. Por lo tanto, los registros tienen una longitud variable.

Los bytes 14-15 del bloque de parámetros de la función SORTL-SVLR se ignoran.

Los bytes menos significativos, por ejemplo, 2 del campo de longitud de carga útil de cada registro contienen un entero binario sin signo que especifica la longitud, en bytes, de la carga útil en el mismo registro. Una longitud de carga útil igual a cero es válida. La longitud de carga útil debe ser un múltiplo de, por ejemplo, 8; de lo contrario, se reconoce una excepción general de datos de operandos, en un ejemplo. Los 6 bytes más significativos (como ejemplo) del campo de longitud de carga útil están reservados y deben contener ceros; de lo contrario, es posible que el programa no funcione de manera compatible en el futuro. La suma de la longitud de clave, ocho y la longitud de carga útil no debe ser mayor que, por ejemplo, 4096; de lo contrario, se reconoce una excepción general de datos de operandos, en un ejemplo. Cuando se reconoce una excepción general de datos de operandos como resultado de una longitud de carga útil inapropiada, la dirección de la lista de entrada correspondiente a la lista de entrada activa que encuentra la excepción especifica la dirección lógica del byte más a la izquierda del registro errante. Cuando se almacena un registro de longitud variable en la ubicación de primer operando, los bytes reservados del campo de longitud de carga útil no se modifican.

Es posible que no se reconozca una lista de entrada incompleta durante una unidad de operación que solo intente hacer referencia a la clave de un registro desde una lista de entrada con una longitud de lista de entrada mayor que el tamaño de clave y menor que el tamaño de registro. En este caso, la lista de entrada incompleta se reconocerá al intentar almacenar el registro de la lista de entrada incompleta en la ubicación de primer operando.

El bloque de parámetros para la función SORTL-SVLR es el mismo que el bloque de parámetros para la función SORTL-SFLR, excepto para los bytes 14-15, como se ha indicado anteriormente.

Consúltese Otras condiciones a continuación para obtener descripciones de otras condiciones que se aplican a la función SORTL-SVLR.

Condiciones especiales

Se reconoce una excepción de especificación cuando se intenta ejecutar Clasificar Listas y, en una realización, se aplica cualquiera de las siguientes condiciones:

Los bits 57-63 del registro general 0 designan un código de función no asignado o desinstalado.

El campo R_1 designa un registro numerado impar o un registro general 0.

El campo R_2 designa un registro numerado impar o un registro general 0. Esto se aplica cuando el modo de fusión (MM) es cero o uno.

El bloque de parámetros no se designa en un límite de dos palabras.

Se especifica la función SORTL-SFLR o la función SORTL-SVLR y el primer operando no se designa en un límite de dos palabras.

Se especifica la función SORTL-SFLR o SORTL-SVLR y el segundo operando no se designa en un límite de dos palabras cuando MM es cero.

Se reconoce una excepción de datos de operando general cuando se intenta la ejecución de Clasificar Listas y se aplica cualquiera de las siguientes condiciones, en una realización:

Se especifica la función SORTL-SFLR o SORT-SVLR y ningún bit, o varios bits, de los bits 0-7 del número de versión del bloque de parámetros contienen un valor de uno, en cuyo caso se suprime la operación.

Se especifica la función SORTL-SFLR o SORTL-SVLR y el modelo no admite el tamaño o el formato del bloque de parámetros, tal como se especifica en el número de versión del bloque de parámetros, en cuyo caso se suprime la operación.

Se especifica la función SORTL-SFLR o SORTL-SVLR y la longitud de clave de registro especifica un tamaño de clave igual a cero, un tamaño de clave que no sea múltiplo de 8 o un tamaño de clave superior a 4096, en cuyo caso se suprime la operación.

Se especifica la función SORTL-SFLR y la longitud de carga útil del registro especifica un tamaño de carga útil que no es un múltiplo de 8, o un tamaño de carga útil, cuando se agrega al tamaño de clave, es superior a 4096, en cuyo caso se suprime la operación.

Se especifica la función SORTL-SVLR y la longitud de carga útil del registro especifica un tamaño de carga útil que no es un múltiplo de 8, o un tamaño de carga útil, cuando se agrega al tamaño de clave, es superior a 4088, en cuyo caso depende del modelo si se suprime o termina la operación.

Se especifica la función SORTL-SFLR o SORTL-SVLR y el valor del código de recuento de listas de entrada activas (AILCC) más uno es mayor que el número de listas de entrada descritas por el bloque de parámetros, en cuyo caso se suprime la operación.

Se especifica la función SORTL-SFLR o SORTL-SVLR y no se designa una dirección de lista de entrada, correspondiente a una lista de entrada activa, en un límite de dos palabras, en cuyo caso se suprime la operación.

Otras condiciones

En una realización, se aplican las siguientes condiciones:

La ejecución de la instrucción es interrumpible. Cuando se produce una interrupción, las direcciones en los registros generales R_1 y R_2, las longitudes en los registros generales $R_1 + 1$ y $R_2 + 1$ y campos específicos del bloque de parámetros se actualizan, de modo que la instrucción, cuando se vuelve a ejecutar, se reanuda en el punto de interrupción.

Las excepciones de acceso no se reconocen para las ubicaciones de más de 4 K bytes a la derecha de la ubicación designada por la dirección de primer operando. Las excepciones de acceso no se reconocen para las ubicaciones de más de 4 K bytes a la derecha de la ubicación designada por una dirección de lista de entrada.

Si debe reconocerse una excepción de acceso para el primer operando, el segundo operando o cualquier lista de entrada, el resultado es que se reconoce la excepción o se establece el código de condición 3. Si se establece el código de condición 3, la excepción se reconocerá cuando la instrucción se ejecute nuevamente para continuar procesando los mismos operandos, asumiendo que la condición de excepción aún exista.

Cuando la clave de un registro cruza el límite de una página y existen condiciones de excepción de acceso para ambas páginas, es posible que se reconozca cualquiera de las excepciones de acceso.

Cuando existen condiciones de excepción de acceso para el procesamiento de varias claves durante una sola unidad de operación, se puede reconocer cualquiera de estas condiciones.

Cuando el bloque de parámetros cruza el límite de una página y existen condiciones de excepción de acceso para ambas páginas, se reconoce la excepción de acceso de la página situada más a la izquierda.

Cuando la operación finaliza con una finalización parcial, es posible que se hayan almacenado hasta 4 K bytes de datos en ubicaciones dentro del primer operando que están en la ubicación, o a la derecha de esta, designada por la dirección de primer operando actualizada. Dichos almacenes dan como resultado la configuración de bits de cambio, cuando corresponde, y el reconocimiento de los eventos de alteración del almacenamiento PER, cuando corresponde. El almacenamiento en estas ubicaciones se repetirá cuando la instrucción se ejecute de nuevo para continuar procesando los mismos operandos.

Como se observa mediante esta CPU, otras CPU y programas de canal, las referencias al bloque de parámetros, el primer operando, el búfer de delineaciones de listas de salida y las listas de entrada en el estado activo pueden ser referencias de acceso múltiple, los accesos a estas ubicaciones de almacenamiento no son necesariamente simultáneos en bloques y la secuencia de estos accesos o referencias no está definida.

En una realización, los resultados son impredecibles cuando la función especificada es SORTL-SFLR o SORTL-SVLR y se aplica algo de lo siguiente:

El bloque de parámetros se superpone a cualquier lista de entrada activa o el primer operando.

Cualquier lista de entrada activa se superpone al primer operando.

El modo de fusión es cero y el bloque de parámetros se superpone al segundo operando o al búfer de recuperación de registros de continuación.

El modo de fusión es cero y cualquier lista de entrada activa se superpone al segundo operando o al búfer de recuperación de registros de continuación.

El modo de fusión es cero y el primer operando se superpone al segundo operando o al búfer de recuperación de

registros de continuación.

El modo de fusión es cero y el segundo operando se superpone al búfer de recuperación de registros de continuación.

Otro programa de canal o CPU almacena una clave de un registro en una lista de entrada o en el búfer de recuperación de registros de continuación.

Ejemplo de Código de Condición Resultante:

0 Finalización normal

1 La longitud de primer operando es menor que el tamaño de un registro, o el modo de fusión es cero y la longitud de segundo operando es inferior a 16 (es decir, la longitud de operando de primero o segundo es insuficiente para continuar)

2 Se encontró una lista de entrada incompleta (IILF=1) o la EILCL no es cero y la longitud de una lista de entrada pasó a ser igual a cero durante la operación (es decir, se encontró una lista de entrada incompleta o vacía)

3 Cantidad de datos procesados determinada por la CPU (es decir, finalización determinada por la CPU)

Excepciones de programa:

Acceso (búsqueda, listas de entrada; búsqueda y almacenamiento, bloque de parámetros y búfer de recuperación de registros de continuación; almacenamiento, operandos 1 y 2)

Datos con DXC (Código de Excepción de Datos) 0, operando general

Operación (si la instalación de mejora corta no está instalada)

Especificación

Restricción de transacción

La prioridad de ejecución de la instrucción Clasificar Listas se muestra a continuación. Cuando existen varias condiciones que tienen valores de prioridad que comienzan por 13, la condición reconocida es la que se encuentra primero, a medida que avanza la operación. Cuando se reanuda la operación (el indicador de continuación es uno al comienzo de la ejecución de la instrucción), se puede usar un historial de comparaciones previas entre claves en lugar de acceder inicialmente a las listas de entrada que están activas y no vacías. Como resultado, es posible que no se encuentre una excepción de acceso para acceder a una lista de entrada específica en el mismo punto de procesamiento, en comparación con cuando no se usa un historial de comparaciones anteriores. Cuando se procesan registros de longitud variable, las condiciones que son una función de la longitud de un registro pueden evaluarse parcialmente antes de que se determine la longitud de carga útil y evaluarse completamente después de que se determine la longitud de carga útil. Como resultado, la prioridad observada entre dichas condiciones puede diferir cuando se determina que una condición existe después de evaluar solo parcialmente los requisitos, en lugar de después de evaluar completamente todos los requisitos.

Prioridad de ejecución (SORTL)

1.-6. Excepciones con la misma prioridad que la prioridad de las condiciones de interrupción de programa en el caso general.

7.A Excepciones de acceso para la segunda instrucción de media palabra.

7.B Excepción de operación.

7.C Restricción de transacción.

8.A Excepción de especificación debida a un código de función no válido o un número de registro no válido.

8.B Excepción de especificación debido a que el primer operando no está designado en el límite de dos palabras.

8.C Excepción de especificación debido a que el primer operando no está designado en el límite de dos palabras.

8.D Excepción de especificación debido a que el segundo operando no está designado en el límite de dos palabras y el modo de fusión es cero.

9. Excepciones de acceso para un acceso a bytes 0-7 del bloque de parámetros.

10. Excepción general de datos de operandos debida a un valor no admitido del campo PBVN en el bloque de parámetros.

11. Excepciones de acceso para un acceso a bytes del bloque de parámetros distintos a los bytes 0-7.

12. Excepción general de datos de operandos debida a un valor no válido de un campo del bloque de parámetros distinto del PBVN.

13.A Excepciones de acceso para un acceso a una lista de entrada activa.

13.B Excepciones de acceso para un acceso al búfer de recuperación de registros de continuación cuando el modo de fusión es cero.

13.C Excepciones de acceso para un acceso al primer operando.

13.D Excepciones de acceso para acceder al segundo operando cuando el modo de fusión es cero.

13.E Código de condición 2 debido a una lista de entrada incompleta.

13.F Código de condición 1 debido a la longitud insuficiente del primer operando.

13.G Código de condición 1 debido a la longitud insuficiente del segundo operando cuando el modo de fusión es cero.

13.H Excepción general de datos de operandos debido a una longitud de carga útil no válida de un registro de longitud variable.

13.I Código de condición 2 debido a una lista de entrada vacía.

14. Código de condición 3.

Notas de programación. En una realización:

1. Los usos previstos del control de listas de entrada vacías (EILCL) son los siguientes:

EILCL (0:1)

Descripción (binaria)

00 Detenerse después de clasificar los registros de las listas de entrada activas (por ejemplo, todos los registros de todas las listas de entrada activas).

10 Detenerse después de que la lista de entrada0 (siempre activa) quede vacía.

11 Detenerse después de que cualquier lista de entrada activa quede vacía.

2. Cuando el código de recuento de listas de entrada activas (AILCC) es cero, hay, por ejemplo, solo una lista de entrada activa y los resultados almacenados en la ubicación de primer operando son los mismos que los datos obtenidos de la lista de entrada0.
3. Los modelos que implementan cachés de instrucciones y datos separadas pueden usar la caché de instrucciones para realizar referencias de búsqueda de operandos de almacenamiento a datos en listas de entrada activas.

4. Cuando un programa espera invocar Clasificar Listas varias veces con un modo de fusión igual a cero, como parte del procesamiento de un conjunto de datos grande, el programa utilizará, en un ejemplo, las listas de entrada disponibles y dividirá uniformemente los registros entre las listas de entrada. Esto reduce el número de veces que se accede a los datos al clasificar todo el conjunto de datos.

5. Después de Clasificar Listas con modo de fusión igual a cero que termina con el código de condición 0 establecido y múltiples delineaciones de listas de salida (OLD) en el segundo operando, un programa que pretenda generar una lista única de registros clasificados debe invocar otra operación Clasificar Listas con listas de entrada especificadas como las OLD resultantes de la anterior invocación Clasificar Listas. En este caso, en un ejemplo, la segunda invocación de Clasificar Listas especifica el modo de fusión igual a uno.

De manera similar, en una realización posterior a la invocación de Clasificar Listas con un modo de fusión igual a cero, tantas veces como sea necesario o deseado, para generar un conjunto completo de listas clasificadas a partir de un gran número de registros ordenados aleatoriamente, se invoca Clasificar Listas, en un ejemplo, con un modo de fusión igual a uno, tantas veces como sea necesario o deseado, para generar una única lista clasificada.

6. Para reducir el número de veces que se accede a cada registro al fusionar varias listas clasificadas en una sola lista con orden de clasificación ascendente (por ejemplo), el programa realiza el siguiente proceso, en una realización:

Determinar el número máximo, N, de listas de entrada disponibles para Clasificar Listas.

Comparar las claves del primer registro de las listas clasificadas que aún no se han fusionado en la lista única. Seleccionar las N listas que tengan los valores de primera clave más bajos.

Ejecutar Clasificar Listas con el modo de fusión (MM) igual a uno, el control de listas de entrada vacías (EILCL) equivale a 10 binarias, la lista de entrada 0 especifica solo el primer registro de la lista con el valor de primera clave más alto

de las N listas seleccionadas y las listas de entrada restantes especificando las otras N-1 listas seleccionadas.

Después de que Clasificar Listas terminen con el código de condición 2, IILF igual a cero y EILF igual a cero, repetir el proceso.

7. Después de que Clasificar Listas termina con el código de condición 1 establecido, el programa realiza las siguientes acciones, en un ejemplo, antes de volver a invocar las Clasificar Listas, para reanudar la operación:

Si la longitud de primer operando es menor que la longitud de registro más grande de los registros que se están procesando, entonces la longitud de primer operando o la dirección y longitud de primer operando deben actualizarse, según corresponda.

Si el modo de fusión (MM) es cero y la longitud de segundo operando es inferior a 16, la longitud de segundo operando o la dirección y longitud de segundo operando deben actualizarse, según corresponda.

Si la longitud de cualquier lista de entrada activa es igual a cero, entonces la dirección y longitud de la lista de entrada correspondientes pueden actualizarse para designar otra lista de registros que se incluirá en la operación de clasificación.

8. Después de que Clasificar Listas termina con el código de condición 2 establecido, el programa realiza las siguientes acciones, en un ejemplo, antes de volver a invocar Clasificar Listas, para reanudar la operación:

Si el indicador de lista de entrada incompleta (IILF) es uno, entonces la longitud de la lista de entrada o la dirección de la lista de entrada y la longitud de la lista de entrada identificada por el número de lista de entrada incompleta (IILN) deben actualizarse, según corresponda.

Si el indicador de lista de entrada vacía (EILF) es uno, entonces la longitud de la lista de entrada o la dirección de la lista de entrada y la longitud de la lista de entrada identificada por el número de lista de entrada vacía (EILN) deben actualizarse, según corresponda.

Si el IILF es cero, el EILF es cero y la longitud de la lista 0 de entrada es cero, la longitud de la lista 0 de entrada o la dirección y longitud de la lista de entrada 0 deben actualizarse, según corresponda. Además, la dirección y la longitud de la lista de entrada para las listas de entrada activas pueden actualizarse, lo que puede ser la acción apropiada si solo había un registro designado originalmente por la lista de entrada 0 y el control de listas de entrada vacías (EILCL) es 10 binario.

Si el modo de fusión (MM) es uno y la longitud de primer operando es menor que la longitud de registro más grande de los registros que se están procesando, entonces la longitud de primer operando o la dirección y longitud de primer operando deben actualizarse, según corresponda.

Si MM es cero y IILF es uno o EILF es uno, no se deben actualizar la dirección y longitud de primer operando y la dirección y longitud de segundo operando.

Si MM es cero, IILF es cero, EILF es cero, y la longitud de primer operando es menor que la longitud de registro más grande de los registros que se están procesando, entonces la longitud de primer operando o la dirección y longitud de primer operando deben actualizarse, según corresponda.

Si MM es cero, IILF es cero, EILF es cero, y la longitud de segundo operando es inferior a 16, la longitud de segundo operando o la dirección y longitud de segundo operando deben actualizarse, según corresponda.

Como se describe en la presente memoria, en un aspecto se proporciona una única instrucción (por ejemplo, una única instrucción de máquina diseñada Clasificar Listas) para realizar operaciones de clasificación y/o fusión en un procesador de uso general. En un ejemplo, un programa que implementa operaciones de clasificación y/o fusión para una base de datos y que se ejecuta en un procesador de uso general, puede reemplazar un subconjunto significativo de instrucciones primitivas para implementar las operaciones con una sola instrucción. Esta instrucción es, por ejemplo, una instrucción de hardware definida en una Arquitectura de Conjunto de Instrucciones (ISA). Como resultado de ello se reduce la complejidad del programa relacionado con las operaciones de clasificación y/o fusión. Además se mejora el rendimiento de las operaciones y, por lo tanto, del procesador.

Ventajosamente, la instrucción Clasificar Listas se ejecuta en un procesador de uso general (por ejemplo, una unidad de procesamiento central, denominada aquí procesador), en lugar de en un procesador de propósito especial, tal como una unidad de procesamiento gráfico (GPU), un motor de base de datos (DBE) u otros tipos de procesadores de propósito especial.

Aunque se describen diversos campos y registros, uno o más aspectos de la presente invención pueden usar otros campos o registros, campos o registros adicionales o menos campos o registros, u otros tamaños de campos y registros, etc. Son posibles muchas variaciones. Por ejemplo, pueden usarse registros implicados en lugar de registros o campos de la instrucción especificados explícitamente y/o pueden usarse registros o campos especificados explícitamente en lugar de campos o registros implicados. También son posibles otras variaciones.

En un ejemplo, la instrucción Clasificar Listas funciona con una gran cantidad de datos de una base de datos (por ejemplo, una base de datos comercial), como megabytes o terabytes de datos. Por lo tanto, la instrucción es

interrumpible y el procesamiento puede reanudarse cuando se interrumpe.

Cuando se interrumpe la instrucción, la operación (por ejemplo, clasificar y/o fusionar) solo se completa parcialmente. La ejecución de la instrucción finaliza con el establecimiento de un código de condición en un valor que informa al programa (por ejemplo, al programa que emite la instrucción) de la finalización parcial de la operación. El programa puede entonces volver a ejecutar la instrucción para reanudar el procesamiento.

La instrucción, en una realización, emplea un número (por ejemplo, un número significativo) de ciclos de ejecución para preparar el procesador con metadatos antes de que se produzcan los resultados. La preparación del procesador con los metadatos se realiza cada vez que se ejecuta o se vuelve a ejecutar la instrucción. Por lo tanto, según un aspecto de la presente invención, se almacenan y usan los metadatos producidos previamente. de manera que los metadatos producidos anteriormente no tengan que reproducirse en el momento en que se vuelva a ejecutar la instrucción.

En un ejemplo, los metadatos incluyen estado interno del procesador, incluyendo, por ejemplo, información sobre las listas de entrada, tal como información sobre comparaciones anteriores de registros de las listas de entrada para determinar las próximas comparaciones que se realizarán.

El procesador extrae los metadatos y los almacena en una ubicación proporcionada por el programa. Luego, cuando la instrucción se va a volver a ejecutar después de una interrupción, los metadatos se extraen de la ubicación y se cargan en el procesador, sin utilizar tareas para reproducir los metadatos. Esto ahorra el tiempo que habría sido necesario para generar los metadatos de la operación.

Se describen detalles adicionales de una realización de reanudación de la ejecución de una operación con referencia a las FIGS. 10A-10B. Por ejemplo, la FIG. 10A representa el procesamiento asociado con una instrucción parcialmente completada; y la FIG. 10B representa el procesamiento asociado con la reanudación de la ejecución de una instrucción parcialmente completada. En este ejemplo, la instrucción es una instrucción que realiza la clasificación y/o la fusión, tal como la instrucción Clasificar Listas; sin embargo, en otros ejemplos, pueden ser otras instrucciones las que se puedan interrumpir. El procesamiento de las FIGS. 10A-10B se realiza, por ejemplo, mediante un procesador (por ejemplo, el procesador 102 o 204).

Haciendo referencia inicialmente a la FIG. 10A, una instrucción que se ejecuta en un procesador y que realiza una operación, tal como clasificar y/o fusionar, termina antes de completar la operación, ETAPA 1000. Sobre la base en la finalización parcial, se establece un indicador de continuación (indicador de continuación 368 del bloque de parámetros 360), por ejemplo, en uno, que indica la finalización parcial, ETAPA 1002. Además, el estado interno del procesador, tal como para una operación de clasificación o fusión, la información relativa a las comparaciones previas de los registros de las listas de entrada, se almacena en el bloque de parámetros, por ejemplo, en el búfer de estado de continuación 390, ETAPA 1004.

A continuación, con referencia a la FIG. 10B, se ejecuta una instrucción (por ejemplo, una instrucción de clasificación/fusión), ETAPA 1050. Se determina si el indicador de continuación se configura para indicar que se trata de una reejecución de la instrucción, CONSULTA 1052. Si el indicador de continuación se configura para indicar la reanudación de una operación, ETAPA 1053, entonces se extrae el estado interno del procesador guardado, por ejemplo, en el búfer de estado de continuación 390, ETAPA 1054, y se carga en una o más ubicaciones seleccionadas dentro del procesador, ETAPA 1056. Los datos de estado internos extraídos y cargados se usan entonces en la reejecución de la instrucción, ETAPA 1058. Por ejemplo, en lugar de repetir las comparaciones realizadas anteriormente, los datos de estado internos se utilizan para determinar las siguientes comparaciones que se realizarán. reduciendo así el tiempo de ejecución y mejorando el rendimiento.

Volviendo a la CONSULTA 1052, si el indicador de continuación no se configura para indicar una nueva ejecución, entonces, en un ejemplo, la operación se inicia, no se reanuda, ETAPA 1060. Por lo tanto, en un ejemplo, la operación continúa sin usar, por ejemplo, datos de estado internos del búfer de estado de continuación (CSB) 390 de un bloque de parámetros en la memoria, por ejemplo, el bloque de parámetros 360.

Uno o más aspectos de la presente invención están inextricablemente ligados a la tecnología informática y facilitan el procesamiento dentro de un ordenador, mejorando el rendimiento del mismo. El uso de metadatos extraídos y guardados cuando una instrucción se vuelve a ejecutar después de una interrupción, en lugar de reproducir los metadatos, ahorra tiempo y mejora el rendimiento en el entorno informático.

En un ejemplo particular, la instrucción que se vuelve a ejecutar es una instrucción Clasificar Listas, que es una única instrucción de máquina diseñada para realizar la clasificación y/o la fusión de un gran número de registros de base de datos de una base de datos que reemplaza a muchas instrucciones de software, mejorando el rendimiento en el entorno informático. Los registros clasificados y/o fusionados pueden usarse en muchos campos técnicos que gestionan y/o utilizan grandes cantidades de datos, tal como en procesamiento informático, procesamiento médico, seguridad, etc. Al proporcionar optimizaciones en la clasificación/fusión, estos campos técnicos se mejoran al reducir el tiempo de ejecución para obtener información y utilizarla, y reducir los requisitos de almacenamiento.

Con referencia a las FIGS. 11A-11B, se describen detalles adicionales de una realización para facilitar el procesamiento dentro de un entorno informático, en lo que se refiere a uno o más aspectos de la presente invención.

Haciendo referencia a la FIG. 11A, en una realización, se determina que el procesamiento de una operación de una

instrucción que se ejecuta en el procesador se ha interrumpido antes de su finalización (1100). Sobre la base de la determinación de que el procesamiento de la operación se ha interrumpido, se extraen los metadatos del procesador (1102). Los metadatos son, por ejemplo, metadatos actuales del procesador (1104). Los metadatos se almacenan en una ubicación asociada con la instrucción (1106) y se usan para volver a ejecutar la instrucción para reanudar el procesamiento directo de la instrucción desde donde se interrumpió (1108).

En un ejemplo, la instrucción es una instrucción de clasificación (1110) y los metadatos incluyen información acerca de una o más listas de entrada para la instrucción de clasificación (1112). Por ejemplo, los metadatos incluyen información sobre las comparaciones anteriores realizadas de los registros de una o más listas de entrada para indicar las próximas comparaciones que se realizarán (1114). Las siguientes comparaciones a realizar se indican, por ejemplo, sin repetir las comparaciones anteriores (1116).

En un ejemplo, la determinación incluye comprobar un código de condición establecido en función de la terminación de la instrucción; el código de condición establecido en un valor seleccionado indica la finalización parcial de la instrucción (1118).

Como ejemplo, haciendo referencia a la FIG. 11B, la ubicación en la que se almacenan los metadatos es un bloque de parámetros en la memoria designado por la instrucción (1120). La ubicación del bloque de parámetros en la memoria se designa, por ejemplo, mediante el contenido de un registro implicado de la instrucción (1122). En un ejemplo particular, el bloque de parámetros incluye un búfer de estado de continuación para almacenar los metadatos (1124), los metadatos incluyen datos de estado internos del procesador (1126). Además, en un ejemplo, el bloque de parámetros incluye un indicador de continuación para indicar la finalización parcial de la operación (1128).

En un aspecto, el uso de los metadatos para volver a ejecutar la instrucción incluye además volver a ejecutar la instrucción para reanudar el procesamiento (1132); extraer los metadatos de la ubicación (1134); y cargar los metadatos extraídos de la ubicación en una o más ubicaciones seleccionadas del procesador. en donde los metadatos se proporcionan al procesador sin repetir una o más tareas para producir los metadatos (1136).

Son posibles otras variaciones y realizaciones.

Los aspectos de la presente invención pueden utilizarse por muchos tipos de entornos informáticos. Con referencia a la FIG. 12A se describe otra realización de un entorno informático para incorporar y utilizar uno o más aspectos de la presente invención. En este ejemplo, un entorno informático 10 incluye, por ejemplo, una unidad de procesamiento central (CPU) 12, una memoria 14, y uno o más dispositivos de entrada/salida y/o interfaces 16 acoplados entre sí mediante, por ejemplo, uno o más buses 18 y/u otras conexiones. Como ejemplos, el entorno informático 10 puede incluir un procesador PowerPC® ofrecido por International Business Machines Corporation, Armonk, Nueva York; un HP Superdome con procesadores Intel Itanium II ofrecidos por Hewlett Packard Co., Palo Alto, California; y/u otras máquinas basadas en arquitecturas ofrecidas por International Business Machines Corporation, Hewlett Packard, Intel Corporation, Oracle u otros. IBM, z/Architecture, IBM Z, z/OS, PR/SM y PowerPC son marcas comerciales o marcas comerciales registradas de International Business Machines Corporation en al menos una jurisdicción. Intel e Itanium son marcas comerciales o marcas comerciales registradas de Intel Corporation o sus subsidiarias en los Estados Unidos y otros países. =

La unidad de procesamiento central nativa 12 incluye uno o más registros nativos 20, tales como uno o más registros de propósito general y/o uno o más registros de propósito especial utilizados durante el procesamiento dentro del entorno. Estos registros incluyen información que representa el estado del entorno en cualquier instante particular de tiempo.

Además, la unidad de procesamiento central nativa 12 ejecuta instrucciones y código que están almacenados en la memoria 14. En un ejemplo particular, la unidad de procesamiento central ejecuta el código de emulador 22 almacenado en la memoria 14. Este código permite que el entorno informático configurado en una arquitectura emule otra arquitectura. Por ejemplo, el código de emulador 22 permite que máquinas basadas en arquitecturas diferentes de la arquitectura de hardware z/Architecture, tales como los procesadores PowerPC, los servidores HP Superdome u otros, emulen la arquitectura de hardware z/Architecture y ejecuten el software y las instrucciones desarrolladas en función de la arquitectura de hardware z/Architecture.

Con referencia a la FIG. 12B se describen detalles adicionales relacionados con el código de emulador 22. Las instrucciones de invitado 30 almacenadas en la memoria 14 comprenden instrucciones de software (por ejemplo, que se correlacionan con instrucciones de máquina) que fueron desarrolladas para ejecutarse en una arquitectura distinta a la de la CPU nativa 12. Por ejemplo, las instrucciones de invitado 30 pueden haber sido diseñadas para ejecutarse en un procesador basado en la arquitectura de hardware z/Architecture, pero en cambio, se están emulando en una CPU 12 nativa, que puede ser, por ejemplo, un procesador Intel Itanium II. En un ejemplo, el código de emulador 22 incluye una rutina de búsqueda de instrucciones 32 para obtener una o más instrucciones de invitado 30 de la memoria 14 y, opcionalmente, proporcionar almacenamiento en memoria intermedia local para las instrucciones obtenidas. También incluye una rutina de traducción de instrucciones 34 para determinar el tipo de instrucción de invitado que se ha obtenido y para traducir la instrucción de invitado en una o más instrucciones nativas 36 correspondientes. Esta traducción incluye, por ejemplo, identificar la función que realizará la instrucción de invitado y elegir la instrucción o instrucciones nativas para realizar esa función.

Además, el código de emulador 22 incluye una rutina de control de emulación 40 para provocar que se ejecuten las

instrucciones nativas. La rutina de control de emulación 40 puede hacer que la CPU 12 nativa ejecute una rutina de instrucciones nativas que emulen una o más instrucciones de invitado obtenidas previamente y, al concluir dicha ejecución, devuelva el control a la rutina de búsqueda de instrucciones para emular la obtención de la siguiente instrucción de invitado o un grupo de instrucciones de invitado. La ejecución de las instrucciones nativas 36 puede incluir cargar datos en un registro desde la memoria 14; almacenar datos de vuelta a la memoria desde un registro; o realizar algún tipo de operación aritmética o lógica, según lo determinado por la rutina de traducción.

Cada rutina se implementa, por ejemplo, en un software, que se almacena en la memoria y se ejecuta mediante la unidad de procesamiento central nativa 12. En otros ejemplos, una o más de las rutinas u operaciones se implementan en firmware, hardware, software o alguna combinación de los mismos. Los registros del procesador emulado pueden emularse utilizando los registros 20 de la CPU nativa o utilizando ubicaciones en la memoria 14. En realizaciones, las instrucciones de invitado 30, las instrucciones nativas 36 y el código de emulador 22 pueden residir en la misma memoria o pueden distribuirse entre diferentes dispositivos de memoria.

Los entornos informáticos descritos anteriormente son solo ejemplos de entornos informáticos que se pueden utilizar. Otros entornos, incluidos, pero sin limitación a esto, entornos no particionados, entornos particionados y/o entornos emulados; las realizaciones no se limitan a ningún entorno.

Cada entorno informático puede configurarse para incluir uno o más aspectos de la presente invención. Por ejemplo, cada uno puede configurarse para proporcionar un procesamiento de desbordamiento, según uno o más aspectos de la presente invención.

Uno o más aspectos pueden estar relacionados con la informática en la nube.

Debe entenderse que, aunque esta divulgación incluye una descripción detallada de la informática en la nube, la implementación de las enseñanzas mencionadas en esta memoria no se limita a un entorno informático en la nube. Más bien, las realizaciones de la presente invención son capaces de ser implementadas junto con cualquier otro tipo de entorno informático conocido ahora o desarrollado posteriormente.

La informática en la nube es un modelo de prestación de servicios para permitir un acceso de red conveniente bajo demanda a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, ancho de banda de red, servidores, procesamiento, memoria, almacenamiento, aplicaciones, máquinas virtuales y servicios) que se pueden aprovisionar y liberar rápidamente con un mínimo esfuerzo de gestión o interacción con un proveedor del servicio. Este modelo en la nube puede incluir al menos cinco características, al menos tres modelos de servicio y al menos cuatro modelos de implementación.

Las características son como sigue:

Autoservicio bajo demanda: un consumidor en la nube puede aprovisionar capacidades informáticas de forma unilateral, tales como la hora del servidor y el almacenamiento de red, según sea necesario de forma automática, sin requerir interacción humana con el proveedor del servicio.

Amplio acceso a la red: las capacidades están disponibles sobre una red y se accede a ellas a través de mecanismos estándar que promueven el uso por plataformas heterogéneas de clientes ligeros o pesados (por ejemplo, teléfonos móviles, ordenadores portátiles y PDA).

Agrupación de recursos: los recursos informáticos del proveedor se agrupan para servir a múltiples consumidores usando un modelo de múltiples inquilinos, con diferentes recursos físicos y virtuales asignados dinámicamente y reasignados según la demanda. Hay un sentido de independencia de ubicación en el sentido de que el consumidor generalmente no tiene control ni conocimiento sobre la ubicación exacta de los recursos proporcionados, pero puede ser capaz de especificar la ubicación en un nivel de abstracción superior (por ejemplo, país, estado o centro de datos).

Elasticidad rápida: las capacidades se pueden aprovisionar rápida y elásticamente, en algunos casos de manera automática, para poner a escala rápidamente y liberar rápidamente para escalar rápidamente. Para el consumidor, las capacidades disponibles para el aprovisionamiento a menudo parecen ilimitadas y pueden adquirirse en cualquier cantidad en cualquier momento.

Servicio medido: los sistemas en la nube controlan y optimizan automáticamente el uso de los recursos al aprovechar una capacidad de medición en algún nivel de abstracción apropiado para el tipo de servicio (por ejemplo, almacenamiento, procesamiento, ancho de banda y cuentas de usuario activas). El uso de recursos se puede monitorizar, controlar e informar, proporcionando transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

Los modelos de servicio son como sigue:

Software como servicio (SaaS): la capacidad que se proporciona al consumidor es para utilizar las aplicaciones del proveedor que se ejecutan en una infraestructura en la nube. Las aplicaciones son accesibles desde diversos dispositivos cliente a través de una interfaz de cliente ligero, tal como un navegador web (por ejemplo, correo electrónico basado en web). El consumidor no gestionar ni controla la infraestructura de nube subyacente que incluye la red, los servidores, los sistemas operativos, el almacenamiento o incluso las capacidades de aplicaciones individuales, con la posible excepción de ajustes limitados de configuración de aplicaciones específicas del usuario.

Plataforma como Servicio (PaaS): la capacidad proporcionada al consumidor es para implementar en la infraestructura en la nube aplicaciones creadas o adquiridas por el consumidor, creadas usando lenguajes de programación y herramientas soportadas por el proveedor. El consumidor no gestionar ni controla la infraestructura de nube subyacente que incluye las redes, los servidores, los sistemas operativos o el almacenamiento, pero tiene control sobre las aplicaciones implementadas y, posiblemente, las configuraciones del entorno del alojamiento de aplicaciones.

Infraestructura como Servicio (IaaS): la capacidad proporcionada al consumidor es para aprovisionar procesamiento, almacenamiento, redes y otros recursos informáticos fundamentales donde el consumidor es capaz de implementar y ejecutar software arbitrario, que puede incluir sistemas operativos y aplicaciones. El consumidor no gestiona ni controla la infraestructura en la nube subyacente, pero tiene control sobre los sistemas operativos, el almacenamiento, las aplicaciones implementadas y, posiblemente, un control limitado de los componentes de interconexión de redes seleccionados (por ejemplo, los firewalls del anfitrión).

Los modelos de implementación son como sigue:

Nube privada: la infraestructura en la nube se opera únicamente para una organización. Se puede gestionar por la organización o por un tercero y puede existir en las instalaciones o fuera de las instalaciones.

Nube comunitaria: la infraestructura en la nube es compartida por varias organizaciones y soporta una comunidad específica que tiene intereses compartidos (por ejemplo, misión, requisitos de seguridad, políticas y consideraciones de cumplimiento). Se puede gestionar por las organizaciones o por un tercero y puede existir en las instalaciones o fuera de las instalaciones.

Nube pública: la infraestructura en la nube se pone a disposición del público en general o de un gran grupo industrial y es propiedad de una organización que vende servicios en la nube.

Nube híbrida: la infraestructura en la nube es una composición de dos o más nubes (privada, comunitaria o pública) que siguen siendo entidades únicas, pero están unidas entre sí por una tecnología estandarizada o propietaria que permite la portabilidad de datos y aplicaciones (por ejemplo, la ráfaga en la nube para equilibrar la carga entre nubes).

Un entorno informático en la nube está orientado a servicio con un enfoque en ausencia de estado, bajo acoplamiento, modularidad e interoperabilidad semántica. En el corazón de la informática en la nube hay una infraestructura que incluye una red de nodos interconectados.

Haciendo referencia ahora a la FIG. 13, se representa el entorno de informática en la nube 50 ilustrativo. Como se muestra, el entorno informático en la nube 50 incluye uno o más nodos informáticos en la nube 52 con los que pueden comunicarse los dispositivos informáticos locales utilizados por los consumidores en la nube, tales como, por ejemplo, un asistente digital personal (PDA) o un teléfono móvil 54A, un ordenador de sobremesa 54B, un ordenador portátil 54C y/o un sistema informático de automóvil 54N. Los nodos 52 pueden comunicarse unos con otros. Se pueden agrupar (no mostrado) física o virtualmente, en una o más redes, tales como nubes privadas, comunitarias, públicas o híbridas, como se ha descrito anteriormente, o una combinación de las mismas. Esto permite que el entorno informático en la nube 50 ofrezca infraestructura, plataformas y/o software como servicios para los que un consumidor de la nube no necesita mantener recursos en un dispositivo informático local. Se entiende que los tipos de dispositivos informáticos 54A-N mostrados en la FIG. 13 se pretende que sean únicamente ilustrativos y que los nodos informáticos 52 y el entorno informático en la nube 50 pueden comunicarse con cualquier tipo de dispositivo informatizado sobre cualquier tipo de red y/o conexión direccionable de red (por ejemplo, utilizando un navegador web).

Haciendo referencia ahora a la FIG. 14, se muestra un conjunto de capas de abstracción funcionales proporcionadas por el entorno de informática en la nube 50 (FIG. 13). Se debería entender de antemano que los componentes, las capas y las funciones mostrados en la FIG. 14 se pretende que sean únicamente ilustrativos y las realizaciones de la invención no están limitadas a los mismos. Como se representa, se proporcionan las siguientes capas y funciones correspondientes:

La capa de hardware y software 60 incluye componentes de hardware y software. Ejemplos de componentes de hardware incluyen: ordenadores centrales 61; servidores basados en la arquitectura RISC (por las siglas en inglés de Reduced Instruction Set Computer - Ordenador de Conjunto de Instrucciones Reducido) 62; servidores 63; servidores blade 64; dispositivos de almacenamiento 65; y redes y componentes de interconexión de redes 66. En algunas realizaciones, los componentes de software incluyen software de servidor de aplicaciones de red 67 y el software de base de datos 68.

La capa de virtualización 70 proporciona una capa de abstracción a partir de la que se pueden proporcionar los siguientes ejemplos de entidades virtuales: servidores virtuales 71; almacenamiento virtual 72; redes virtuales 73, incluyendo redes privadas virtuales; aplicaciones virtuales y sistemas operativos 74; y clientes virtuales 75.

En un ejemplo, la capa de gestión 80 puede proporcionar las funciones descritas a continuación. El aprovisionamiento de recursos 81 proporciona una adquisición dinámica de recursos informáticos y otros recursos que se utilizan para realizar tareas dentro del entorno informático en la nube. La Medición y Fijación de precios 82 proporcionan un seguimiento de los costes a medida que se utilizan los recursos dentro del entorno informático en la nube, y el cobro o facturación por el consumo de estos recursos. En un ejemplo, estos recursos pueden incluir licencias de software de aplicaciones. La seguridad proporciona comprobación de identidad para los consumidores y las tareas en la nube,

así como protección para los datos y otros recursos. El portal de usuario 83 proporciona acceso al entorno informático en la nube para consumidores y administradores de sistemas. La gestión de nivel de servicio 84 proporciona la asignación y gestión de recursos informáticos en la nube. de manera que se cumplan los niveles de servicio requeridos. La planificación y el cumplimiento del Acuerdo de Nivel de Servicio (SLA) 85 proporcionan un acuerdo previo para recursos informáticos, y la adquisición de estos, en la nube para los que se anticipa un requisito futuro según un SLA.

La capa de cargas de trabajo 90 proporciona ejemplos de funcionalidad para lo cual se puede utilizar el entorno informático en la nube. Ejemplos de cargas de trabajo y funciones que se pueden proporcionar desde esta capa incluyen: mapeo y navegación 91; desarrollo de software y gestión del ciclo de vida 92; distribución de educación en el aula virtual 93; procesamiento de analíticas de datos 94; procesamiento de transacciones 95; y procesamiento de clasificación y/o fusión 96.

Aspectos de la presente invención puede ser un sistema, un método y/o un producto de programa informático en cualquier posible nivel de detalle técnico de integración. El producto de programa informático puede incluir un medio (o medios) de almacenamiento legible por ordenador que tiene instrucciones de programa legibles por ordenador en el mismo para hacer que un procesador lleve a cabo aspectos de la presente invención.

El soporte de almacenamiento legible por ordenador puede ser un dispositivo tangible que puede retener y almacenar instrucciones para su uso por un dispositivo de ejecución de instrucciones. El soporte de almacenamiento legible por ordenador puede ser, por ejemplo, pero no se limita a, un dispositivo de almacenamiento electrónico, un dispositivo de almacenamiento magnético, un dispositivo de almacenamiento óptico, un dispositivo de almacenamiento electromagnético, un dispositivo de almacenamiento de semiconductores o cualquier combinación adecuada de los anteriores. Una lista no exhaustiva de ejemplos más específicos del soporte de almacenamiento legible por ordenador incluye lo siguiente: un disquete de ordenador portátil, un disco duro, una memoria de acceso aleatorio (RAM), una memoria de solo lectura (ROM), una memoria de solo lectura programable y borrable (EPROM o memoria Flash), una memoria estática de acceso aleatorio (SRAM), un disco compacto de memoria de solo lectura portátil (CD-ROM), un disco versátil digital (DVD), una memoria USB, un disco flexible, un dispositivo codificado mecánicamente, tal como tarjetas perforadas o estructuras elevadas en una ranura que tiene instrucciones grabadas en las mismas, y cualquier combinación adecuada de los anteriores. Un soporte de almacenamiento legible por ordenador, como se usa en esta memoria, no debe interpretarse como señales transitorias *en sí*, tales como ondas de radio u otras ondas electromagnéticas que se propagan libremente, ondas electromagnéticas que se propagan a través de una guía de ondas u otros medios de transmisión (por ejemplo, pulsos de luz que pasan a través de un cable de fibra óptica) o señales eléctricas transmitidas a través de un cable.

Las instrucciones de programa legibles por ordenador descritas en esta memoria pueden descargarse a los respectivos dispositivos informáticos/de procesamiento desde un soporte de almacenamiento legible por ordenador o a un ordenador externo o dispositivo de almacenamiento externo a través de una red, por ejemplo, Internet, una red de área local, una red de área amplia y/o una red inalámbrica. La red puede comprender cables de transmisión de cobre, fibras de transmisión óptica, transmisión inalámbrica, enrutadores, firewalls, conmutadores, ordenadores de puerta de enlace y/o servidores periféricos. Una tarjeta adaptadora de red o interfaz de red en cada dispositivo informático/de procesamiento recibe instrucciones de programa legibles por ordenador desde la red y envía las instrucciones de programa legibles por ordenador para su almacenamiento en un soporte de almacenamiento legible por ordenador dentro del dispositivo informático/de procesamiento respectivo.

Las instrucciones de programa legibles por ordenador para llevar a cabo las operaciones de la presente invención pueden ser instrucciones de ensamblador, instrucciones de arquitectura de conjunto de instrucciones (ISA), instrucciones de máquina, instrucciones dependientes de la máquina, microcódigo, instrucciones de firmware, datos de establecimiento de estado, datos de configuración para circuitos integrados, o bien código fuente o código objeto escrito en cualquier combinación de uno o más lenguajes de programación. incluyendo un lenguaje de programación orientado a objetos como Smalltalk, C++ o similares, y lenguajes de programación procedimentales, como el lenguaje de programación "C" o lenguajes de programación similares. Las instrucciones de programa legibles por ordenador pueden ejecutarse completamente en el ordenador del usuario, parcialmente en el ordenador del usuario, como un paquete de software independiente, parcialmente en el ordenador del usuario y parcialmente en un ordenador remoto o completamente en el ordenador o servidor remoto. En este último escenario, el ordenador remoto puede conectarse al ordenador del usuario a través de cualquier tipo de red, incluyendo una red de área local (LAN) o una red de área amplia (WAN), o la conexión puede realizarse a un ordenador externo (por ejemplo, a través de Internet utilizando un Proveedor de Servicios de Internet). En algunas realizaciones, circuitos electrónicos que incluyen, por ejemplo, circuitos lógicos programables, agrupaciones de puertas programables de campo (FPGA, por sus siglas en inglés) o agrupaciones lógicas programables (PLA, por sus siglas en inglés) pueden ejecutar instrucciones de programa legibles por ordenador mediante el uso de información de estado de las instrucciones de programa legibles por ordenador para personalizar los circuitos electrónicos, con el fin de realizar aspectos de la presente invención.

Aspectos de la presente invención se describen en esta memoria con referencia a ilustraciones de diagrama de flujo y/o diagramas de bloques de métodos, aparatos (sistemas) y productos de programas informáticos según las realizaciones de la invención. 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, pueden implementarse por instrucciones de programa legibles por ordenador.

Estas instrucciones de programa legibles por ordenador pueden proporcionarse a un procesador de un ordenador de uso general, un 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 programable de procesamiento de datos, creen medios para implementar las funciones/actos especificados en el bloque o bloques del diagrama de flujo y/o diagrama de bloques. Estas instrucciones de programa legibles por ordenador también pueden almacenarse en un soporte de almacenamiento legible por ordenador que puede dirigir un ordenador, un aparato de procesamiento de datos programables y/u otros dispositivos para que funcionen de una manera particular. de manera que el soporte de almacenamiento legible por ordenador que tiene instrucciones almacenadas en el mismo comprenda un artículo de fabricación incluidas instrucciones que implementen aspectos de la función/acto especificados en el bloque o bloques del diagrama de flujo y/o diagrama de bloques.

Las instrucciones de programa legibles por ordenador también pueden cargarse en un ordenador, otro aparato de procesamiento de datos programable u otro dispositivo para provocar que se realicen una serie de acciones operativas en el ordenador, otro aparato programable u otro dispositivo para producir un proceso implementado por ordenador, de manera que las instrucciones que se ejecutan en el ordenador, otro aparato programable u otro dispositivo implementen las funciones/actos especificados en el bloque o bloques del diagrama de flujo y/o del diagrama de bloques.

El diagrama de flujo y los diagramas de bloques de las Figuras ilustran la arquitectura, la funcionalidad y la operación de posibles implementaciones de sistemas, métodos y productos de programas informáticos según diversas realizaciones de la presente invención. A este respecto, cada bloque en el diagrama de flujo o diagramas de bloques puede representar un módulo, segmento o porción de instrucciones, que comprende una o más instrucciones ejecutables para implementar la función o funciones lógicas especificadas. En algunas implementaciones alternativas, las funciones indicadas en los bloques pueden ocurrir fuera del orden indicado en las Figuras. Por ejemplo, dos bloques mostrados en sucesión pueden, de hecho, ejecutarse sustancialmente de manera simultánea, o los bloques pueden ejecutarse en ocasiones en el 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, pueden implementarse mediante sistemas basados en hardware de propósito especial que realizan las funciones o actos especificados o llevan a cabo combinaciones de hardware de propósito especial e instrucciones de ordenador.

Además de lo anterior, un proveedor de servicios que ofrece la gestión de los entornos de cliente puede proporcionar, ofrecer, implementar, gestionar, atender, etc., uno o más aspectos. Por ejemplo, el proveedor de servicios puede crear, mantener, soportar, etc. un código informático y/o una infraestructura informática 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 tarifa, 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 implementar una aplicación para realizar una o más realizaciones. Como ejemplo, la implementación de una aplicación comprende proporcionar una infraestructura informática operable para realizar una o más realizaciones.

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

Aunque anteriormente se han descrito diversas realizaciones, estas son solamente ejemplos. Por ejemplo, se pueden usar entornos informáticos de otras arquitecturas para incorporar y usar una o más realizaciones. Además, se pueden usar diferentes instrucciones u operaciones. Además, se pueden usar diferentes registros y/o se pueden especificar otros tipos de indicaciones (distintos de los números de registro). Son posibles muchas variaciones.

Además, otros tipos de entornos informáticos pueden beneficiarse y utilizarse. Como ejemplo, se puede utilizar un sistema de procesamiento de datos adecuado para almacenar y/o ejecutar código de programa que incluya 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, la memoria local empleada durante la ejecución real del código de programa, un almacenamiento masivo y una memoria caché que proporciona almacenamiento temporal de al menos parte del código de programa con el fin de reducir el número de veces que el código debe recuperarse del almacenamiento masivo durante la ejecución.

Dispositivos de entrada/salida o E/S (incluyendo, pero sin limitación a esto, teclados, pantallas, dispositivos señaladores, DASD, cintas, CD, DVD, memorias USB y otros medios de memoria, etc.) se pueden acoplar al sistema o bien directamente o a través de controladores de E/S intervinientes. Los adaptadores de red también pueden se pueden acoplar al sistema para permitir que el sistema de procesamiento de datos llegue a acoplarse a otros sistemas de procesamiento de datos o impresoras remotas o dispositivos de almacenamiento a través de redes privadas o públicas intervinientes. Los módems, módems de cable y tarjetas Ethernet son solo algunos de los tipos de adaptadores de red disponibles.

La terminología utilizada en esta memoria tiene el propósito de describir las realizaciones particulares solamente y no se pretende que sea limitante. Como se usa en esta memoria, las formas singulares “un”, “una” y “el”, “la” se pretende que incluyan también las formas plurales, 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 memoria descriptiva, especifican la presencia de características, entidades, etapas, operaciones, elementos y/o componentes indicados, pero no excluyen la presencia o adición de una o más características, entidades, etapas, operaciones, elementos, componentes y/o grupos de los mismos.

REIVINDICACIONES

1. Un método implementado por ordenador para facilitar el procesamiento dentro de un entorno informático, que comprende:

determinar (1100) que el procesamiento de una operación de una instrucción que se ejecuta en el procesador se ha interrumpido antes de su finalización;

extraer (1102) los metadatos del procesador, sobre la base de la determinación de que el procesamiento de la operación se ha interrumpido, siendo los metadatos los metadatos actuales del procesador (1104), en donde, la instrucción es una instrucción de clasificación (1110) y los metadatos incluyen información acerca de una o más listas de entrada para la instrucción de clasificación (1112).

almacenar (1106) los metadatos en una ubicación asociada con la instrucción, en donde la ubicación es un bloque de parámetros (360) en la memoria designado por la instrucción;

extraer los metadatos del bloque de parámetros (1054) y cargar los metadatos en el procesador (1058), sin repetir tareas para reproducir los metadatos; y

usar (1108) los metadatos que se cargan al volver a ejecutar la instrucción para reanudar el procesamiento de la instrucción desde donde se interrumpió, produciéndose previamente los metadatos y almacenándose los metadatos utilizados para preparar el procesador cuando la instrucción se vuelve a ejecutar,

en donde la información se almacena en un búfer de estado de continuación en el bloque de parámetros (1054) y la información depende del modelo en el sentido de que se almacena o presenta de manera diferente según el modelo de procesador, y

en donde, al volver a ejecutar la instrucción para reanudar el procesamiento, un número de versión de modelo en el bloque de parámetros indica qué contenidos del búfer de estado de continuación se pueden interpretar.

2. El método de la reivindicación 1, en donde los metadatos comprenden información sobre las comparaciones anteriores realizadas de los registros de una o más listas de entrada para indicar las siguientes comparaciones que se realizarán (1114).

3. El método de la reivindicación 1, en donde las siguientes comparaciones a realizar se indican sin repetir las comparaciones anteriores (1116).

4. El método de la reivindicación 3, en donde la determinación comprende comprobar un código de condición establecido en función de la terminación de la instrucción; el código de condición establecido en un valor seleccionado que indica la finalización parcial de la instrucción.

5. El método de la reivindicación 1, en donde el bloque de parámetros incluye un búfer de estado de continuación para almacenar los metadatos, comprendiendo los metadatos datos de estado internos del procesador.

6. Un sistema informático para facilitar el procesamiento seguro dentro de un entorno informático, comprendiendo el sistema informático:

una memoria; y

un procesador en comunicación con la memoria, en donde el sistema informático se configura para realizar un método que comprende:

determinar (1100) que el procesamiento de una operación de una instrucción que se ejecuta en el procesador se ha interrumpido antes de su finalización;

extraer (1102) los metadatos del procesador, sobre la base de la determinación de que el procesamiento de la operación se ha interrumpido, siendo los metadatos los metadatos actuales del procesador (1104), en donde la instrucción es una instrucción de clasificación (1110), y los metadatos incluyen información relativa a las comparaciones anteriores realizadas de registros de una o más listas de entrada con la instrucción de clasificación para indicar las siguientes comparaciones que se realizarán (1112);

almacenar (1106) los metadatos en una ubicación asociada con la instrucción, en donde la ubicación es un bloque de parámetros (360) en la memoria designado por la instrucción;

extraer los metadatos del bloque de parámetros (1054) y cargar los metadatos en el procesador (1058), sin repetir tareas para reproducir los metadatos; y

usar (1108) los metadatos que se cargan al volver a ejecutar la instrucción para reanudar el procesamiento de la instrucción desde donde se interrumpió, produciéndose previamente los metadatos y almacenándose los metadatos utilizados para preparar el procesador cuando la instrucción se vuelve a ejecutar,

en donde la información se almacena en un búfer de estado de continuación en el bloque de parámetros (1054) y

la información depende del modelo en el sentido de que se almacena o presenta de manera diferente según el modelo de procesador, y

en donde, al volver a ejecutar la instrucción para reanudar el procesamiento, un número de versión de modelo en el bloque de parámetros indica qué contenidos del búfer de estado de continuación se pueden interpretar.

7. El sistema informático de la reivindicación 6, en donde el bloque de parámetros incluye un búfer de estado de continuación para almacenar los metadatos, comprendiendo los metadatos datos de estado internos del procesador.

8. Un producto de programa informático para facilitar el procesamiento dentro de un entorno informático, comprendiendo el producto de programa informático: un soporte de almacenamiento legible por ordenador legible por un circuito de procesamiento y que almacena instrucciones para su ejecución por el circuito de procesamiento para realizar un método según cualquiera de las reivindicaciones a 1 a 5 cuando se ejecuta el programa informático.

9. Un programa informático almacenado en un medio legible por ordenador y que se puede cargar en la memoria interna de un ordenador digital, que comprende partes de código de software, cuando dicho programa se ejecuta en un ordenador, para llevar a cabo el método de cualquiera de las reivindicaciones 1 a 5.

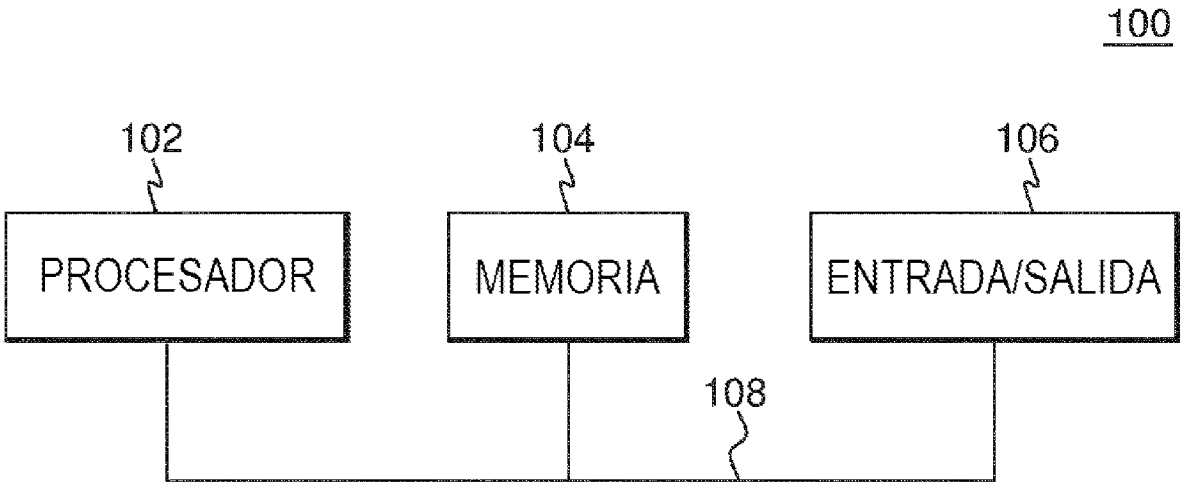

100
102
104
106
PROCESADOR
MEMORIA
ENTRADA/SALIDA
108

FIG. 1A

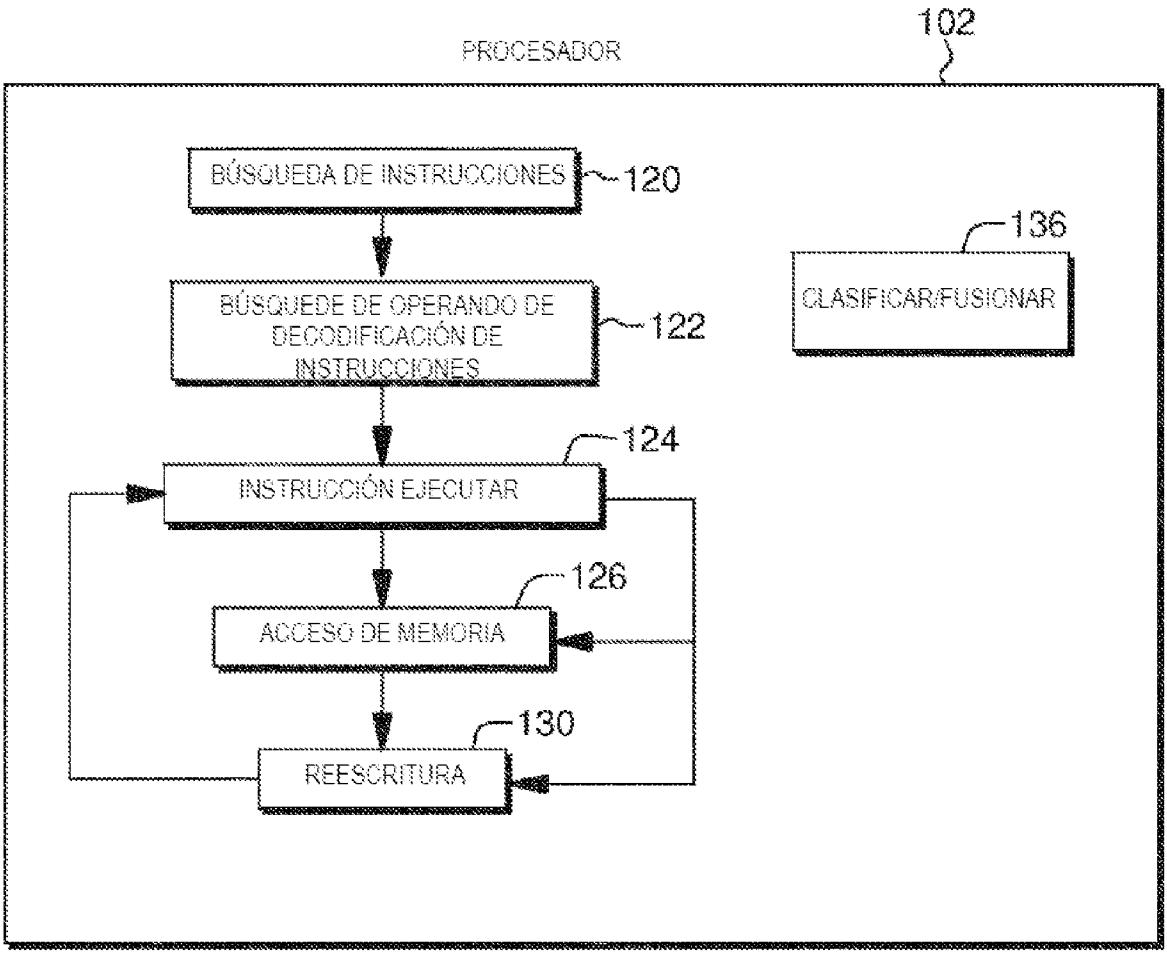

PROCESADOR
102
BÚSQUEDA DE INSTRUCCIONES
120
BÚSQUEDE DE OPERANDO DE DECODIFICACIÓN DE INSTRUCCIONES
122
INSTRUCCIÓN EJECUTAR
124
ACCESO DE MEMORIA
126
REESCRITURA
130
CLASIFICAR/FUSIONAR
136

FIG. 1B

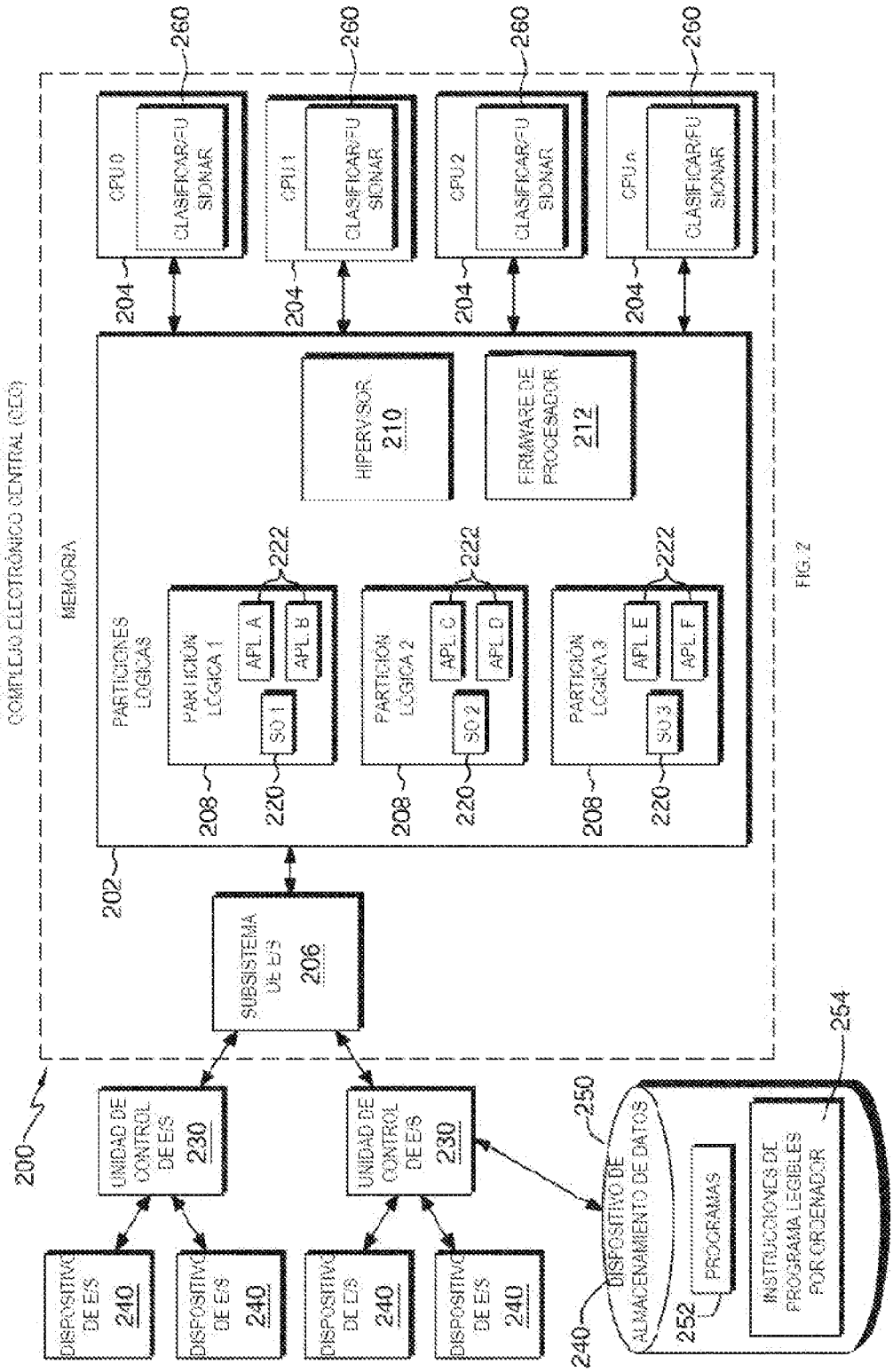

COMPLEJO ELECTRÓNICO CENTRAL (CEC)
MEMORIA
PARTICIONES LÓGICAS
PARTICIÓN LÓGICA 1
PARTICIÓN LÓGICA 2
PARTICIÓN LÓGICA 3
APL. A
APL. B
APL. C
APL. D
APL. E
APL. F
SO 1
SO 2
SO 3
HIPERVISOR 210
FIRMWARE DE PROCESADOR 212
SUBSISTEMA DE E/S 206
UNIDAD DE CONTROL DE E/S 230
DISPOSITIVO DE E/S 240
DISPOSITIVO DE ALMACENAMIENTO DE DATOS
PROGRAMAS
INSTRUCCIONES DE PROGRAMA LEGIBLES POR ORDENADOR
CPU 0
CPU 1
CPU 2
CPU n
CLASIFICAR/FU SIDNAR
200
202
204
208
220
222
240
250
252
254
260

FIG. 2

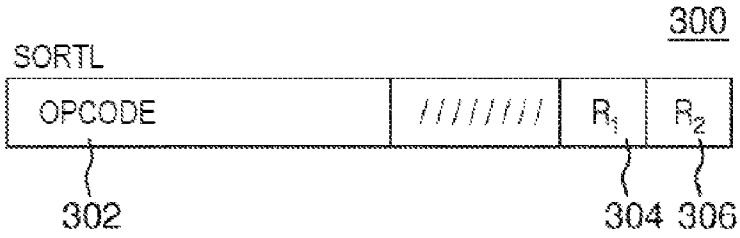


FIG. 3A

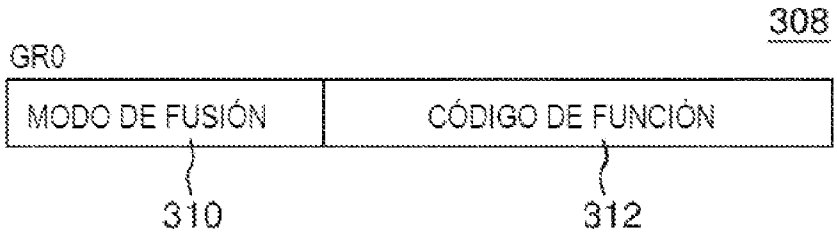


FIG. 3B

	CÓDIGO	FUNCIÓN	TAMAÑO DE BLOQUE DE PARÁMETROS (BYTES)
313	0	SORTL - QAF	32
315	1	SORTL - SFLR	$576 + 16 \times N_{IS}$
317	2	SORTL - SVLR	$576 + 16 \times N_{IS}$

FIG. 3C

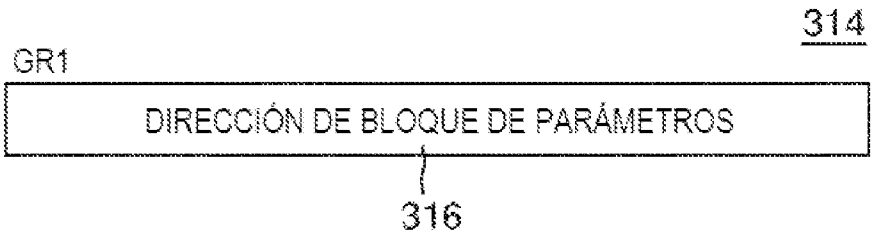


FIG. 3D

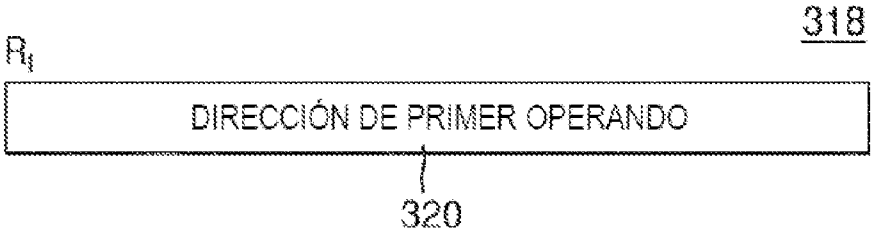


FIG. 3E

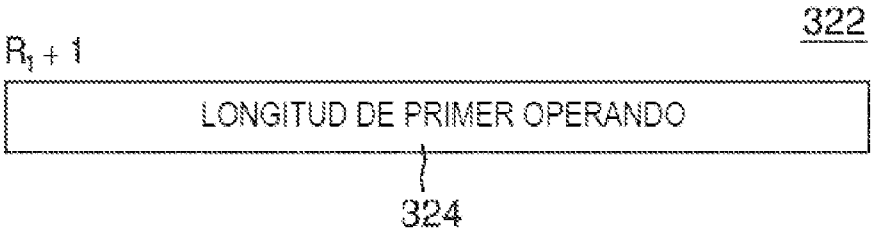


FIG. 3F

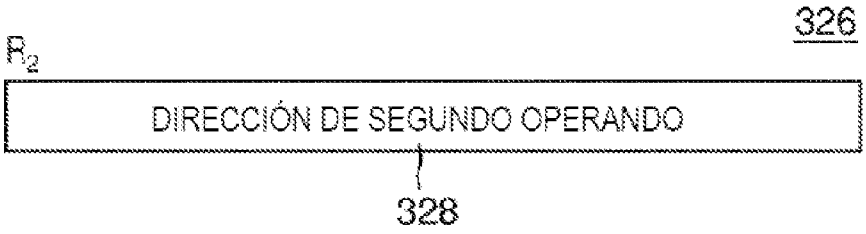


FIG. 3G

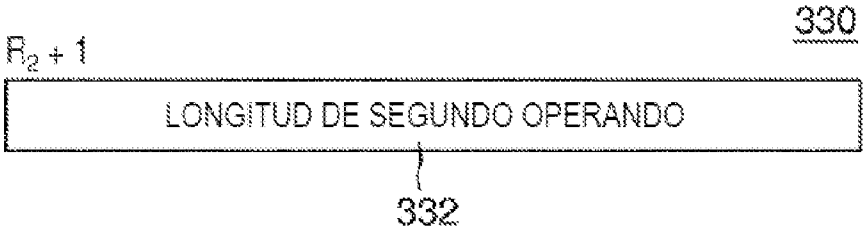


FIG. 3H

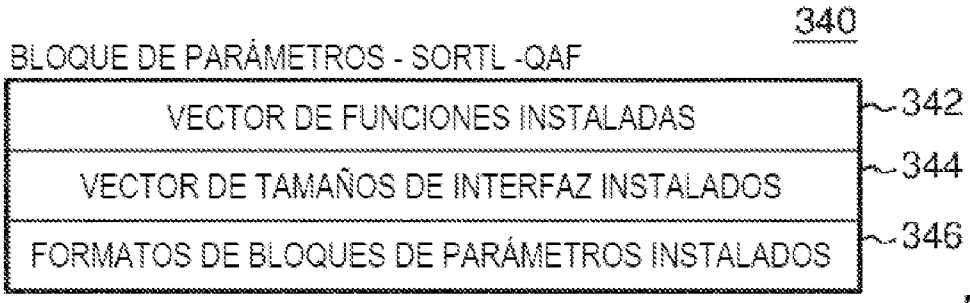


FIG. 3I

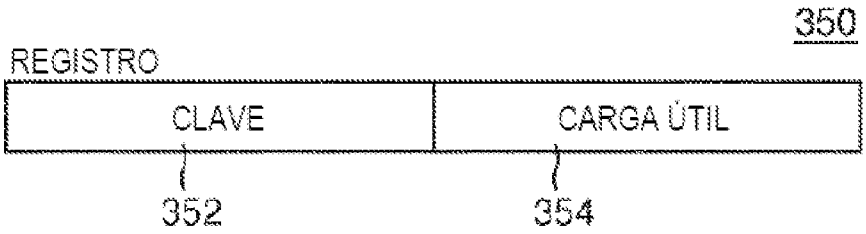


FIG. 3J


360
362
364
366
368
NÚMERO DE VERSION DE BLOQUE DE PARÁMETROS
NÚMERO DE VERSIÓN DE MODELO
ORDEN DE CLASIFICACIÓN
INDICADOR DE CONTINUACIÓN
370
LONGITUD DE CLAVE DE REGISTRO
LONGITUD DE CARGA ÚTIL DE REGISTRO
372
374
INTENCIÓN DE ACCESO AL OPERANDO
CÓDIGO DE RECUENTO DE LISTAS DE ENTRADA ACTIVAS
376
378
CONTROL DE LISTAS DE ENTRADA VACÍAS
INDICADOR DE LISTA DE ENTRADA VACÍA
NÚMERO DE LISTA DE ENTRADA VACÍA
INDICADOR DE LISTA DE ENTRADA INCOMPLETA
NÚMERO DE LISTA DE ENTRADA INCOMPLETA
386
380
382
384
ORIGEN DE BÚFER DE RECUPERACIÓN DE REGISTROS DE CONTINUACIÓN
388
BUFER DE ESTADO DE CONTINUACIÓN
390
DIRECCIÓN LISTA DE ENTRADA 0
392
LONGITUD DE LISTA DE ENTRADA 0
393
DIRECCIÓN LISTA DE ENTRADA 1
394
LONGITUD DE LISTA DE ENTRADA 1
395
DIRECCIÓN DE LISTA DE ENTRADA (Ns -1)
396
LONGITUD DE LA LISTA DE ENTRADA (Ns -1)
397

FIG. 3K

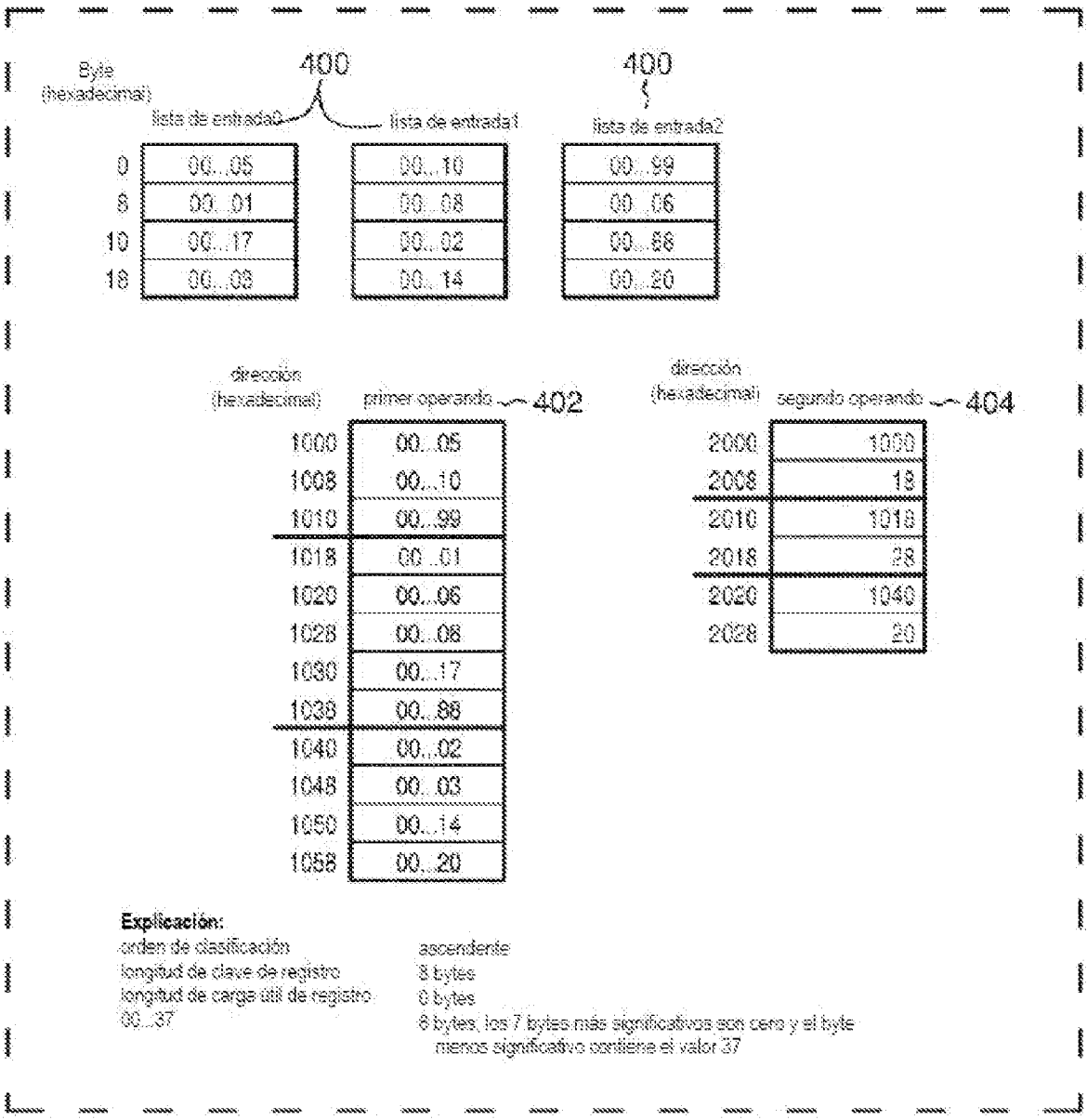

Byte (hexadecimal)
400
lista de entrada0
lista de entrada1
400
lista de entrada2
0
8
10
18
00...05
00...01
00...17
00...08
00...10
00...08
00...02
00...14
00...99
00...06
00...88
00...20
dirección (hexadecimal)
primer operando
402
1000
1008
1010
1018
1020
1028
1030
1038
1040
1048
1050
1058
00...05
00...10
00...99
00...01
00...06
00...08
00...17
00...88
00...02
00...03
00...14
00...20
dirección (hexadecimal)
segundo operando
404
2000
2008
2010
2018
2020
2028
1000
18
1018
28
1040
20
Explicación:
orden de clasificación
ascendente
longitud de clave de registro
8 bytes
longitud de carga útil de registro
0 bytes
00...37
8 bytes, los 7 bytes más significativos son cero y el byte menos significativo contiene el valor 37

FIG. 4A

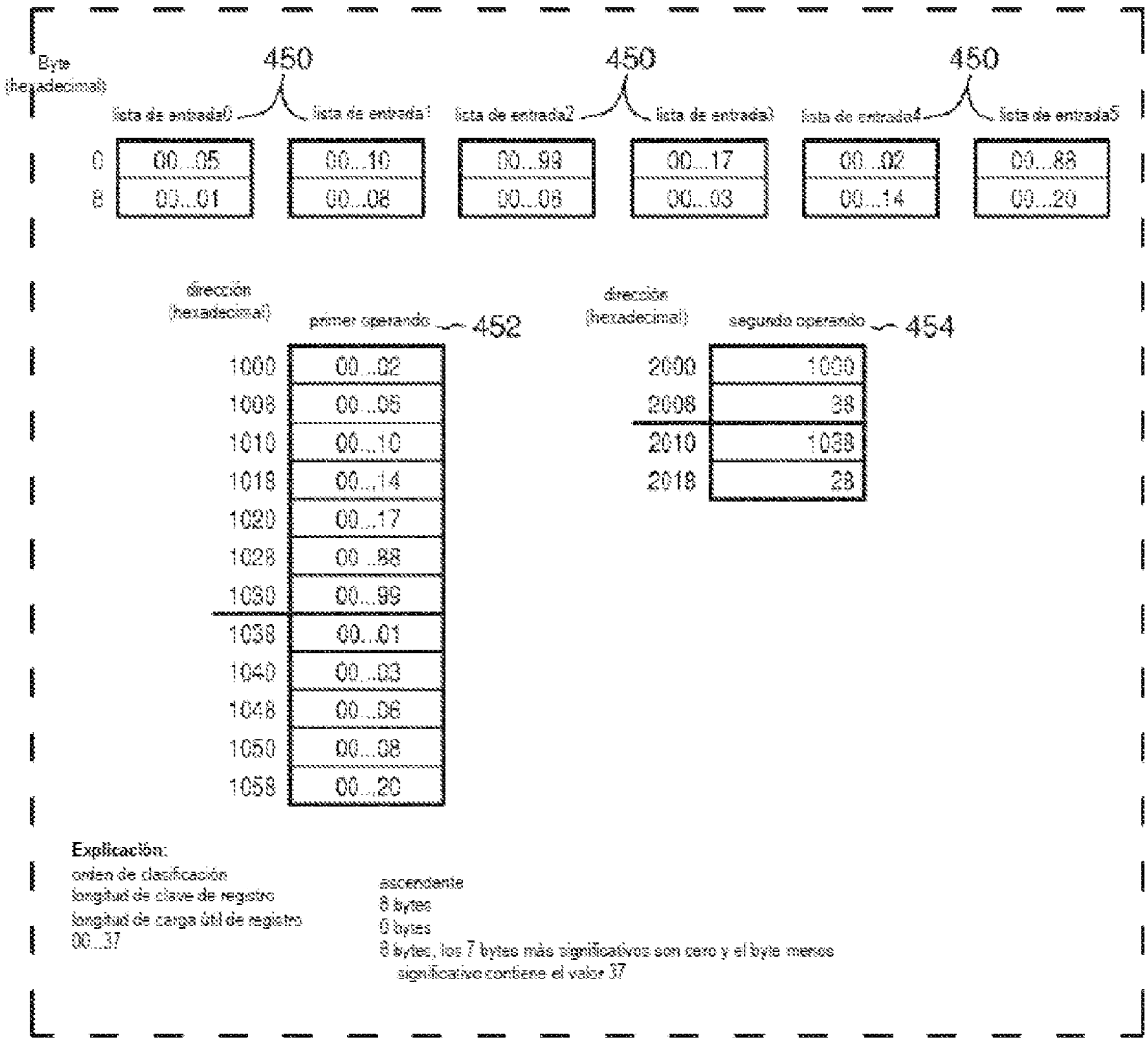
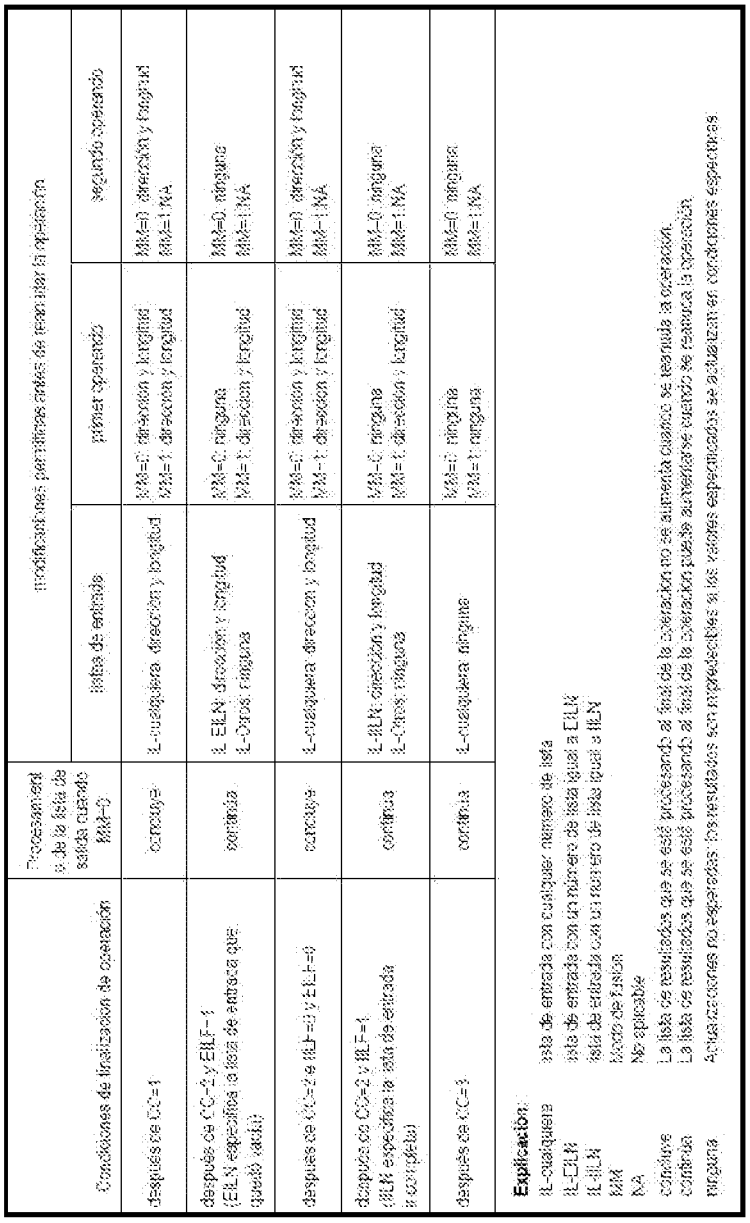

Byte (hexadecimal)
450
450
450
lista de entrada0
lista de entrada1
lista de entrada2
lista de entrada3
lista de entrada4
lista de entrada5
0
8
00...05
00...01
00...10
00...08
00...99
00...06
00...17
00...03
00...02
00...14
00...88
00...20
dirección (hexadecimal)
primer operando
452
1000
1008
1010
1018
1020
1028
1030
1038
1040
1048
1050
1058
00...02
00...05
00...10
00...14
00...17
00...88
00...99
00...01
00...03
00...06
00...08
00...20
dirección (hexadecimal)
segundo operando
454
2000
2008
2010
2018
1030
38
1038
28
Explicación:
orden de clasificación
longitud de clave de registro
longitud de carga útil de registro
00...37
ascendente
8 bytes
8 bytes
8 bytes, los 7 bytes más significativos son cero y el byte menos significativo contiene el valor 37

FIG. 4B

Entrada a SORTL SFLR	Antes de la operación	Después de completar parcialmente la operación (CC=1, 2 o 3)	Antes de reanudar la operación	Tras completar la operación (CC=0)
número de versión de bloque de parámetros	elección del programa	sin cambios	igual que antes de BO	sin cambios
número de versión de modelo	ceros recomendados	establecido	igual que después de PC	establecido
orden de clasificación	elección del programa	sin cambios	igual que antes de BO	sin cambios
indicador de continuación	se requiere cero	establecer en uno	igual que después de PC	establecer en cero
longitud de clave de registro	elección del programa	sin cambios	igual que antes de BO	sin cambios
longitud de carga útil de registro	elección del programa	sin cambios	igual que antes de BO	sin cambios
intención de acceso a operando	elección del programa	sin cambios	igual que antes de BO	sin cambios
recuento de listas de entrada activas	elección del programa	sin cambios	igual que antes de BO	sin cambios
control de listas de entrada vacías	elección del programa	sin cambios	igual que antes de BO	sin cambios
indicador de lista de entrada vacía	cero recomendado	establecido	igual que después de PC	establecido
número de lista de entrada vacía	ceros recomendados	establecido	igual que después de PC	establecido
indicador de lista de entrada incompleta	cero recomendado	establecido	igual que después de PC	establecido
número de lista de entrada incompleta	ceros recomendados	establecido	igual que después de PC	establecido
origen de búfer de recuperación de registros de continuación	elección del programa	sin cambios	igual que antes de BO	sin cambios
búfer de estado de continuación	ceros recomendados	establecido	igual que después de PC	indefinido
dirección lista de entradaN	elección del programa	modificado	se aplican restricciones; consúltese la FIG. 5B	modificado
longitud lista de entradaN	elección del programa	modificado	se aplican restricciones; consúltese la FIG. 5B	modificado
modo de fusión	elección del programa	sin cambios	igual que antes de BO	sin cambios
dirección de primer operando	elección del programa	modificado	se aplican restricciones; consúltese la FIG. 5B	modificado
longitud de primer operando	elección del programa	modificado	se aplican restricciones; consúltese la FIG. 5B	modificado
dirección de segundo operando (cuando MM=0)	elección del programa	modificado	se aplican restricciones; consúltese la FIG. 5B	modificado
longitud de segundo operando (cuando MM=0)	elección del programa	modificado	se aplican restricciones; consúltese la FIG. 5B	modificado
búfer de recuperación de registros de continuación (cuando MM=0)	ceros recomendados	establecido	igual que después de PC	indefinido

Explicación:
BO operación de comienzo
MM Modo de fusión
PC finalización parcial

FIG. 5A



Condiciones de finalización de operación	Procesamiento de la lista de salida cuando MM=0	modificaciones permitidas antes de reanudar la operación		
		lista de entrada	primer operando	segundo operando
después de CC=1	continúa	4-cualquiera dirección y longitud	MM=0: dirección y longitud MM=1: dirección y longitud	MM=0: dirección y longitud MM=1:NA
después de CC=2 y ELF=1 (ELN especifica la lista de entrada que causó error)	continúa	4: ELN: dirección y longitud 4-Otros: ninguna	MM=0: ninguna MM=1: dirección y longitud	MM=0: ninguna MM=1:NA
después de CC=2 e ILF=0 y ELF=0	continúa	4-cualquiera dirección y longitud	MM=0: dirección y longitud MM=1: dirección y longitud	MM=0: dirección y longitud MM=1:NA
después de CC=2 y ILF=1 (ILN especifica la lista de entrada incompleta)	continúa	4: ILN: dirección y longitud 4-Otros: ninguna	MM=0: ninguna MM=1: dirección y longitud	MM=0: ninguna MM=1:NA
después de CC=3	continúa	4-cualquiera ninguna	MM=0: ninguna MM=1: ninguna	MM=0: ninguna MM=1:NA

Explicación:

4-cualquiera	lista de entrada con cualquier número de lista
4-ELN	lista de entrada con un número de lista igual a ELN
4-ILN	lista de entrada con un número de lista igual a ILN
MM	Modo de Fusión
NA	No aplicable
continúa	La lista de resultados que se está procesando al final de la operación no se aumenta cuando se reanuda la operación.
continúa	La lista de resultados que se está procesando al final de la operación puede aumentarse cuando se reanuda la operación.
ninguna	Aplicaciones no esperadas: los resultados son impredecibles si las reglas especificadas se aplicaran en condiciones esperadas

FIG. 5B

UBICACIÓN DE PRIMER OPERANDO ANTES DE EJECUTAR SORTL CON MM = 0:

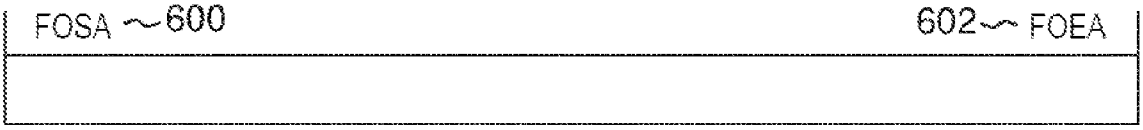


FIG. 6A

UBICACIÓN DE PRIMER OPERANDO DESPUÉS DE EJECUTAR SORTL CON MM = 0:

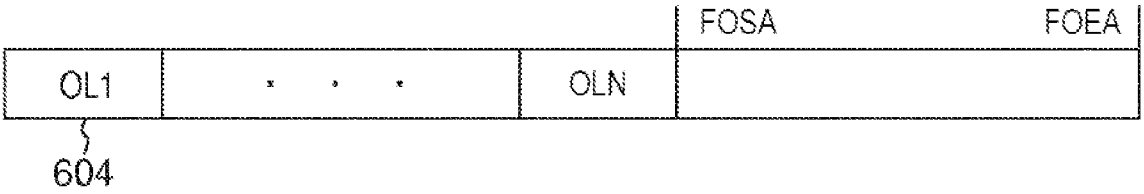


FIG. 6B

UBICACIÓN DE SEGUNDO OPERANDO ANTES DE EJECUTAR SORTL CON MM = 0:

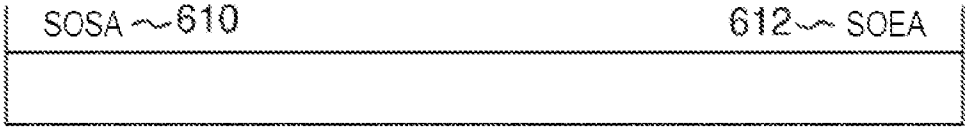


FIG. 6C

UBICACIÓN DE SEGUNDO OPERANDO DESPUÉS DE EJECUTAR SORTL CON MM = 0:

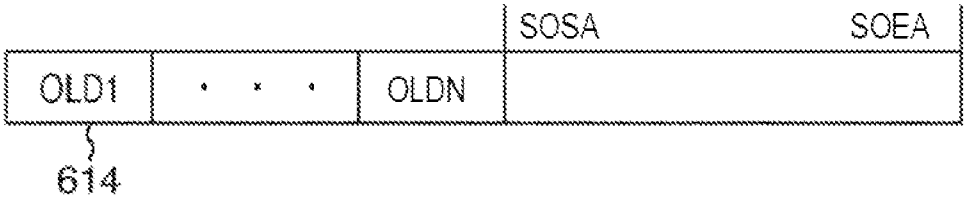


FIG. 6D

UBICACIÓN DE PRIMER OPERANDO ANTES DE EJECUTAR SORTL CON MM = 1:

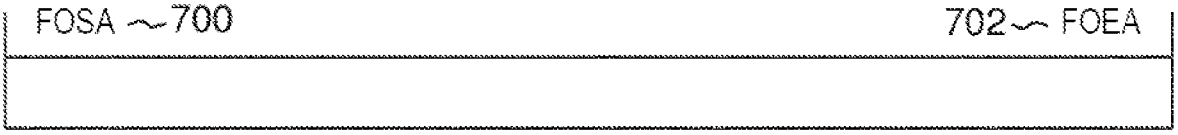


FIG. 7A

UBICACIÓN DE PRIMER OPERANDO DESPUÉS DE EJECUTAR SORTL CON MM = 1:

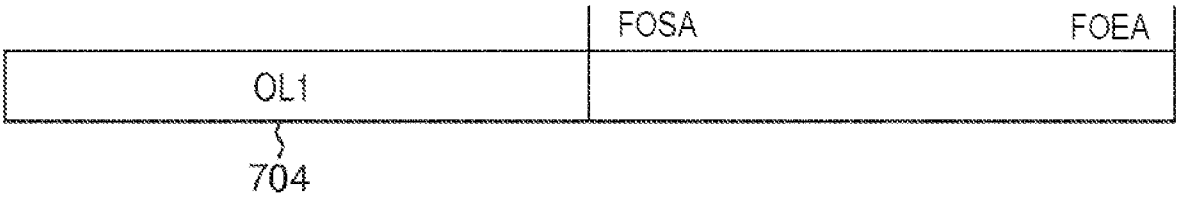


FIG. 7B

EILCL (bin)	condición que hace que finalice la operación	resultados				
		CC	IILF	IILN	EILF	EILN
-	finalización normal	0	0	0	0	0
-	la longitud de primer o segundo operando es insuficiente	1	0	0	0	0
-	Se determina que IL0 está incompleta	2	1	0 (IL0)	0	0
-	Se determina que ILN está incompleta	2	1	N (ILN)	0	0
10	El IL0 quedó vacío (ILN también puede estar vacía)	2	0	0	0	0
01	ILN quedó vacía (IL0 también puede estar vacía)	2	0	0	0	0
11	IL0 quedó vacío (solo una lista de entrada quedó vacía)	2	0	0	1	0 (IL0)
11	ILN quedó vacío (solo una lista de entrada quedó vacía)	2	0	0	1	N (ILN)
-	Número de bytes procesados determinado por la CPU	3	0	0	0	0

Explicación:

- cualquier valor
IL0 lista de entrada con número de lista 0
ILN lista de entrada con el número de lista N, donde N > 0

FIG. 8

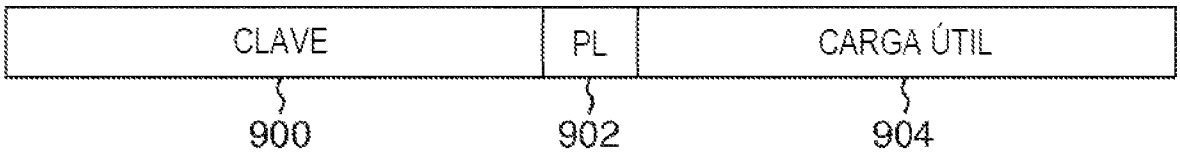


FIG. 9

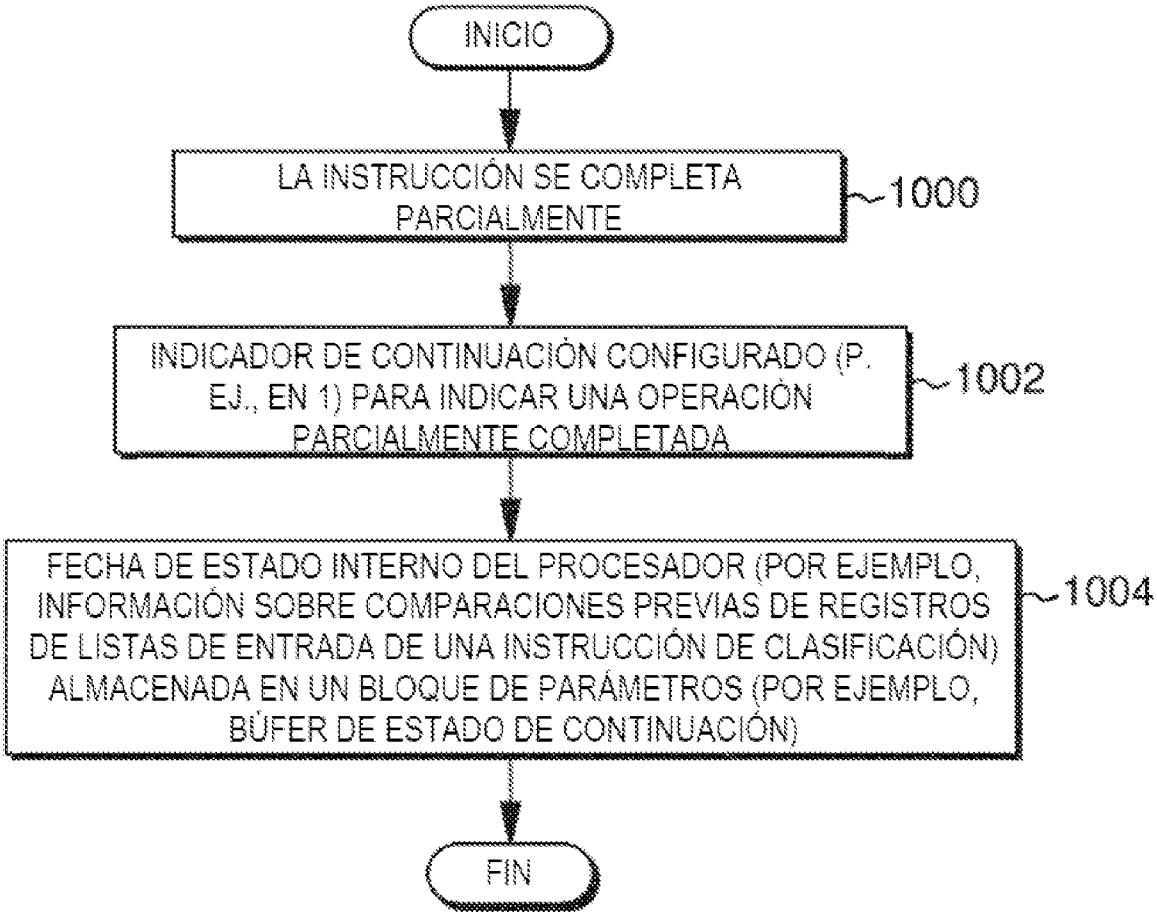

INICIO
LA INSTRUCCIÓN SE COMPLETA PARCIALMENTE
1000
INDICADOR DE CONTINUACIÓN CONFIGURADO (P. EJ., EN 1) PARA INDICAR UNA OPERACIÓN PARCIALMENTE COMPLETADA
1002
FECHA DE ESTADO INTERNO DEL PROCESADOR (POR EJEMPLO, INFORMACIÓN SOBRE COMPARACIONES PREVIAS DE REGISTROS DE LISTAS DE ENTRADA DE UNA INSTRUCCIÓN DE CLASIFICACIÓN) ALMACENADA EN UN BLOQUE DE PARÁMETROS (POR EJEMPLO, BÚFER DE ESTADO DE CONTINUACIÓN)
1004
FIN

FIG. 10A

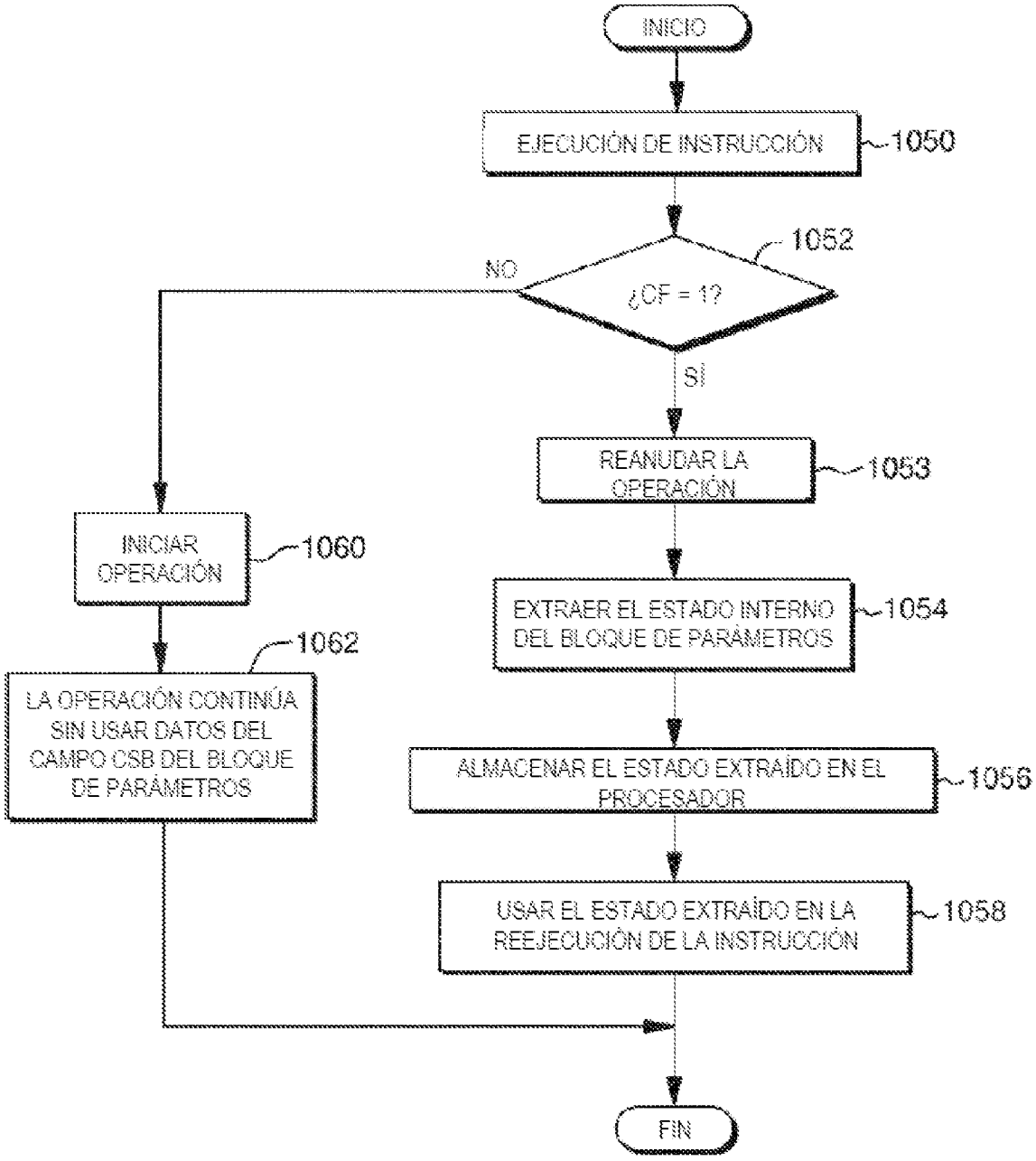

INICIO
EJECUCIÓN DE INSTRUCCIÓN
1050
1052
NO
¿CF = 1?
SÍ
REANUDAR LA OPERACIÓN
1053
INICIAR OPERACIÓN
1060
EXTRAER EL ESTADO INTERNO DEL BLOQUE DE PARÁMETROS
1054
1062
LA OPERACIÓN CONTINÚA SIN USAR DATOS DEL CAMPO CSB DEL BLOQUE DE PARÁMETROS
ALMACENAR EL ESTADO EXTRAÍDO EN EL PROCESADOR
1056
USAR EL ESTADO EXTRAÍDO EN LA REEJECUCIÓN DE LA INSTRUCCIÓN
1058
FIN

FIG. 10B

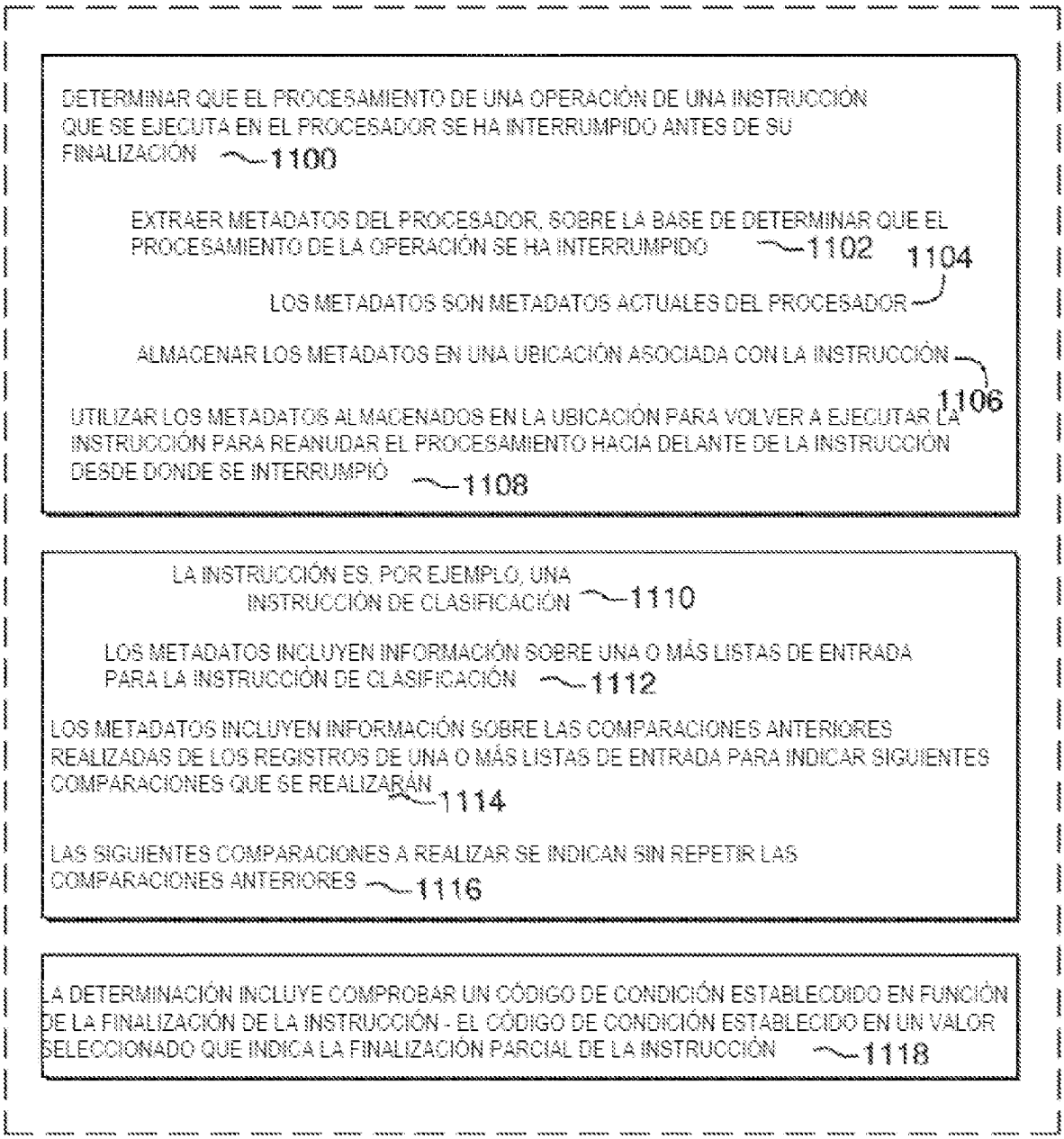

DETERMINAR QUE EL PROCESAMIENTO DE UNA OPERACIÓN DE UNA INSTRUCCIÓN QUE SE EJECUTA EN EL PROCESADOR SE HA INTERRUMPIDO ANTES DE SU FINALIZACIÓN 1100
EXTRAER METADATOS DEL PROCESADOR, SOBRE LA BASE DE DETERMINAR QUE EL PROCESAMIENTO DE LA OPERACIÓN SE HA INTERRUMPIDO 1102
1104
LOS METADATOS SON METADATOS ACTUALES DEL PROCESADOR
ALMACENAR LOS METADATOS EN UNA UBICACIÓN ASOCIADA CON LA INSTRUCCIÓN
1106
UTILIZAR LOS METADATOS ALMACENADOS EN LA UBICACIÓN PARA VOLVER A EJECUTAR LA INSTRUCCIÓN PARA REANUDAR EL PROCESAMIENTO HACIA DELANTE DE LA INSTRUCCIÓN DESDE DONDE SE INTERRUMPIÓ 1108
LA INSTRUCCIÓN ES, POR EJEMPLO, UNA INSTRUCCIÓN DE CLASIFICACIÓN 1110
LOS METADATOS INCLUYEN INFORMACIÓN SOBRE UNA O MÁS LISTAS DE ENTRADA PARA LA INSTRUCCIÓN DE CLASIFICACIÓN 1112
LOS METADATOS INCLUYEN INFORMACIÓN SOBRE LAS COMPARACIONES ANTERIORES REALIZADAS DE LOS REGISTROS DE UNA O MÁS LISTAS DE ENTRADA PARA INDICAR SIGUIENTES COMPARACIONES QUE SE REALIZARÁN 1114
LAS SIGUIENTES COMPARACIONES A REALIZAR SE INDICAN SIN REPETIR LAS COMPARACIONES ANTERIORES 1116
LA DETERMINACIÓN INCLUYE COMPROBAR UN CÓDIGO DE CONDICIÓN ESTABLECIDO EN FUNCIÓN DE LA FINALIZACIÓN DE LA INSTRUCCIÓN - EL CÓDIGO DE CONDICIÓN ESTABLECIDO EN UN VALOR SELECCIONADO QUE INDICA LA FINALIZACIÓN PARCIAL DE LA INSTRUCCIÓN 1118

FIG. 11A

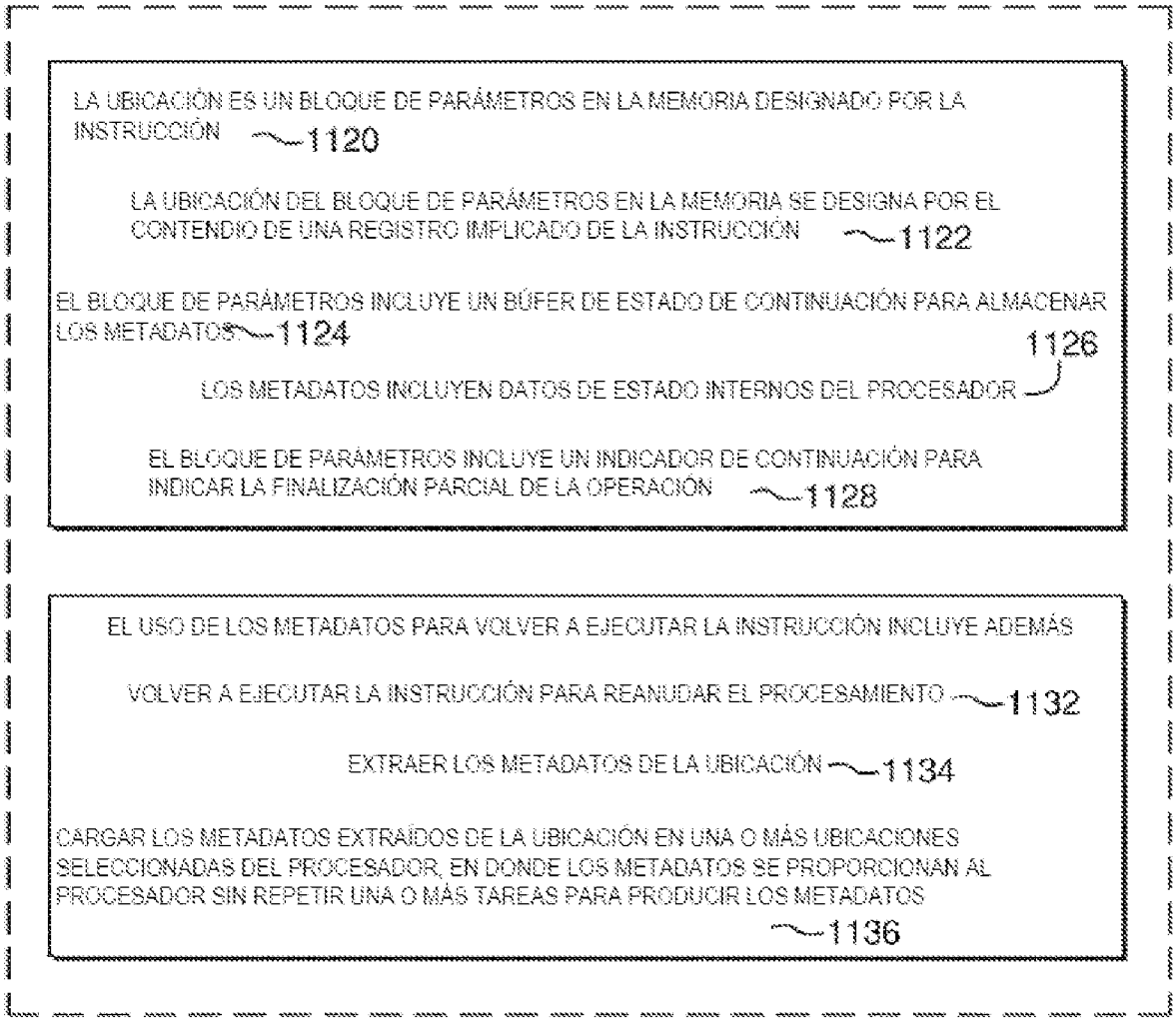

LA UBICACIÓN ES UN BLOQUE DE PARÁMETROS EN LA MEMORIA DESIGNADO POR LA INSTRUCCIÓN 1120
LA UBICACIÓN DEL BLOQUE DE PARÁMETROS EN LA MEMORIA SE DESIGNA POR EL CONTENDIO DE UNA REGISTRO IMPLICADO DE LA INSTRUCCIÓN 1122
EL BLOQUE DE PARÁMETROS INCLUYE UN BÚFER DE ESTADO DE CONTINUACIÓN PARA ALMACENAR LOS METADATOS 1124
1126
LOS METADATOS INCLUYEN DATOS DE ESTADO INTERNOS DEL PROCESADOR
EL BLOQUE DE PARÁMETROS INCLUYE UN INDICADOR DE CONTINUACIÓN PARA INDICAR LA FINALIZACIÓN PARCIAL DE LA OPERACIÓN 1128
EL USO DE LOS METADATOS PARA VOLVER A EJECUTAR LA INSTRUCCIÓN INCLUYE ADEMÁS
VOLVER A EJECUTAR LA INSTRUCCIÓN PARA REANUDAR EL PROCESAMIENTO 1132
EXTRAER LOS METADATOS DE LA UBICACIÓN 1134
CARGAR LOS METADATOS EXTRAÍDOS DE LA UBICACIÓN EN UNA O MÁS UBICACIONES SELECCIONADAS DEL PROCESADOR, EN DONDE LOS METADATOS SE PROPORCIONAN AL PROCESADOR SIN REPETIR UNA O MÁS TAREAS PARA PRODUCIR LOS METADATOS 1136

FIG. 11B

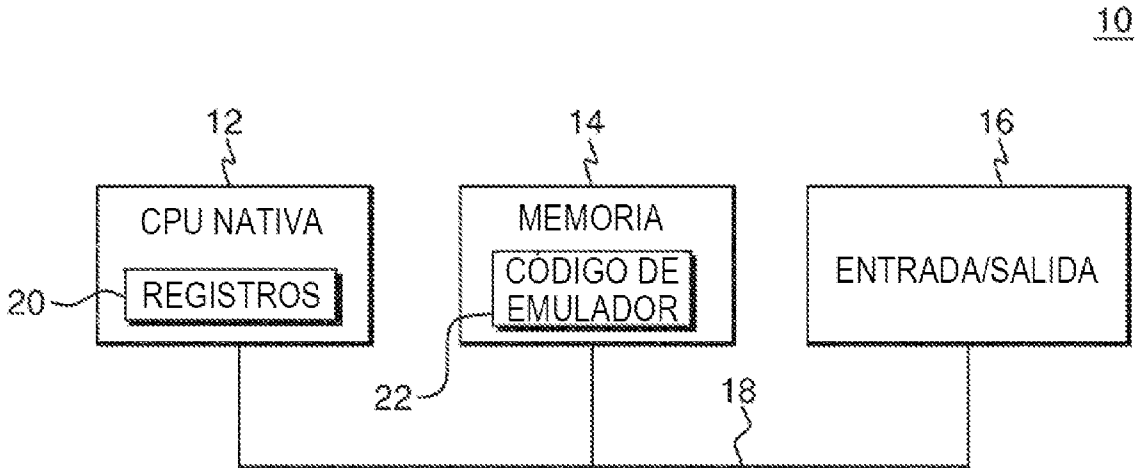


FIG. 12A

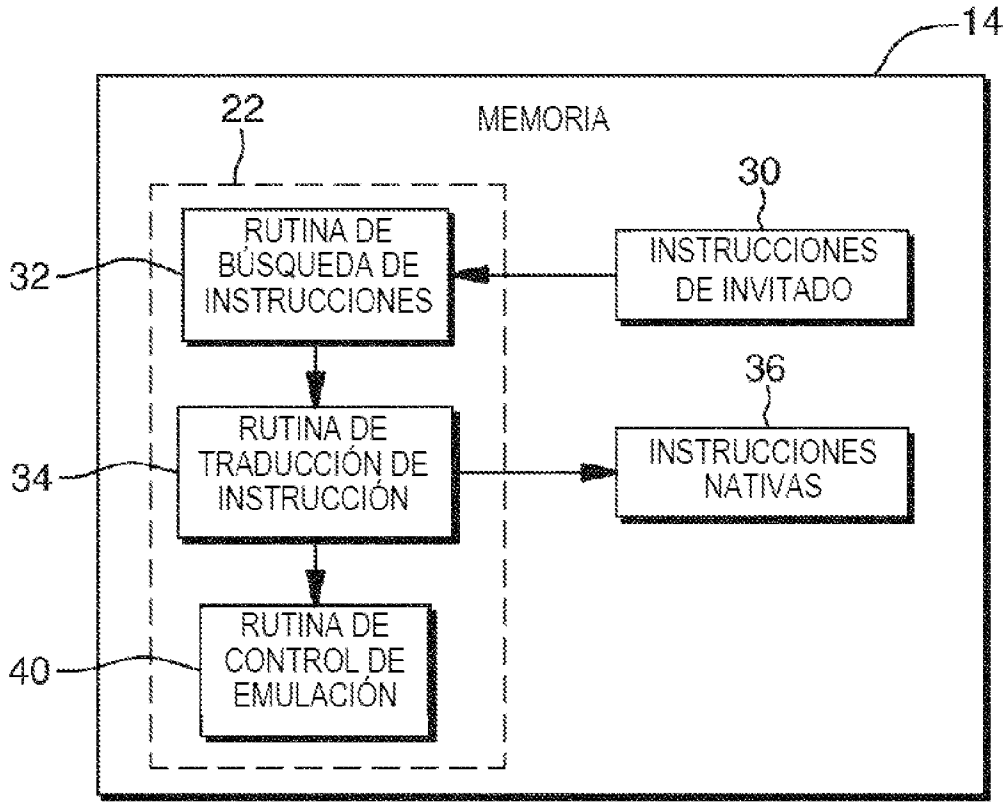


FIG. 12B

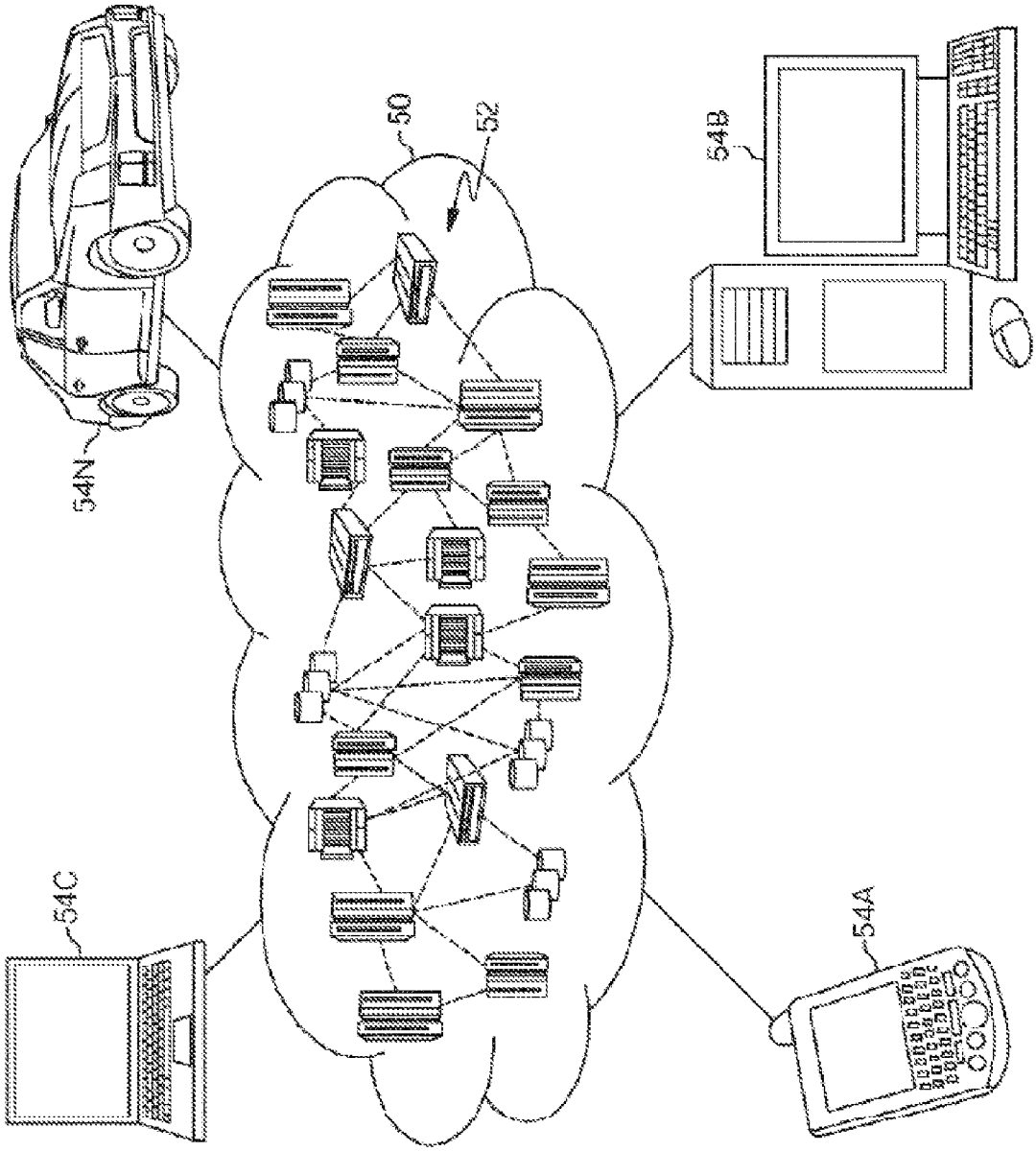


FIG. 13

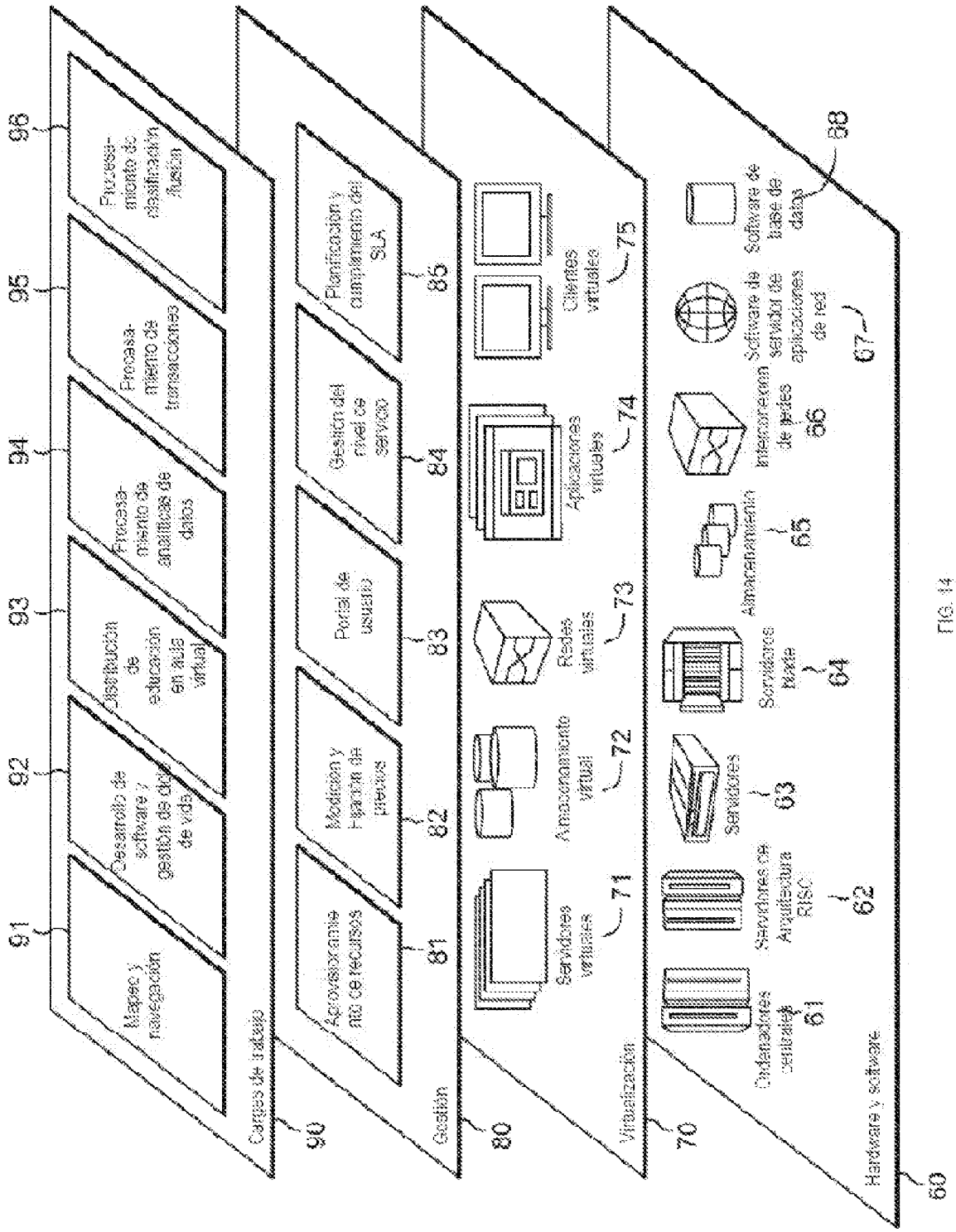

Hardware y software
60
Virtualización
70
Gestión
80
Cargas de trabajo
90
Ordenadores centrales
61
Servidores de Arquitectura RISC
62
Servidores
63
Servidores blade
64
Almacenamiento
65
Interconexión de redes
66
Software de servidor de aplicaciones de red
67
Software de base de datos
68
Servidores virtuales
71
Almacenamiento virtual
72
Redes virtuales
73
Aplicaciones virtuales
74
Clientes virtuales
75
Aprovisionamiento de recursos
81
Medición y fijación de precios
82
Portal de usuario
83
Gestión del nivel de servicio
84
Planificación y cumplimiento del SLA
85
Mapeo y navegación
91
Desarrollo de software y gestión del ciclo de vida
92
Distribución de educación en aula virtual
93
Procesamiento de analíticas de datos
94
Procesamiento de transacciones
95
Procesamiento de clasificación y fusión
96

FIG. 14