(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 552 175**

(51) Int. CI.:

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

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **07.03.2013 E 13761350 (1)**

(97) Fecha y número de publicación de la concesión europea: **07.10.2015 EP 2758891**

(54) Título: **Instrucción Encontrar Elemento Igual de Vector**

(30) Prioridad:

15.03.2012 US 201213421448

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
26.11.2015

(73) Titular/es:

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

(72) Inventor/es:

BRADBURY, JONATHAN DAVID;
SLEGEL, TIMOTHY;
SCHWARZ, ERIC MARK y
GSCHWIND, MICHAEL KARL

(74) Agente/Representante:

DE ELZABURU MÁRQUEZ, Alberto

DESCRIPCIÓN

Instrucción Encontrar Elemento Igual de Vector

Antecedentes

Un aspecto de la invención se refiere, en general, a procesamiento de texto y, en particular, a facilitar procesamiento asociado con datos de caracteres.

Un procesamiento de texto a menudo requiere la comparación de datos de caracteres, incluyendo, pero no limitado a, la comparación de cadenas de datos de caracteres. Típicamente, las instrucciones usadas para comparar datos de caracteres comparan un único octeto de datos a la vez.

Además, el procesamiento de texto a menudo requiere otros tipos de procesamiento de cadenas, incluyendo encontrar el punto de terminación (por ejemplo, el final de una cadena), determinar la longitud de los datos de caracteres, encontrar un carácter particular, etc. Las instrucciones actuales para realizar estos tipos de procesamiento tienden a ser ineficientes.

El documento US 2004/215 924 A1 describe un método de localización de un valor objetivo que incluye cargar el valor objetivo en elementos de un primer registro.

El documento EP 1296 222 A2 describe un circuito comparador de magnitud.

Breve compendio

Los defectos de la técnica anterior se superan y se proporcionan ventajas a través del suministro de un método según la reivindicación 1.

También se proporcionan un producto de programa de ordenador y un sistema de ordenador correspondientes.

Rasgos y ventajas adicionales se realizan a través de las técnicas de la presente invención. Otras realizaciones y aspectos de la invención se describen en detalle en la presente memoria y se consideran una parte de la invención reivindicada.

Breve descripción de las diversas vistas de los dibujos

Uno o más aspectos de la presente invención se señalan particularmente y reivindican distintivamente como ejemplos en las reivindicaciones en la finalización de esta especificación. Los precedentes y otros objetivos, rasgos y ventajas de la invención son evidentes a partir de la siguiente descripción detallada tomada en conjunto con los dibujos anexos en los cuales:

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

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

La FIG. 2B representa detalles adicionales de la memoria de la FIG. 2A, según un aspecto de la presente invención;

La FIG. 3 representa una realización de un formato de una instrucción Encontrar Elemento Igual de Vector, según un aspecto de la presente invención;

La FIG. 4 representa una realización de la lógica asociada con una instrucción Encontrar Elemento Igual de Vector, según un aspecto de la presente invención;

La FIG. 5 representa una realización de varios bloques de procesamiento para realizar la lógica de la FIG. 4, según un aspecto de la presente invención;

La FIG. 6 representa un ejemplo de un campo de registro, según un aspecto de la presente invención;

La FIG. 7 representa una realización de un formato de una instrucción Encontrar Cualquier Igual de Vector, según un aspecto de la presente invención;

La FIG. 8 representa una realización de la lógica asociada con una instrucción Encontrar Cualquier Igual de Vector, según un aspecto de la presente invención;

La FIG. 9 representa una realización de un producto de programa de ordenador que incorpora uno o más aspectos de la presente invención;

La FIG. 10 representa una realización de un sistema de ordenador central para incorporar y usar uno o más aspectos de la presente invención;

La FIG. 11 representa un ejemplo adicional de un sistema de ordenador para incorporar y usar uno o más aspectos de la presente invención;

La FIG. 12 representa otro ejemplo adicional de un sistema de ordenador que comprende una red de ordenadores para incorporar y usar uno o más aspectos de la presente invención;

La FIG. 13 representa una realización de varios elementos de un sistema de ordenador para incorporar y usar uno o más aspectos de la presente invención;

La FIG. 14A representa una realización de la unidad de ejecución del sistema de ordenador de la FIG. 13 para incorporar y usar uno o más aspectos de la presente invención;

La FIG. 14B representa una realización de la unidad de ramal del sistema de ordenador de la FIG. 13 para incorporar y usar uno o más aspectos de la presente invención;

La FIG. 14C representa una realización de la unidad de carga/almacén del sistema de ordenador de la FIG. 13 para incorporar y usar uno o más aspectos de la presente invención; y

La FIG. 15 representa una realización de un sistema de ordenador central emulado para incorporar y usar uno o más aspectos de la presente invención.

Descripción detallada

Según un aspecto de la presente invención, se proporciona una capacidad para facilitar procesamiento de datos de caracteres, pero no limitada a, caracteres alfabéticos, en cualquier idioma; dígitos numéricos; puntuación; y/u otros símbolos. Los datos de caracteres pueden ser o no ser cadenas de datos. Asociados con datos de caracteres están estándares, ejemplos de los cuales incluyen, pero no se limitan a, ASCII (Código Estándar Americano para Intercambio de Información); Unicode, incluyendo, pero no limitado a, UTF (Formato de Transformación Unicódigo) 8; UTF16; etc.

En un ejemplo, se proporciona una instrucción Encontrar Elemento Igual que compara datos de múltiples vectores para igualdad y proporciona una indicación de igualdad, si existe la igualdad. En un ejemplo, un índice asociado con el elemento igual se almacena en un registro de vector objetivo.

Como se describe en la presente memoria, un elemento de un registro de vector (también conocido como un vector) es de uno, dos o cuatro octetos de longitud, como ejemplos; y un operando de vector es, por ejemplo, un operando SIMD (Instrucción Única, Datos Múltiples) que tiene una pluralidad de elementos. En otras realizaciones, los elementos pueden ser de otros tamaños; y un operando de vector necesita no ser SIMD y/o puede incluir un elemento.

En una realización adicional, la misma instrucción, la instrucción Encontrar Elemento Igual, también busca un vector seleccionado para elementos nulos, también conocidos como elementos cero (por ejemplo, el elemento entero es cero). Un elemento nulo o cero indica terminación de los datos de caracteres; por ejemplo, un final de una cadena de datos particular. Un resultado de la instrucción es dependiente de si se proporciona la búsqueda nula o solo la comparación.

Aún en una realización adicional, se proporciona una instrucción Encontrar Cualquier Igual de Vector que busca un vector para caracteres particulares y/o para elementos cero y devuelve una máscara o índice de octeto del carácter o elemento cero coincidente.

Una realización de un entorno informático para incorporar y usar uno o más aspectos de la presente invención se describe con referencia a la FIG. 1. Un entorno informático 100 incluye, por ejemplo, un procesador 102 (por ejemplo, una unidad central de proceso), una memoria 104 (por ejemplo, una memoria principal) y uno o más dispositivos de entrada/salida (I/O) y/o interfaces 106 acoplados entre sí a través, por ejemplo, de uno o más canales principales 108 y/u otras conexiones.

En un ejemplo, el procesador 102 se basa en la z/Architecture ofrecida por International Business Machines Corporation y es parte de un servidor, tal como el servidor del System z, que también se ofrece por International Business Machines Corporation e implementa la z/Architecture. Una realización de la z/Architecture se describe en una publicación de IBM® titulada, “z/Architecture Principles of Operation”, Publicación de IBM® Nº SA22-7832-08, Novena Edición, agosto de 2010. En un ejemplo, el procesador ejecuta un sistema operativo, tal como z/OS, también ofrecido por International Business Machines Corporation. IBM®, Z/ARCHITECTURE® y Z/OS® son marcas registradas de International Business Machines Corporation, Armonk, Nueva York, EE.UU. Otros nombres usados en la presente memoria pueden ser marcas registradas, marcas comerciales o nombres de productos de International Business Machines Corporation u otras compañías.

En una realización adicional, el procesador 102 se basa en la Power Architecture ofrecida por International Business Machines Corporation. Una realización de la Power Architecture se describe en “Power ISA™ Version 2.06 Revision B”, International Business Machines Corporation, 23 de julio de 2010. POWER ARCHITECTURE® es una marca registrada de International Business Machines Corporation.

Aún en una realización adicional, el procesador 102 se basa en una arquitectura Intel ofrecida por Intel Corporation. Una realización de la arquitectura Intel se describe en “Intel® 64 and IA-32 Architectures Developer’s Manual: Vol. 2B, Instructions Set Reference, A-L”, Número de Orden 253666-041US, diciembre de 2011 y en “Intel® 64 and IA-32 Architectures Developer’s Manual: Vol. 2B, Instructions Set Reference, M-Z”, Número de Orden 253667-041US, diciembre de 2011. Intel® es una marca registrada de Intel Corporation, Santa Clara, California.

Otra realización de un entorno informático para incorporar y usar uno o más aspectos de la presente invención se describe con referencia a la FIG. 2A. En este ejemplo, un entorno informático 200 incluye, por ejemplo, una unidad de procesamiento central nativo 202, una memoria 204 y uno o más dispositivos de entrada/salida y/o interfaces 206 acoplados a otra vía, por ejemplo, uno o más canales principales 208 y/u otras conexiones. Como ejemplos, el entorno informático 200 puede incluir un procesador PowerPC, un servidor pSeries o un servidor xSeries ofrecidos por International Business Machines Corporation, Armonk, Nueva York; y un HP Superdome con procesadores Intel Itanium II ofrecido por Hewlett Packard Co., Palo Alto, California; y/u otras máquinas basadas en arquitecturas ofrecidas por International Business Machines Corporation, Hewlett Packard, Intel, Oracle u otros.

La unidad central de proceso nativa 202 incluye uno o más registros nativos 210, tales como uno o más registros de propósito general y/o uno o más registros de propósito especial usados durante el procesamiento dentro del entorno. Estos registros incluyen información que representa el estado del entorno en cualquier punto de tiempo particular.

Además, la unidad central de proceso nativa 202 ejecuta instrucciones y código que están almacenados en la memoria 204. En un ejemplo particular, la unidad central de proceso ejecuta un código emulador 212 almacenado en la memoria 204. Este código permite al entorno de procesamiento configurado en una arquitectura emular otra arquitectura. Por ejemplo, el código emulador 212 permite a máquinas basadas en arquitecturas distintas de la z/Architecture, tal como procesadores PowerPC, servidores pSeries, servidores xSeries, servidores HP Superdome u otros, emular la z/Architecture y ejecutar software e instrucciones desarrolladas basadas en la z/Architecture.

Detalles adicionales relativos al código emulador 212 se describen con referencia a la FIG. 2B. Las instrucciones invitadas 250 comprenden instrucciones software (por ejemplo, instrucciones máquina) que se desarrollaron para ser ejecutadas en una arquitectura distinta de la de la CPU nativa 202. Por ejemplo, las instrucciones invitadas 250 se pueden haber diseñado para ejecutar sobre un procesador z/Architecture 102, pero en su lugar, están siendo emuladas sobre la CPU nativa 202, que puede ser, por ejemplo, un procesador Intel Itanium II. En un ejemplo, el código emulador 212 incluye una unidad de búsqueda de instrucciones 252 para obtener una o más instrucciones invitadas 250 desde la memoria 204 y para proporcionar opcionalmente almacenamiento temporal local para las instrucciones obtenidas. También incluye una rutina de traducción de instrucciones 254 para determinar el tipo de instrucción invitada que se ha obtenido y para traducir la instrucción invitada en una o más instrucciones nativas correspondientes 256. Esta traducción incluye, por ejemplo, identificar la función a ser realizada por la instrucción invitada y elegir la(s) instrucción(instrucciones) nativa(s) para realizar esa función.

Además, el emulador 212 incluye una rutina de control de emulación 260 para hacer que las instrucciones nativas sean ejecutadas. La rutina de control de emulación 260 puede causar a la CPU nativa 202 ejecutar una rutina de instrucciones nativas que emulan una o más instrucciones invitadas obtenidas previamente y, a la terminación de tal ejecución, devolver el control a la rutina de búsqueda de instrucción para emular la obtención de la siguiente instrucción invitada o un grupo de instrucciones invitadas. La ejecución de instrucciones nativas 256 puede incluir datos de carga en un registro desde la memoria 204; almacenar datos de vuelta a memoria desde un registro; o realizar algún tipo de operación aritmética o lógica, como se determina por la rutina de traducción.

Cada rutina, por ejemplo, se implementa en software, que se almacena en memoria y ejecuta por la unidad central de proceso nativa 202. En otros ejemplos, una o más de las rutinas u operaciones se implementan en microprogramas, hardware, software o alguna combinación de los mismos. Los registros del procesador emulado se pueden emular usando registros 210 de la CPU nativa o usando localizaciones en la memoria 204. En las realizaciones, las instrucciones invitadas 250, las instrucciones nativas 256 y el código emulador 212 pueden residir en la misma memoria o se pueden desembolsar entre diferentes dispositivos de memoria.

Como se usa en la presente memoria, los microprogramas incluyen, por ejemplo, el microcódigo, milicódigo y/o macrocódigo del procesador. Incluyen, por ejemplo, las instrucciones a nivel de hardware y/o estructuras de datos usadas en la implementación de código máquina de mayor nivel. En una realización, incluyen, por ejemplo, código propietario que se entrega típicamente como microcódigo que incluye software de confianza o microcódigo específico al hardware subyacente y controla el acceso del sistema operativo al hardware del sistema.

En un ejemplo, una instrucción invitada 250 que se obtiene, traduce y ejecuta es una de las instrucciones descritas en la presente memoria. La instrucción, que es de una arquitectura (por ejemplo, la z/Architecture) se trae de la

memoria, traduce y representa como una secuencia de instrucción nativa 256 de otra arquitectura (por ejemplo, PowerPC, pSeries, xSeries, Intel, etc.). Entonces se ejecutan estas instrucciones nativas.

En una realización, las instrucciones descritas en la presente memoria son instrucciones de vector, que son parte de una facilidad de vector, proporcionadas según un aspecto de la presente invención. La facilidad de vector proporciona, por ejemplo, vectores dimensionados fijos que oscilan de uno a dieciséis elementos. Cada vector incluye datos sobre los que se opera por instrucciones de vector definidas en la facilidad. En una realización, si un vector está compuesto de múltiples elementos, entonces cada elemento se procesa en paralelo con los otros elementos. La terminación de instrucción no ocurre hasta que el procesamiento de todos los elementos está completo.

Como se describe en la presente memoria, las instrucciones de vector se pueden implementar como parte de varias arquitecturas, incluyendo, pero no limitado a, la z/Architecture, PowerPC, Intel, etc. Aunque una realización descrita en la presente memoria es para la z/Architecture, las instrucciones de vector y uno o más aspectos de la presente invención se pueden basar en muchas otras arquitecturas. La z/Architecture es solamente un ejemplo.

En una realización en la que la facilidad de vector se implementa como parte de la z/Architecture, para usar los registros e instrucciones de vector, un control de habilitación de vector y un control de registro en un registro de control especifico (por ejemplo, el registro de control 0) se fija a, por ejemplo, uno. Si la facilidad de vector se instala y se ejecuta una instrucción de vector sin el conjunto de controles de habilitación, se reconoce una excepción de datos. Si la facilidad de vector no se instala y se ejecuta una instrucción de vector, se reconoce una excepción de operación.

Los datos de vector aparecen en el almacenamiento, por ejemplo, en la misma secuencia de izquierda a derecha que otros formatos de datos. Los bits de un formato de datos que se numeran 0-7 constituyen el octeto en la localización de octeto de más a la izquierda (numerado más bajo) en el almacenamiento, los bits 8-15 forman el octeto en la siguiente localización secuencial y así sucesivamente. En un ejemplo adicional, los datos de vector pueden aparecer en el almacenamiento en otra secuencia, tal como de derecha a izquierda.

Muchas de las instrucciones de vector dotadas con la facilidad de vector tienen un campo de bits especificados. Este campo, conocido como el bit de extensión de registro o RXB, incluye el bit más significativo para cada uno de los operandos designados de registro de vector. Los bits para designaciones de registro no especificadas por la instrucción van a ser reservados y fijados a cero.

En un ejemplo, el campo RXB incluye cuatro bits (por ejemplo, los bits 0-3) y los bits se definen, como sigue:

0 – Bit más significativo para la primera designación de registro de vector de la instrucción.

1 – Bit más significativo para la segunda designación de registro de vector de la instrucción, en su caso.

2 – Bit más significativo para la tercera designación de registro de vector de la instrucción, en su caso.

3 – Bit más significativo para la cuarta designación de registro de vector de la instrucción, en su caso.

Cada bit se fija a cero o uno mediante, por ejemplo, el ensamblador dependiendo del número de registro. Por ejemplo, para los registros 0-15, el bit se fija a 0; para los registros 16-31, el bit se fija a 1, etc.

En una realización, cada bit RXB es un bit de extensión para una localización particular en una instrucción que incluye uno o más registros de vector. Por ejemplo, en una o más instrucciones de vector, el bit 0 de RXB es un bit de extensión para la localización 8-11, que se asigna, por ejemplo, a V_1; el bit 1 de RXB es un bit de extensión para la localización 12-15, que se asigna, por ejemplo, a V_2; y así sucesivamente.

En una realización adicional, el campo RXB incluye bits adicionales y más de un bit se usa como una extensión para cada vector o localización.

Una instrucción, proporcionada según un aspecto de la presente invención que incluye el campo RXB es una instrucción Encontrar Elemento Igual de Vector, un ejemplo de la cual se representa en la FIG. 3. En un ejemplo, la instrucción Encontrar Elemento Igual de Vector 300 incluye campos de código de operación 302a (por ejemplo, los bits 0-7), 302b (por ejemplo, los bits 40-47) que indican una operación Encontrar Elemento Igual de Vector; un primer campo de registro de vector 304 (por ejemplo, los bits 8-11) usado para designar un primer registro de vector (V_1); un segundo campo de registro de vector 306 (por ejemplo, los bits 12-15) usados para designar un segundo registro de vector (V_2); un tercer campo de registro de vector 308 (por ejemplo, los bits 16-19) usados para designar un tercer registro de vector (V_3); un primer campo de máscara (M_5) 310 (por ejemplo, los bits 24-27); un segundo campo de máscara (M_4) 312 (por ejemplo, los bits 32-35); y el campo RXB 314 (por ejemplo, los bits 36-39). Cada uno de los campos 304-314, en un ejemplo, está separado y es independiente del(de los) campo(s) de código de operación. Además, en una realización, están separados y son independientes uno de otro; no obstante, en otras realizaciones, se puede combinar más de un campo. Información adicional sobre el uso de estos campos se describe más adelante.

En un ejemplo, los bits seleccionados (por ejemplo, los primeros dos bits) del código de operación designado por el campo de código de operación 302a especifican la longitud y formato de la instrucción. En este ejemplo particular, los bits seleccionados indican que la longitud es tres medias palabras y el formato es un registro de vector y operación de registro con un código de operación extendido. Cada uno de los campos de vector (V), junto con su bit de extensión de registro correspondiente especificado por RXB, designa un registro de vector. En particular, para registros de vector, el registro que contiene el operando se especifica usando, por ejemplo, un campo de cuatro bit del campo de registro con la adición del bit de registro de extensión de registro (RXB) como el bit más significativo. Por ejemplo, si el campo de cuatro bits es 0110 y el bit de extensión es 0, entonces el campo de cinco bits 00110 indica el número de registro 6.

El número de subíndice asociado con un campo de la instrucción indica el operando al que se aplica el campo. Por ejemplo, el número de subíndice 1 asociado con el registro de vector V_1 indica el primer operando y así sucesivamente. Un operando de registro es un registro en longitud, que es, por ejemplo, de 128 bits.

El campo M_4 que tiene, por ejemplo, cuatro bits, 0-3, especifica un control de tamaño de elemento en, por ejemplo, los bits 1-3. El control de tamaño de elemento especifica el tamaño de los elementos en los operandos de registro de vector. Como ejemplos, el control de tamaño de elemento especifica o bien un octeto, media palabra (por ejemplo, 2 octetos) o bien una palabra (por ejemplo, 4 octetos). Por ejemplo, un 0 indica un octeto; un 1 indica una media palabra; y un 2 indica una palabra, también conocida como, palabra completa. Si se especifica un valor reservado, se reconoce una excepción de especificación.

El campo M_5, es, por ejemplo, un campo de cuatro bits, los bits 0-3, que incluye, por ejemplo:

Un campo de búsqueda de ceros (ZS, bit 2), el cual si es uno, cada elemento del segundo operando se compara con cero (o nulo). (En un ejemplo adicional, es cada elemento del tercer operando u otro operando el que se compara con cero); y

Un campo de conjunto de códigos de condición (CC, bit 3), que si es cero, el código de condición no está fijado y permanece sin cambios. Si es uno, el código de condición se fija como se especifica más adelante, como ejemplo:

0 – Si el bit de búsqueda de ceros está fijado, la comparación detectó un elemento cero en el segundo operando en un elemento con un índice menor que cualquier comparación igual.

1 – La comparación detectó una coincidencia entre el segundo y tercer operandos en algún elemento. Si el bit de búsqueda de ceros está fijado, esta coincidencia ocurrió en un elemento con un índice menor o igual que el elemento de comparación de ceros.

2 - --

3- Ningún elemento comparado es igual.

En la ejecución de una realización de la instrucción Encontrar Elemento Igual de Vector, procediendo en una realización de izquierda a derecha, los elementos enteros binarios sin signo del segundo operando (incluido dentro del registro designado por V_2 más su bit RXB) se comparan con los elementos enteros binarios sin signo correspondientes del tercer operando (incluidos dentro del registro designado por V_3 más su bit RXB). Si dos elementos son iguales, un índice de octeto del primer octeto del elemento igual de más a la izquierda se coloca en un octeto seleccionado (por ejemplo, el octeto 7) del primer operando (incluido dentro de un registro designado por V_1 más su bit RXB). Los ceros se almacenan en los octetos restantes del primer operando.

Por ejemplo, si el tamaño de elemento es un octeto, entonces el índice de octeto del elemento igual de más a la izquierda se devuelve (por ejemplo, si hay 16 elementos, 0-15 y el elemento 6 es igual, entonces se devuelve el índice de octeto 6). De manera similar, si el tamaño de elemento es de media palabra y hay 8 elementos, 0-7 y ambos octetos 6 o 7 del elemento tres son iguales, entonces se devuelve el índice de octeto 6. Del mismo modo, si el tamaño de elemento es de palabra completa y hay cuatro elementos, 0-3 y todos los octetos 4-7 del elemento uno son iguales, se devuelve el índice de octeto 4.

Si no se encuentran octetos que sean iguales o cero si la se fija la búsqueda de ceros, entonces un índice igual al tamaño de vector (por ejemplo, número de bytes; por ejemplo, 16) se almacena en el octeto especificado (por ejemplo, el octeto 7) del primer operando. Los ceros se almacenan en los octetos restantes.

Si el bit de búsqueda de ceros se fija en el campo M_5, entonces cada elemento en el segundo operando (o en otra realización, otro operando, tal como el operando tres) también se compara para igualdad con cero (o nulo; final de la cadena). Si se encuentra un elemento cero en el segundo operando antes de que se encuentre que cualquier otro elemento del segundo o tercer operandos sean iguales, el índice de octeto del primer octeto del elemento encontrado que sea cero se almacena en el octeto especifico (por ejemplo, el octeto 7) del primer operando y se almacenan ceros en todas las otras localizaciones de octeto. Si la marca de conjunto de códigos de condición es uno, entonces el código de condición se fija a cero.

En una realización, la comparación de los elementos se realiza en paralelo. Por ejemplo, si los registros de vector que se comparan son de 16 octetos de longitud, entonces se comparan en paralelo 16 octetos. Además, en una realización, la dirección de los vectores, izquierda a derecha o derecha a izquierda, se proporciona en un tiempo de ejecución. Por ejemplo, la instrucción accede a un registro, control de estado u otra entidad que indica la dirección de procesamiento como o bien de izquierda a derecha o bien de derecha a izquierda, como ejemplos. En una realización, este control de dirección no se codifica como parte de la instrucción, sino que se proporciona a la instrucción en el tiempo de ejecución.

En una realización adicional, la instrucción no incluye el campo RXB. En su lugar, no se usa ninguna extensión o la extensión se proporciona de otra manera, tal como desde un control exterior de la instrucción o proporciona como parte de otro campo de la instrucción.

Detalles adicionales con respecto a una realización del procesamiento de la instrucción Encontrar Elemento Igual de Vector se describen con referencia a la FIG. 4. En un ejemplo, un procesador del entorno informático está realizando esta lógica.

Inicialmente, se hace una determinación en cuanto a si va a ser realizada una búsqueda de nulo (también conocido como elemento cero, final de la cadena, terminador, etc.), CONSULTA 400. Si va a ser realizada una búsqueda de nulo, se hace una comparación contra caracteres nulos, es decir, para elementos cero, PASO 402 y el resultado se saca al nullidx 403. Por ejemplo, si el tamaño de elemento es octetos y un elemento cero se encuentra en el octeto 5, el índice del octeto en el que se encuentra el elemento cero (por ejemplo, 5) se coloca en nullidx. De manera similar, si el tamaño de elemento es media palabra y hay 8 elementos, 0-7 y el elemento tres (es decir, los octetos 6-7) es cero, entonces 6 (para el índice de octeto 6) se coloca en nullidx. Del mismo modo, si el tamaño de elemento es la palabra completa y hay cuatro elementos, 0-3 y el elemento uno (es decir, los octetos 4-7) es cero, entonces 4 (para el índice de octeto 4) se coloca en nullidx. Si no se encuentra ningún elemento nulo, entonces, en un ejemplo, el tamaño del vector (por ejemplo, en octetos; por ejemplo, 16) se coloca en nullidx.

Adicionalmente o si no va a ser realizada ninguna búsqueda de nulos, una pluralidad de comparaciones (por ejemplo, 16) se realizan en paralelo comparando A con B en base a una operación de comparación, PASO 404. En un ejemplo, A son los contenidos del segundo operando y B son los contenidos del tercer operando y la operación de comparación es igual.

Un resultado de la comparación se almacena en una variable 406, conocida como o bien un índice izquierdo, cmpidxl o un índice derecho, cmpidxr, dependiendo de si la búsqueda es desde la izquierda o la derecha. Por ejemplo, si la comparación es una comparación de igual, la búsqueda es de izquierda a derecha y la comparación provoca una o más igualdades, el índice asociado con el primer octeto del elemento igual más bajo se coloca en cmpidxl. Como ejemplo, si el tamaño de elemento es octetos y hay 16 elementos en el vector (0-15) y se encuentra una igualdad en el elemento 6, entonces 6 se almacena en cmpidxl. De manera similar, si el tamaño de elemento es medias palabras y hay 8 elementos en el vector (0-7) y se encuentra una igualdad en el elemento 3 (por ejemplo, en el octeto 6 o 7), se devuelve el índice del primer octeto del elemento (octeto 6). Del mismo modo, si el tamaño del elemento es palabra completa y hay cuatro elementos (0-3) y se encuentra una igualdad en el elemento 1 (por ejemplo, en el octeto 4-7), se devuelve el índice del primer octeto del elemento (octeto 4). Si no hay comparaciones iguales, entonces, en una realización, cmpidxl o cmpidxr, dependiendo de la dirección de la comparación, se fija igual al tamaño del vector (por ejemplo, en octetos; por ejemplo, 16).

A partir de entonces, se hace una determinación en cuanto a si la búsqueda es desde la izquierda o la derecha, CONSULTA 408. Si la búsqueda es desde la izquierda, una variable cmpidx se fija igual a cmpidxl, PASO 410; de otro modo, cmpidx se fija igual a cmpidxr, PASO 412.

Posterior a ajustar cmpidx, se hace una determinación en cuanto a si una búsqueda fue realizada para caracteres nulos, CONSULTA 414. Si no hubo búsqueda de caracteres nulos, entonces una variable, idx, se fija, por ejemplo o el índice de comparación, cmpidx, PASO 416. Si se busca un nulo, entonces idx se fija al mínimo del índice de comparación o el índice de nulo, nullidx, PASO 418. Esto concluye el procesamiento.

Un ejemplo de lógica de bloque para el procesamiento de la FIG. 4 se representa en la FIG. 5. En este ejemplo, hay dos entradas, Vector B 500 y Vector A 502. Ambas entradas se introducen a la lógica de comparación 504, que realiza las comparaciones (por ejemplo, igual) en paralelo. Además, una entrada, Vector A, también se introduce a la lógica de detección de ceros 506, que realiza procesamiento de nulos.

La salida de la lógica de comparación, idxL o idxR 508, se introduce a la lógica de determinación de resultado 512, así como a la salida de la lógica de detección de ceros, nullidx 510. La lógica de determinación de resultado también toma como entrada los siguientes controles: izquierda/derecha 514 que indica la dirección de la búsqueda; detección de ceros 516 que indica si el procesamiento de nulos va a ser realizado; y tamaño de elemento 518 que proporciona el tamaño de cada elemento (por ejemplo, octeto, media palabra, palabra); y produce un índice resultante 520, resultidx, que se almacena en un vector de salida 522 (por ejemplo, en el octeto 7).

Además, la lógica de determinación de resultado incluye procesamiento de código de condición 523, que opcionalmente saca un código de condición 524.

Ejemplos de seudo código para lógica de comparación 504 es como sigue:

```
idxL = 16; idxR = 16
Para i = 0 a vector_length
    Si A[i] = a B[i] ENTONCES
        idxL = i
        Hecho
Para i = vector_length hasta 0
    Si A[i] = a B[i] ENTONCES
        idxR = i
        Hecho
```

Como se muestra, la variable idxL o idxR, dependiendo de la dirección, se inicializa al tamaño del vector (por ejemplo, en número de octetos; por ejemplo, 16). Entonces, cada elemento del Vector A se compara con un elemento correspondiente del Vector B. En un ejemplo, las comparaciones son comparaciones de octetos, así que se hace una comparación para cada uno de los 16 octetos (i). En este ejemplo, la operación de comparación es igual y si se encuentra una igualdad, el índice del octeto se almacena en idxL si se busca desde la izquierda o idxR si se busca desde la derecha.

Un ejemplo de seudo código para lógica de detección de ceros 506 es como sigue:

```
nullidx = 16
Para j = 0 a vector_length
Si A[j] == 0 ENTONCES
    nullidx = j x element_size
    Hecho
```

Como se muestra, cada elemento (j) del vector se prueba para ver si es igual a cero. Si un elemento es igual a cero, nullidx se fija igual al índice de ese elemento tantas veces el tamaño del elemento. Por ejemplo, si el tamaño del elemento es medias palabras (2 octetos) y se detecta un carácter nulo en el elemento 3, 3 se multiplica por 2 y nullidx se fija a 6, que representa el octeto 6. De manera similar, si el tamaño de elemento es la palabra completa (4 octetos) y se detecta un carácter nulo en el elemento 3, 3 se multiplica por 4 y nullidx se fija a 12.

Del mismo modo, un ejemplo de seudo código para lógica de determinación de resultado 512 es como sigue:

```
Si Izquierda/Derecha = Izquierda ENTONCES
    cmpidx = idxL
DE OTRO MODO
    cmpidx = idxR
SI zero_detect = ENCENDIDO ENTONCES
    resultidx = min(cmpidx, nullidx)
    SI set_CC = ENCENDIDO && nullidx <= cmpidx ENTONCES
        CC = 0
DE OTRO MODO
    resultidx = cmpidx
SI element_size = octeto ENTONCES element_size_mask = ‘11111’b
SI element_size = 2octetos ENTONCES element_size_mask = ‘11110’b
SI element_size = 4octetos ENTONCES element_size_mask = ‘11100’b
```

```
resultidx = resultidx & element_size_mask
SI SetCC = ENCENDIDO ENTONCES
    SI nullidx < cmpidx ENTONCES
        CC = 0
    DE OTRO MODO SI cmpidx < 16 ENTONCES
        CC = 1
    DE OTRO MODO
        CC = 3
```

Como se muestra, si el control izquierda/derecha indica izquierda, entonces cmpidx se fija igual a idxL; de otro modo, cmpidx se fija igual a idxR. Además, si el indicador de detección de cero está encendido, entonces resultidx se fija igual al mínimo de cmpidx o nullidx; y si el control de conjunto de códigos de condición está encendido y cmpidx es mayor que nullidx, el código de condición se fija a cero. De otro modo, si la detección de ceros no está encendida, resultidx se fija igual a cmpidx.

Además, si el tamaño de elemento es igual a un octeto, entonces una máscara de tamaño de elemento se fija a '11111'; si el tamaño de elemento es igual a 2 octetos, la máscara se fija a '11110' y si el tamaño de elemento es igual a 4 octetos, la máscara se fija a '11100'.

A partir de entonces, resultidx se fija igual a resultidx aplicando la función AND con la máscara de tamaño de elemento. Por ejemplo, si el tamaño de elemento es media palabra y el octeto 7 es resultidx, entonces resultidx = 00111 AND 11110, que proporciona 00110; así resultidx se fija igual a 6 (es decir, 00110 en binario), que es el primer octeto del elemento.

Adicionalmente, se fija opcionalmente un código de condición. Si el control de código de condición fijado de la instrucción se fija a encendido, entonces se proporciona un código de condición; de otro modo, no se fija ningún código de condición. Como ejemplos, si el control se fija a encendido, entonces si nullidx < cmpidx, el código de condición se fija a 0. De otro modo, si cmpidx < 16, entonces el código de condición se fija a 1; de otro modo, el código de condición se fija a 3.

Descrito anteriormente está un ejemplo de una instrucción de vector usada para facilitar el procesamiento de datos de caracteres. Como se describe en la presente memoria, para un vector de 128 bit, la lógica de comparación solamente realiza comparaciones de 16 octetos, más que, por ejemplo, 256 comparaciones. Esto proporciona escalado para mayores vectores. Además, un control izquierda/derecha se puede proporcionar como un valor de tiempo de ejecución y no se codifica dentro de la instrucción. Aún además, el valor devuelto como el resultado es una posición de octeto, más que un índice de elemento. Además, se soportan comparaciones de 4 octetos junto con comparaciones de 1 octeto y de 2 octetos.

En una realización, hay 32 registros de vectores y otros tipos de registros pueden correlacionarse con un cuadrante de los registros de vectores. Por ejemplo, como se muestra en la FIG. 6, si hay un fichero de registro 600 que incluye 32 registros de vectores 602 y cada registro es de 128 bit de longitud, entonces los 16 registros de punto flotante 604, que son de 64 bit de longitud, pueden solapar los registros de vector. De esta manera, como ejemplo, cuando se modifica un registro de punto flotante 2, entonces también se modifica el registro de vector 2. También son posibles otras correlaciones para otros tipos de registros.

En un aspecto adicional de la invención, otra instrucción dotada con la facilidad del vector y usada según un aspecto de la presente invención es una instrucción de Encontrar Cualquier Igual de Vector, en la que todos los caracteres (o un subconjunto en otra realización) en un vector de entrada se comparan con cada carácter (o caracteres seleccionados) en otro vector de entrada. La salida se graba o bien como una máscara o bien un índice a un carácter que coincide. Esto es útil, por ejemplo, cuando se analizan sintácticamente datos de caracteres, tales como cadenas de datos. Cuando se analiza sintácticamente, una operación que se realiza es para buscar varios caracteres especiales antes de proceder al siguiente paso de análisis sintáctico. Esta instrucción permite buscar varios caracteres a la vez.

Un ejemplo de la instrucción Encontrar Cualquier Igual de Vector se representa en la FIG. 7, en la que en una realización, una instrucción Encontrar Cualquier Igual de Vector 700 incluye, por ejemplo, campos de código de operación 702a (por ejemplo, los bits 0-7), 702b (por ejemplo, los bits 40-47) que indican una operación Encontrar Cualquier Igual de Vector; un primer campo de registro de vector 704 (por ejemplo, los bits 8-11) usado para designar un primer registro de vector (V_1); un segundo campo de registro de vector 706 (por ejemplo, los bits 12-15) usado para designar un segundo registro de vector (V_2); un tercer campo de registro de vector 708 (por ejemplo, bits 16-19) usado para designar un tercer registro de vector (V_3); un primer campo de máscara (M_5) 710 (por ejemplo, los bits 24-27); un segundo campo de máscara (M_4) (por ejemplo, los bits 32-35) 712; y un campo RXB 714 (por

ejemplo, los bits 36-39). Cada uno de los campos 704-714, en un ejemplo, está separado y es independiente del(de los) campo(s) de código de operación. Además, en una realización, están separados y son independientes unos de otros; no obstante, en otras realizaciones, se puede combinar más de un campo. Información adicional sobre el uso de estos campos se describe más adelante.

Similar a lo anterior, en este ejemplo, los bits seleccionados (por ejemplo, los primeros dos bits) del código de operación designado por el campo de código de operación 702a especifican la longitud y formato de la instrucción. En este ejemplo, los bits seleccionados indican que la longitud es tres medias palabras y el formato es un registro de vector – y la operación de registro con un campo de código de operación extendido. Cada uno de los campos de vector (V), junto con su bit de extensión correspondiente especificado por RXB, designa un registro de vector. En particular, para registros de vector, el registro que contiene el operando se especifica usando, por ejemplo, un campo de cuatro bits del campo de registro con la adición del bit de extensión de registro (RXB) como el bit más significativo. Por ejemplo, si el campo de cuatro bits es 0110 y el bit de extensión es 0, entonces el campo de cinco bits 00110 indica el número de registro 6.

El número de subíndice asociado con un campo de la instrucción indica el operando al que se aplica el campo. Por ejemplo, el número de subíndice 1 asociado con el registro de vector V_1 indica el primer operando y así sucesivamente. Un operando de registro es un registro en longitud, que es, por ejemplo, de 128 bits.

El campo M_4 que tiene, por ejemplo, cuatro bits, 0-3, especifica el control de tamaño de elemento en, por ejemplo, los bits 1-3. El control de tamaño de elemento especifica el tamaño de los elementos en los operandos de registro de vector. En un ejemplo, el control de tamaño de elemento especifica un octeto, media palabra (por ejemplo, 2 octetos) o palabra (por ejemplo, 4 octetos). Por ejemplo, un 0 indica un octeto; un 1 indica una media palabra; y un 2 indica una palabra, también conocida como una palabra completa. Si se especifica un valor reservado, se reconoce una excepción de especificación.

El campo M_5 es, por ejemplo, un campo de cuatro bits, bits 0-3, que incluye, por ejemplo:

Un campo de tipo resultado (RT, bit 1), que si es cero, cada elemento resultante es una máscara de toda la gama de comparaciones sobre ese elemento. Si es uno, un índice de octeto se almacena en un octeto especificado (por ejemplo, el octeto 7) del primer operando y se almacenan ceros en todos los otros elementos;

Un campo de búsqueda de ceros (ZS, bit 2), que si es uno, cada elemento del segundo operando (u otro operando) se compara con cero; y

Un campo de conjunto de códigos de condición (CC, bit 3), que si es cero, el código de condición no se fija y permanece sin cambios. Si es uno, el código de condición se fija como sigue, en un ejemplo:

0 – Si el bit de búsqueda de ceros está fijado, no hubo coincidencias en un elemento de índice inferior distinto de cero en el segundo operando.

1 – Algunos elementos del segundo operando coinciden con al menos un elemento en el tercer operando;

2 – Todos los elementos del segundo operando coinciden con al menos un elemento en el tercer operando; y

3 – Ningún elemento del segundo operando coincide con ningún elemento en el tercer operando.

En ejecución de una realización de la instrucción Encontrar Cualquier Igual de Vector, procediendo, en un ejemplo, de izquierda a derecha, cada elemento entero binario sin signo del segundo operando (incluido dentro de un registro designado por V_2 más RXB) se compara para igualdad con cada elemento entero binario sin signo del tercer operando (incluido dentro de un registro designado por V_3 más RXB) y opcionalmente, cero, si la marca de búsqueda de ceros se fija en el campo M_5.

Si la marca de tipo resultado en el campo M_5 es cero, entonces para cada elemento en el segundo operando que coincide con cualquier elemento en el tercer operando u opcionalmente cero, las posiciones de bits del elemento correspondiente en el primer operando (incluido dentro de un registro designado por V_1 más RXB) se fijan a unos; de otro modo, se fijan a cero.

Si la marca de tipo resultado en el campo M_5 es uno, entonces un índice de octeto del elemento de más a la izquierda (por ejemplo, un índice de octeto del primer octeto del elemento) en el segundo operando que coincide con un elemento en el tercer operando o cero se almacena en el octeto especificado (por ejemplo, el octeto 7) del primer operando.

Si la marca de tipo resultado en el campo M_5 es uno y no se encuentran octetos que sean iguales o cero si la marca de búsqueda de ceros está fijada, entonces un índice igual al tamaño del vector (por ejemplo, número de octetos; por ejemplo, 16) se almacena en el octeto especificado (por ejemplo, el octeto 7) del primer operando.

En una realización, la dirección de los vectores, izquierda a derecha o derecha a izquierda, se proporciona en el tiempo de ejecución. Por ejemplo, la instrucción accede a un registro, control de estado u otra entidad que indica la

dirección de procesamiento como o bien izquierda a derecha o bien derecha a izquierda, como ejemplos. En una realización, este control de dirección no se codifica como parte de la instrucción, sino que se proporciona a la instrucción en el tiempo de ejecución.

En una realización adicional, la instrucción no incluye el campo RXB. En su lugar, no se usa ninguna extensión o la extensión se proporciona de otra manera, tal como desde un control fuera de la instrucción o proporcionado como parte de otro campo de la instrucción.

Detalles adicionales con respecto a una realización de la instrucción Encontrar Cualquier Igual de Vector se describen con referencia a la FIG. 8. En un ejemplo, un procesador del entorno informático está realizando esta lógica.

Inicialmente, se inicializan variables conocidas como zeroidx 802, resultidx 804 y resultmask 806, PASO 800. Por ejemplo, zeroidx se fija igual al tamaño del segundo operando (por ejemplo, 16); resultidx se fija igual al tamaño del segundo operando (por ejemplo, 16); y resultmask se fija igual a todo ceros. Resultmask, en una realización, incluye 128 bits que corresponden a los 128 bits del segundo operando.

A partir de entonces, se carga un carácter (es decir, un elemento) desde un operando, conocido en la presente memoria como opA, que es, por ejemplo, el segundo operando de la instrucción, PASO 808. A partir de entonces, se hace una determinación en cuanto a si el campo de búsqueda de ceros se fija indicando una búsqueda de ceros y zeroidx se fija igual a 16, CONSULTA 810. Si es así, se realiza una búsqueda de ceros, PASO 812 y el resultado se saca a zeroidx 802 y resultmask 806. Por ejemplo, un índice de octeto del octeto de más a la izquierda del elemento cero se indica en zeroidx y los bits que corresponden a ese elemento en resultmask se fijan a uno. Por ejemplo, para zeroidx, si el tamaño de elemento es octetos y un elemento cero se encuentra en el octeto 5, el índice del octeto en el que se encuentra el elemento cero (por ejemplo, 5) se coloca en zeroidx. De manera similar, si el tamaño de elemento es media palabra y hay 8 elementos, 0-7 y el elemento tres (es decir, los octetos 6-7) es cero, entonces 6 (para índice de octeto 6) se coloca en zeroidx. Del mismo modo, si el tamaño de elemento es palabra completa y hay cuatro elementos, 0-3 y el elemento uno (es decir, los octetos 4-7) es cero, entonces 4 (para índice de octeto 4) se coloca en zeroidx. Si no se encuentra ningún elemento nulo, entonces, en un ejemplo, el tamaño del vector (por ejemplo, en octetos; por ejemplo, 16) se coloca en zeroidx.

A partir de entonces o si una búsqueda de ceros no va a ser realizada, entonces el carácter cargado se compara con cada carácter en un operando, conocido en la presente memoria como opB, que es, por ejemplo, el tercer operando en la instrucción, PASO 818. Si hay una coincidencia, la coincidencia se indica en resultidx 804 y resultmask 806. Por ejemplo, resultidx se fija igual al primer octeto del elemento que coincide y los bits en resultmask que corresponde al elemento se fijan iguales a uno. Como ejemplo, para resultidx, si el tamaño de elemento es octetos y hay 16 elementos en el vector (0-15) y se encuentra una igualdad en el elemento 6, entonces 6 se almacena en resultidx. De manera similar, si el tamaño de elemento es medias palabras y por lo tanto hay 8 elementos en el vector (0-7) y se encuentra una igualdad en el elemento 3 (por ejemplo, en el octeto 6 o 7), se devuelve el índice del primer octeto del elemento (octeto 6). Del mismo modo, si el tamaño de elemento es palabra completa y hay cuatro elementos (0-3) y se encuentra una igualdad en el elemento 1 (por ejemplo, en el octeto 4-7), se devuelve el índice del primer octeto del elemento (octeto 4). Si no hay comparaciones iguales, entonces, en una realización, resultidx se fija igual al tamaño del vector (por ejemplo, en octetos; por ejemplo, 16).

A partir de entonces, se hace una determinación en cuanto a si se ha alcanzado el final de opA, CONSULTA 830. Si no, cuando se aumenta la variable i, por ejemplo, en uno, PASO 832 y el procesamiento continúa con el PASO 808. De otro modo, el procesamiento continúa con la determinación de si el tipo de resultado es igual a cero, CONSULTA 834. Si el tipo de resultado es igual a cero, entonces un operando que se conoce en la presente memoria como opC, que es, por ejemplo, el primer operando de la instrucción, se fija igual a resultmask, PASO 836. De otro modo, un octeto especificado (por ejemplo, el octeto 7) de opC se fija igual al mínimo de resultidx y zeroidx (y los otros octetos se fijan a cero), PASO 838.

Posterior a colocar el resultado en opC, se hace una determinación en cuanto a si el campo de conjunto de códigos de condición indica que el código de condición va a ser fijado, CONSULTA 840. Si un campo de conjunto de códigos de condición está fijado a uno, entonces el código de condición se fija, PASO 842. Por ejemplo, el código de condición se fija a cero si el campo ZS está fijado y no hubo coincidencias en un elemento de índice inferior distinto de cero en el segundo operando; fijado a uno si algunos elementos del segundo operando coinciden con al menos un elemento en el tercer operando; fijado a 2 si todos los elementos del segundo operando coincidieron con al menos un elemento en el tercer operando; y fijado a 3 si ningún elemento del segundo operando coincide con ningún elemento en el tercer operando. De otro modo, si el campo de conjuntos de códigos de condición es igual a cero, entonces no se fija ningún código de condición.

Como se describe en la presente memoria, en una realización, se proporciona una instrucción Encontrar Cualquier Igual de Vector que es capaz de alternar entre proporcionar y no proporcionar una búsqueda de ceros y es capaz de devolver condicionalmente un índice de octeto de o bien un elemento cero o bien un elemento igual. Este índice de octeto es el primer octeto del elemento que se notifica. Si se busca un elemento cero, entonces se puede proporcionar la posición de ese elemento cero. De esta manera, se proporciona una instrucción que tiene un código

de operación particular en la que esa instrucción es capaz de realizar una comparación con una búsqueda de ceros o una comparación sin búsqueda de ceros.

Descritos anteriormente están ejemplos de instrucciones de vector usadas para facilitar procesamiento de datos de caracteres. Como se describe en la presente memoria, para la instrucción Encontrar Elemento Igual de Vector con vectores de 128 bits, la lógica de comparación solamente realiza comparaciones de 16 octetos, en lugar de, por ejemplo, 256 comparaciones. Esto proporciona escalado para vectores mayores. Además, o bien para la instrucción Encontrar Elemento Igual de Vector o bien Encontrar Cualquier Igual de Vector, un control de izquierda/derecha se puede proporcionar como un valor de tiempo de ejecución y no codificar dentro de la instrucción. Aún además, el valor devuelto como el resultado es una posición de octeto, en lugar de un índice de elemento. Además, se soportan comparaciones de 4 octetos junto con comparaciones de 1 octeto y 2 octetos.

Según un aspecto de la presente invención, se proporciona opcionalmente un código de condición en base a un control proporcionado con la instrucción. Permitiendo que el código de condición sea fijado, se facilita la programación de una instrucción.

En una realización adicional, la búsqueda de ceros no es una condición, pero en su lugar, se realiza cuando se ejecuta la instrucción. En base a o en respuesta a ejecutar la instrucción, se realiza la búsqueda de ceros y, en un ejemplo, se devuelve la posición (por ejemplo, el índice de octeto) del elemento cero y/o la posición (por ejemplo, el índice de octeto) del primer elemento coincidente. En una realización, el número de comparaciones que se realizan, con independencia de la realización, para la instrucción Encontrar Elemento Igual de Vector corresponde al número de octetos del vector. Por ejemplo, si el vector que es buscado o comparado es de 16 octetos, entonces se realizan a lo sumo 16 comparaciones, por ejemplo, en paralelo. En una realización adicional, una vez que se encuentra una coincidencia o elemento cero, la comparación cesa.

En la presente memoria, se usan intercambiablemente memoria, memoria principal, almacenamiento y almacenamiento principal, a menos que se señale de otro modo explícitamente o por el contexto.

Detalles adicionales relativos a la facilidad de vector, incluyendo ejemplos de otras instrucciones, se proporcionan como parte de esta Descripción Detallada además más adelante.

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

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

Con referencia ahora a la FIG. 9, en un ejemplo, un producto de programa de ordenador 900 incluye, por ejemplo, uno o más medios de almacenamiento legibles por ordenador no transitorios 902 para almacenar medios o lógica de código de programa legibles por ordenador 904 en los mismos para proporcionar y facilitar uno o más aspectos de la presente invención.

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

Un código de programa de ordenador para llevar a cabo operaciones para uno o más aspectos de la presente invención se puede escribir en cualquier combinación de uno o más lenguajes de programación, incluyendo un lenguaje de programación orientado a objeto, tal como Java, Smalltalk, C++ o similares y lenguajes de programación de procedimiento convencional, tales como el lenguaje de programación “C”, ensamblador o lenguajes de programación similares. El código de programa puede ejecutarse enteramente en el ordenador del usuario, parcialmente en el ordenador del usuario, como un paquete software autónomo, parcialmente en el ordenador del

usuario y parcialmente en un ordenador remoto o enteramente en el ordenador o servidor remoto. En este último escenario, el ordenador remoto se puede conectar al ordenador del usuario a través de cualquier tipo de red, incluyendo una red de área local (LAN) o una red de área extensa (WAN) o la conexión se puede hacer a un ordenador externo (por ejemplo, a través de Internet usando un Proveedor de Servicios de Internet).

Uno o más aspectos de la presente invención se describen en la presente memoria con referencia a ilustraciones de diagrama de flujo y/o diagramas de bloques de métodos, aparatos (sistemas) y productos de programa de ordenador según realizaciones de la invención. Se entenderá que cada bloque de las ilustraciones de diagrama de flujo y/o diagramas de bloques y combinaciones de bloques en las ilustraciones de diagrama de flujo y/o diagramas de bloques, se pueden implementar por instrucciones de programa de ordenador. Estas instrucciones de programa de ordenador se pueden proporcionar a un procesador de un ordenador de propósito general, ordenador de propósito especial u otro aparato de procesamiento de datos programable para producir una máquina, de manera que las instrucciones, que se ejecutan a través del procesador del ordenador u otro aparato de procesamiento de datos programable, crean medios para implementar las funcionalidades/actos especificados en el diagrama de flujo y/o bloque o bloques del diagramas de bloques,

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

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

El diagrama de flujo y los diagramas de bloques en las figuras ilustran la arquitectura, funcionalidad y operación de implementaciones posibles de sistemas, métodos y productos de programa de ordenador según varias realizaciones de uno o más aspectos 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 código, que comprende una o más instrucciones ejecutables para implementar la(s) función(funciones) lógica(s) especificada(s). También se debería señalar que, en algunas implementaciones alternativas, las funciones señaladas en el bloque pueden ocurrir fuera del orden señalado en las figuras. Por ejemplo, dos bloques mostrados en sucesión se pueden ejecutar, de hecho, sustancialmente concurrentemente o los bloques algunas veces se pueden ejecutar en el orden inverso, dependiendo de la funcionalidad implicada. También se señalará que cada bloque de los diagramas de bloques y/o ilustración de diagrama de flujo y combinaciones de bloques en los diagramas de bloques y/o ilustración de diagrama de flujo, se puede implementar por sistemas basados en hardware de propósito especial que realizan las funciones o actos especificados o combinaciones de instrucciones hardware y de ordenador de propósito especial.

Además de lo anterior, uno o más aspectos de la presente invención se pueden proporcionar, ofrecer, desplegar, gestionar, servir, etc. por un proveedor de servicios que ofrezca gestión de entornos de cliente. Por ejemplo, el proveedor de servicios puede crear, mantener, soportar, etc. código de ordenador y/o una infraestructura de ordenador que realiza uno o más aspectos de la presente invención para uno o más clientes. A cambio, el proveedor de servicios puede recibir un pago del cliente bajo una suscripción y/o acuerdo de tarifa, como ejemplos. Adicional o alternativamente, el proveedor de servicios puede recibir un pago de la venta de contenido de publicidad a una o más terceras partes.

En un aspecto de la presente invención, se puede desplegar una aplicación para realizar uno o más aspectos de la presente invención. Como ejemplo, el despliegue de una aplicación comprende proporcionar infraestructura de ordenador operable para realizar uno o más aspectos de la presente invención.

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

Aún como un aspecto adicional de la presente invención, se puede proporcionar un proceso para integrar infraestructura informática que comprende integrar código legible por ordenador en un sistema de ordenador. El sistema de ordenador comprende un medio legible por ordenador, en el que el medio de ordenador comprende uno o más aspectos de la presente invención. El código en combinación con el sistema de ordenador es capaz de realizar uno o más aspectos de la presente invención.

Aunque se describieron anteriormente varias realizaciones, estas son solamente ejemplos. Por ejemplo, entornos informáticos de otras arquitecturas pueden incorporar y usar uno o más aspectos de la presente invención. Además, se pueden usar vectores de otros tamaños y se pueden hacer cambios a las instrucciones sin apartarse del espíritu

de la presente invención. Adicionalmente, se pueden usar registros distintos de los registros de vector y/o los datos pueden ser distintos de datos de caracteres, tales como datos enteros u otros tipos de datos.

Además, otros tipos de entornos informáticos pueden beneficiarse de uno o más aspectos de la presente invención. Como ejemplo, es utilizable un sistema de procesamiento de datos adecuado para almacenar y/o ejecutar código de programa que incluye al menos dos procesadores acoplados directa o indirectamente a elementos de memoria a través del canal principal del sistema. Los elementos de memoria incluyen, por ejemplo, memoria local empleada durante la ejecución real del código de programa, almacenamiento en masa y memoria caché que proporcionan almacenamiento temporal de al menos algún código de programa a fin de reducir el número de veces que el código debe ser recuperado de almacenamiento en masa durante la ejecución.

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

Con referencia a la FIG. 10, se representan componentes representativos de un sistema de Ordenador Central 5000 para implementar uno o más aspectos de la presente invención. El ordenador central representativo 5000 comprende una o más CPU 5001 en comunicación con memoria de ordenador (es decir, almacenamiento central) 5002, así como interfaces de I/O para almacenar dispositivos de medios 5011 y redes 5010 para comunicar con otros ordenadores o SAN y similares. La CPU 5001 es compatible con una arquitectura que tiene un conjunto de instrucciones de arquitectura y funcionalidad de arquitectura. La CPU 5001 puede tener traducción de direcciones dinámica (DAT) 5003 para transformar direcciones de programa (direcciones virtuales) en direcciones reales de memoria. Una DAT típicamente incluye un almacenador temporal de traducción anticipada de instrucciones (TLB) 5007 para almacenar en caché traducciones de manera que los accesos posteriores al bloque de memoria de ordenador 5002 no requieren el retardo de la traducción de dirección. Típicamente, una caché 5009 se emplea entre la memoria de ordenador 5002 y el procesador 5001. La caché 5009 puede ser jerárquica que tiene una caché grande disponible para más de una CPU y cachés más pequeñas, más rápidas (menor nivel) entre la caché grande y cada CPU. En algunas implementaciones, las cachés de nivel inferior se dividen para proporcionar cachés de nivel bajo separadas para traer instrucciones y accesos de datos. En una realización, una instrucción se trae de la memoria 5002 por una unidad de búsqueda de instrucción 5004 a través de una caché 5009. La instrucción se decodifica en una unidad de decodificación de instrucción 5006 y despacha (con otras instrucciones en algunas realizaciones) a la unidad o unidades de ejecución de instrucción 5008. Típicamente se emplean varias unidades de ejecución 5008, por ejemplo una unidad de ejecución aritmética, una unidad de ejecución de punto flotante y una unidad de ejecución de instrucciones de ramal. La instrucción se ejecuta por la unidad de ejecución, que accede a operandos desde registros específicos de instrucción o memoria según se necesita. Si un operando va a ser accedido (cargado o almacenado) desde la memoria 5002, una unidad de carga/almacén 5005 típicamente maneja el acceso bajo control de la instrucción que se ejecuta. Las instrucciones se pueden ejecutar en circuitos hardware o en microcódigo interno (microprogramas) o por una combinación de ambos.

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

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

Las cachés separadas se pueden mantener para instrucciones y para operandos de datos. La información dentro de una caché se mantiene en octetos contiguos en un límite integral llamado un bloque de caché o línea de caché (o línea, para abreviar). Un modelo puede proporcionar una instrucción EXTRAER ATRIBUTO DE CACHÉ que devuelve el tamaño de una línea de caché en octetos. Un modelo también puede proporcionar instrucciones TRAER PREVIAMENTE DATOS y TRAER PREVIAMENTE DATOS RELATIVOS LARGOS que efectúa la búsqueda previa de almacenamiento en los datos o instrucción de caché o la liberación de datos desde la caché.

El almacenamiento se ve como una cadena de bits horizontal larga. Para la mayoría de operaciones, los accesos a almacenamiento pasan en una secuencia de izquierda a derecha. La cadena de bits se subdivide en unidades de ocho bits. Una unidad de ocho bits se llama un octeto, que es el bloque de construcción básico de todos los formatos de información. La localización de cada octeto en almacenamiento se identifica por un entero no negativo único, que

es la dirección de localización de ese octeto o, simplemente, la dirección del octeto. Las localizaciones de octeto adyacente tienen direcciones consecutivas, que comienzan con 0 en la izquierda y que proceden en una secuencia de izquierda a derecha. Las direcciones son enteros binarios sin signo y son de 24, 31 o 64 bits.

Se transmite información entre el almacenamiento y una CPU o un subsistema de canal un octeto o un grupo de octetos, a la vez. A menos que se especifique de otro modo, en, por ejemplo, la z/Architecture, un grupo de octetos en almacenamiento se direcciona por el octeto de más a la izquierda del grupo. El número de octetos en el grupo o bien está implícito o bien se especifica explícitamente por la operación a ser realizada. Cuando se usa en una operación de CPU, un grupo de octetos se llama campo. Dentro de cada grupo de octetos, en, por ejemplo, la z/Architecture, los bits se numeran en una secuencia de izquierda a derecha. En la z/Architecture, los bits de más a la izquierda se conocen algunas veces como los bits “de orden alto” y los bits de más a la derecha como los bits “de orden bajo”. Los números de bit no son direcciones de almacenamiento, no obstante. Solamente se pueden direccionar octetos. Para operar sobre bits individuales de un octeto en el almacenamiento, se accede al octeto entero. Los bits en un octeto se numeran de 0 hasta 7, de izquierda a derecha (en, por ejemplo, la z/Architecture). Los bits en una dirección se pueden numerar 8-31 o 40-63 para direcciones de 24 bits o 1-31 o 33-63 para direcciones de 31 bits; se numeran 0-63 para direcciones de 64 bits. Dentro de cualquier otro formato de longitud fija de múltiples octetos, los bits que componen el formato se numeran consecutivamente comenzando desde 0. Para propósitos de detección de errores y preferiblemente para corrección, se pueden transmitir uno o más bits de comprobación con cada octeto o con un grupo de octetos. Tales bits de comprobación se generan automáticamente por la máquina y no se pueden controlar directamente por el programa. Las capacidades de almacenamiento se expresan en número de octetos. Cuando la longitud de un campo de operando de almacenamiento está implícita por el código de operación de una instrucción, el campo se dice que tiene una longitud fija, que puede ser de uno, dos, cuatro, ocho o dieciséis octetos. Pueden estar implicados campos más grandes para algunas instrucciones. Cuando la longitud de un campo de operando de almacenamiento no está implícita sino que se expresa explícitamente, el campo se dice que tiene una longitud variable. Los operandos de longitud variable pueden variar de longitud en incrementos de un octeto (o con algunas instrucciones, en múltiplos de dos octetos u otros múltiplos). Cuando la información se coloca en el almacenamiento, se sustituyen los contenidos de solamente aquellas localizaciones de octetos que se incluyen en el campo designado, incluso aunque la anchura del camino físico al almacenamiento pueda ser mayor que la longitud del campo que se almacena.

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

En dispositivos que implementan cachés separadas para instrucciones y operandos de datos, se puede experimentar un retardo significativo si el programa almacena en una línea de caché desde la cual se traen posteriormente instrucciones, con independencia de si el almacén altera las instrucciones que se traen posteriormente.

En una realización, la invención se puede poner en práctica mediante software (algunas veces conocido como código interno con licencia, microprogramas, microcódigo, milicódigo, pico código y similares, cualquiera de los cuales sería coherente con uno o más aspectos de la presente invención). Con referencia a la FIG. 10, se puede acceder al código de programa software que incorpora uno o más aspectos de la presente invención mediante el procesador 5001 del sistema central 5000 desde dispositivos de medios de almacenamiento a largo plazo 5011, tales como una unidad de CD-ROM, unidad de cinta o disco duro. El código de programa software se puede incorporar en cualquiera de una variedad de medios conocidos para uso con un sistema de procesamiento de datos, tal como un disquete, disco duro o CD-ROM. El código se puede distribuir en tales medios o se puede distribuir a usuarios desde la memoria de ordenador 5002 o almacenamiento de un sistema de ordenador sobre una red 5010 a otros sistemas de ordenador para uso por usuarios de tales otros sistemas.

El código de programa software incluye un sistema operativo que controla la función e interacción de los diversos componentes de ordenador y uno o más programas de aplicaciones. El código de programa está paginado normalmente desde el dispositivo de medios de almacenamiento 5011 al almacenamiento de ordenador de velocidad relativamente más alta 5002 donde está disponible para procesamiento por el procesador 5001. Las técnicas y métodos para incorporar código de programa software en memoria, en medios físicos y/o código software de distribución a través de redes es bien conocido y no ser tratará más en la presente memoria. El código de programa, cuando se crea y almacena en un medio tangible (incluyendo pero no limitado a módulos de memoria electrónicos (RAM), la memoria rápida, Discos Compactos (CD), DVD, Cinta Magnética y similares se conoce a menudo como “producto de programa de ordenador”. El medio de producto de programa de ordenador es

típicamente legible por un circuito de procesamiento preferiblemente en un sistema de ordenador para ejecución por el circuito de procesamiento.

La FIG. 11 ilustra un sistema hardware de estación de trabajo o servidor representativo en el que se pueden poner en práctica uno o más aspectos de la presente invención. El sistema 5020 de la FIG. 11 comprende un sistema de ordenador base representativo 5021, tal como un ordenador personal, una estación de trabajo o un servidor, incluyendo dispositivos periféricos opcionales. El sistema de ordenador base 5021 incluye uno o más procesadores 5026 y un canal principal empleado para conectar y permitir la comunicación entre el(los) procesador(es) 5026 y los otros componentes del sistema 5021 según técnicas conocidas. El canal principal conecta el procesador 5026 a la memoria 5025 y el almacenamiento a largo plazo 5027 que puede incluir un disco duro (incluyendo cualquiera de medios magnéticos, CD, DVD y Memoria Rápida por ejemplo) o una unidad de cinta por ejemplo. El sistema 5021 también pudiera incluir un adaptador de interfaz de usuario, que conecta el microprocesador 5026 a través del canal principal a uno o más dispositivos de interfaz, tales como un teclado 5024, un ratón 5023, una impresora/escáner 5030 y/u otros dispositivos de interfaz, que pueden ser cualquier dispositivo de interfaz de usuario, tal como una pantalla sensible al tacto, almohadilla de entrada digitalizada, etc. El canal principal también conecta un dispositivo de visualización 5022, tal como una pantalla o monitor LCD, al microprocesador 5026 a través de un adaptador de visualización.

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

La FIG. 12 ilustra una red de procesamiento de datos 5040 en la que se pueden poner en práctica uno o más aspectos de la presente invención. La red de procesamiento de datos 5040 puede incluir una pluralidad de redes individuales, tales como una red inalámbrica y una red cableada, cada una de las cuales puede incluir una pluralidad de estaciones de trabajo individuales 5041, 5042, 5043, 5044. Adicionalmente, como apreciarán los expertos en la técnica, se pueden incluir una o más LAN, donde una LAN puede comprender una pluralidad de estaciones de trabajo inteligentes acopladas a un procesador central.

Aún con referencia a la FIG. 12, las redes también pueden incluir grandes ordenadores o servidores, tales como un ordenador pasarela (cliente servidor 5046) o servidor de aplicaciones (servidor remoto 5048 que puede acceder a un repositorio de datos y al que también se puede acceder directamente desde una estación de trabajo 5045). Un ordenador pasarela 5046 sirve como un punto de entrada en cada red individual. Se necesita una pasarela cuando se conecta un protocolo de interconexión con otro. La pasarela 5046 se puede acoplar preferiblemente a otra red (Internet 5047 por ejemplo) por medio de un enlace de comunicaciones. La pasarela 5046 también se puede acoplar directamente a una o más estaciones de trabajo 5041, 5042, 5043, 5044 usando un enlace de comunicaciones. El ordenador pasarela se puede implementar utilizando un servidor IBM eServer™ System z disponible en International Business Machines Corporation.

Con referencia concurrentemente a la FIG. 11 y la FIG. 12, se puede acceder a código de programación software que puede incorporar uno o más aspectos de la presente invención por el procesador 5026 del sistema 5020 desde medios de almacenamiento a largo plazo 5027, tales como una unidad de CD-ROM o disco duro. El código de programación software se puede incorporar en cualquiera de una variedad de medios conocidos para uso con un sistema de procesamiento de datos, tal como un disquete, disco duro o CD-ROM. El código se puede distribuir en tales medios o se puede distribuir a los usuarios 5050, 5051 desde la memoria o almacenamiento de un sistema de ordenador sobre una red a otros sistemas de ordenadores para uso por usuarios de tales otros sistemas.

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

La caché que está disponible más fácilmente al procesador (normalmente más rápida y más pequeña que otras cachés del procesador) es la caché más baja (L1 o nivel uno) y el almacenamiento principal (memoria principal) es la caché del nivel más alto (L3 si hay 3 niveles). La caché del nivel más bajo se divide a menudo en una caché de

instrucción (I-Caché) que mantiene instrucciones máquina a ser ejecutadas y una caché de datos (D-Caché) que mantiene operandos de datos.

Con referencia a la FIG. 13, se representa una realización de procesador ejemplar para el procesador 5026. Típicamente uno o más niveles de caché 5053 se emplean para almacenar temporalmente bloques de memoria a fin de mejorar el rendimiento del procesador. La caché 5053 es un almacenador temporal de alta velocidad que mantiene líneas de caché de datos de memoria que van a ser usadas probablemente. Las líneas de caché típicas son de 64, 128 o 256 octetos de datos de memoria. Se emplean a menudo cachés separadas para almacenar en caché instrucciones diferentes que para almacenamiento en caché de datos. La coherencia de caché (sincronización de copias de líneas en memoria y las cachés) a menudo se proporciona por varios algoritmos "de monitoreo" bien conocidos en la técnica. El almacenamiento de memoria principal 5025 de un sistema de procesador a menudo se conoce como caché. En un sistema de procesador que tiene 4 niveles de caché 5053, el almacenamiento principal 5025 algunas veces se conoce como la caché de nivel 5 (L5) dado que es típicamente más rápida y solamente mantiene una parte del almacenamiento no volátil (DASD, cinta, etc.) que está disponible para un sistema informático. El almacenamiento principal 5025 "almacena en caché" páginas de datos paginados dentro y fuera del almacenamiento principal 5025 por el sistema operativo.

Un contador de programa (contador de instrucción) 5061 hace el seguimiento de la dirección de la instrucción actual a ser ejecutada. Un contador de programa en un procesador de z/Architecture es de 64 bits y se puede truncar a 31 o 24 bits para soportar límites de direccionamiento anteriores. Un contador de programa se incorpora típicamente en una PSW (palabra de estado de programa) de un ordenador de manera que persiste durante la conmutación de contexto. De esta manera, un programa en curso, que tiene un valor de contador de programa, se puede interrumpir, por ejemplo, por el sistema operativo (conmutación de contexto desde el entorno de programa al entorno de sistema operativo). La PSW del programa mantiene el valor de contador de programa mientras que el programa no está activo y el contador de programa (en la PSW) del sistema operativo se usa mientras que se ejecuta el sistema operativo. Típicamente, el contador de programa se incrementa en una cantidad igual al número de octetos de la instrucción actual. Las instrucciones RISC (Ordenador de Juego de Instrucciones Reducido) son típicamente de longitud fija mientras que las instrucciones CISC (Ordenador de Juego de Instrucciones Complejo) son típicamente de longitud variable. Las instrucciones de la z/Architecture de IBM son instrucciones CISC que tienen una longitud de 2, 4 o 6 octetos. El contador de programa 5061 se modifica o bien por una operación de conmutación de contexto o bien por una operación tomada de ramal de una instrucción de ramal por ejemplo. En una operación de conmutación de contexto, el valor de contador de programa actual se ahorra en la palabra de estado de programa junto con otra información de estado acerca del programa que se ejecuta (tal como códigos de condición) y un nuevo valor de contador de programa se carga apuntando a una instrucción de un nuevo módulo de programa a ser ejecutado. Una operación tomada de ramal se realiza a fin de permitir al programa tomar decisiones o hacer bucle dentro del programa cargando el resultado de la instrucción de ramal en el contador de programa 5061.

Típicamente una unidad de búsqueda de instrucciones 5055 se emplea para traer instrucciones en nombre del procesador 5026. La unidad de búsqueda o bien trae las "siguientes instrucciones secuenciales", instrucciones objetivo de instrucciones tomadas de ramal o bien primeras instrucciones de un programa a continuación de una conmutación de contexto. Las unidades de búsqueda de instrucción modernas a menudo emplean técnicas de búsqueda previas para traer previamente especulativamente instrucciones basadas en la probabilidad de que las instrucciones buscadas previamente se pudieran usar. Por ejemplo, una unidad de búsqueda puede traer 16 octetos de instrucción que incluye la siguiente instrucción secuencial y octetos adicionales de instrucciones secuenciales adicionales.

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

Un procesador 5026 típicamente tiene una o más unidades 5057, 5058, 5060 para ejecutar la función de la instrucción. Con referencia a la FIG. 14A, una unidad de ejecución 5057 puede comunicar con registros generales de arquitectura 5059, una unidad de decodificación/despacho 5056, una unidad de carga almacén 5060 y otras unidades de procesador 5065 por medio de la lógica de interfaz 5071. Una unidad de ejecución 5057 puede emplear varios circuitos de registro 5067, 5068, 5069 para mantener información en la que operará la unidad de lógica aritmética (ALU) 5066. La ALU realiza operaciones aritméticas tales como sumar, restar, multiplicar y dividir así como funciones lógicas tales como and, or y or exclusiva (XOR), rotar y desplazar. Preferiblemente la ALU soporta operaciones especializadas que son dependientes del diseño. Otros circuitos pueden proporcionar otras facilidades de arquitectura 5072 incluyendo códigos de condición y lógica de soporte de recuperación por ejemplo. Típicamente el resultado de una operación de ALU se mantiene en un circuito de registro de salida 5070 que puede reenviar el

resultado a una variedad de otras funciones de procesamiento. Hay muchas disposiciones de unidades de procesador, la presente descripción se prevé solamente para proporcionar una comprensión representativa de una realización.

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

La unidad de ejecución 5057 realiza la suma aritmética en dos operandos y almacena el resultado en un tercer operando donde el tercer operando puede ser un tercer registro o uno de los dos registros fuente. La unidad de ejecución utiliza preferiblemente una Unidad de Lógica Aritmética (ALU) 5066 que es capaz de realizar una variedad de funciones lógicas tales como Desplazar, Rotar, And, Or y XOR así como una variedad de funciones algebraicas incluyendo cualquier de sumar, restar, multiplicar, dividir. Algunas ALU 5066 se diseñan para operaciones escalares y algunas para punto flotante. Los datos pueden ser Big Endian (donde el octeto menos significativo está en la dirección de octeto más alta) o Little Endian (donde el octeto menos significativo está en la dirección de octeto más baja) dependiendo de la arquitectura. La z/Architecture de IBM es Big Endian. Los campos de signo pueden ser signo y magnitud, complemento de 1 o complemento de 2 dependiendo de la arquitectura. Un número de complemento de 2 es ventajoso por que la ALU no necesita designar una capacidad de resta dado que o bien un valor negativo o bien un valor positivo en complemento de 2 requiere solamente una suma dentro de la ALU. Los números se describen comúnmente en abreviatura, donde un campo de 12 bits define una dirección de un bloque de 4.096 octetos y se describe comúnmente como un bloque de 4Kbyte (Kilo octetos), por ejemplo.

Con referencia a la FIG. 14B, una información de instrucción de ramal para ejecutar una instrucción de ramal se envía típicamente a una unidad de ramal 5058 que a menudo emplea un algoritmo de predicción de ramal tal como una tabla de historia de ramal 5082 para predecir el resultado del ramal antes de que se completen otras operaciones condicionales. El objetivo de la instrucción de ramal actual se traerá y ejecutará especulativamente antes de que se completen las operaciones condicionales. Cuando las operaciones condicionales se completan las instrucciones de ramal ejecutadas especulativamente o bien se completan o bien se descargan en base a las condiciones de la operación condicional y el resultado especulado. Una instrucción de ramal típica puede probar los códigos de condición y ramal a una dirección objetivo si los códigos de condición cumplen el requisito de ramal de la instrucción de ramal, se puede calcular una dirección objetivo en base a varios números incluyendo unos encontrados en campos de registro o un campo inmediato de la instrucción por ejemplo. La unidad de ramal 5058 puede emplear una ALU 5074 que tiene una pluralidad de circuitos de registro de entrada 5075, 5076, 5077 y un circuito de registro de salida 5080. La unidad de ramal 5058 puede comunicar con registros generales 5059, decodificar la unidad de despacho 5057 u otros circuitos 5073, por ejemplo.

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

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

Con referencia a la FIG. 14C, un procesador accede al almacenamiento usando una unidad de carga/almacén 5060. La unidad de carga/almacén 5060 puede realizar una operación de carga obteniendo la dirección del operando objetivo en memoria 5053 y cargando el operando en un registro 5059 u otra localización de memoria 5053 o puede

realizar una operación de almacén obteniendo la dirección del operando objetivo en la memoria 5053 y almacenando datos obtenidos desde un registro 5059 u otra localización de memoria 5053 en la localización de operando objetivo en la memoria 5053. La unidad de carga/almacén 5060 puede ser especulativa y puede acceder a memoria en una secuencia que está fuera de orden respecto a la secuencia de instrucciones, no obstante la unidad de carga/almacén 5060 tiene que mantener la apariencia a programas cuyas instrucciones fueron ejecutadas en orden. Una unidad de carga/almacén 5060 puede comunicar con registros generales 5059, unidad de decodificación/despacho 5056, interfaz de caché/memoria 5053 u otros elementos 5083 y comprende varios circuitos de registro, las ALU 5085 y la lógica de control 5090 para calcular direcciones de almacenamiento y proporcionar secuenciamiento de canalización para mantener las operaciones en orden. Algunas operaciones pueden estar fuera de orden pero la unidad de carga/almacén proporciona funcionalidad para hacer a las operaciones fuera de orden aparecer al programa como que se han realizado en orden, como es bien conocido en la técnica.

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

En el caso donde el procesador es un procesador de un sistema multiprocesador, cada procesador tiene responsabilidad de mantener recursos compartidos, tales como I/O, cachés, TLB y memoria, entrelazados por coherencia. Típicamente, las tecnologías “de monitoreo” se utilizarán en mantener la coherencia de caché. En un entorno de monitoreo, cada línea de caché se puede marcar como que está en cualquiera de un estado compartido, un estado exclusivo, un estado cambiado, un estado inválido y similares a fin de facilitar la compartición.

Las unidades de I/O 5054 (FIG. 13) dotan al procesador con medios para unirse a dispositivos periféricos incluyendo cinta, disco, impresoras, visualizadores y redes por ejemplo. Las unidades de I/O a menudo se presentan al programa de ordenador por controladores software. En grandes ordenadores, tales como el System z® de IBM®, los adaptadores de canal y adaptadores de sistemas abiertos son unidades de I/O del gran ordenador que proporciona las comunicaciones entre el sistema operativo y los dispositivos periféricos.

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

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

Más particularmente, en un gran ordenador, las instrucciones máquina de arquitectura se usan por los programadores, normalmente hoy en día programadores de “C”, a menudo por medio de una aplicación de compilador. Estas instrucciones almacenadas en el medio de almacenamiento se pueden ejecutar nativamente en un Servidor de z/Architecture de IBM® o alternativamente en máquinas que ejecutan otras arquitecturas. Se pueden emular en los servidores de grandes ordenadores de IBM® existentes y futuros y en otras máquinas de IBM® (por

ejemplo, servidores Power Systems y Servidores System x®). Se pueden ejecutar en máquinas que ejecutan Linux de una variedad de máquinas que usan hardware fabricado por IBM®, Intel®, AMD™ y otros. Además la ejecución en ese hardware bajo una z/Architecture, se puede usar también Linux como máquinas que usan emulación por Hercules, UMX o FSI (Fundamental Software, Inc), donde la ejecución generalmente es en un modo de emulación. En el modo de emulación, el software de emulación se ejecuta por un procesador nativo para emular la arquitectura de un procesador emulado.

El procesador nativo típicamente ejecuta emulación software que comprende o bien microprogramas o bien un sistema operativo nativo para realizar emulación del procesador emulado. El software de emulación es responsable de traer y ejecutar instrucciones de la arquitectura de procesador emulado. El software de emulación mantiene un contador de programa emulado para hacer el seguimiento de límites de instrucción. El software de emulación puede traer una o más instrucciones máquina emuladas a la vez y convertir la una o más instrucciones máquina emuladas a un grupo correspondiente de instrucciones máquina nativas para ejecución por el procesador nativo. Estas instrucciones convertidas se pueden almacenar en caché de manera que se puede consumar una conversión más rápida. No obstante, el software de emulación tiene que mantener las reglas de arquitectura de la arquitectura de procesador emulada para asegurar sistemas operativos y aplicaciones escritas para la operación correcta del procesador emulado. Además, el software de emulación tiene que proporcionar recursos identificados por la arquitectura de procesador emulado incluyendo, pero no limitado a, registros de control, registros de propósito general, registros de punto flotante, función de traducción de dirección dinámica incluyendo tablas de segmentos y tablas de páginas por ejemplo, mecanismos de interrupción, mecanismos de conmutación de contexto, relojes de Hora del Día (TOD) e interfaces de arquitectura a subsistemas I/O de manera que un sistema operativo o un programa de aplicaciones diseñado para ejecutarse en el procesador emulado, se puede ejecutar en el procesador nativo que tiene el software de emulación.

Una instrucción específica que se emula se codifica y se llama a una subrutina para realizar la función de la instrucción individual. Una función de software de emulación que emula una función de un procesador emulado se implementa, por ejemplo, en una subrutina o controlador “C” o algún otro método de proporcionar un controlador para el hardware específico que estará dentro de la habilidad de los expertos en la técnica después de comprender la descripción de la realización preferida. Diversas patentes de emulación software y hardware incluyendo, pero no limitadas a la Concesión de Patente de EE.UU. Nº 5.551.013, titulada “Multiprocessor for Hardware Emulation”, de Beausoleil et al.; y la Concesión de Patente de EE.UU. Nº 6.009.261, titulada “Preprocessing of Stored Target Routines for Emulating Incompatible Instructions on a Target Processor”, de Scalzi et al.; y la Concesión de Patente Nº 5.574.873, titulada “Decoding Guest Instruction to Directly Access Emulation Routines that Emulate the Guest Instructions”, de Davidian et al.; y la Concesión de Patente de EE.UU. Nº 6.308.255, titulada “Symmetrical Multiprocessing Bus and Chipset Used for Coprocessor Support Allowing Non-Native Code to Run in a System”, de Gorishek et al.; y la Concesión de Patente de EE.UU. Nº 6.463.582, titulada “Dynamic Optimizing Object Code Translator for Architecture Emulation and Dynamic Optimizing Object Code Translation Method”, de Lethin et al.; y la Concesión de Patente de EE.UU. Nº 5.790.825, titulada “Method for Emulating Guest Instructions on a Host Computer Through Dynamic Recompilation of Host Instructions”, de Eric Traut; y muchas otras, ilustran una variedad de formas conocidas para lograr emulación de un formato de instrucción de arquitectura para una máquina diferente para una máquina objetivo disponible para los expertos en la técnica.

En la FIG. 15, se proporciona un ejemplo de un sistema de ordenador central emulado 5092 que emula un sistema de ordenador central 5000’ de una arquitectura central. En el sistema de ordenador central emulado 5092, el procesador central (CPU) 5091 es un procesador central emulado (o procesador central virtual) y comprende un procesador de emulación 5093 que tiene una arquitectura de conjunto de instrucciones nativas diferente de la del procesador 5091 del ordenador central 5000’. El sistema de ordenador central emulado 5092 tiene una memoria 5094 accesible al procesador emulado 5093. En la realización ejemplo, la memoria 5094 se divide en una parte de memoria de ordenador central 5096 y una parte de rutinas de emulación 5097. La memoria de ordenador central 5096 está disponible para programas del ordenador central emulado 5092 según la arquitectura de ordenador central. El procesador de emulación 5093 ejecuta instrucciones nativas de un conjunto de instrucciones de arquitectura de una arquitectura distinta de la del procesador emulado 5091, las instrucciones nativas obtenidas de la memoria de rutinas de emulación 5097 y puede acceder a una instrucción central para ejecución desde un programa en la memoria de ordenador central 5096 empleando una o más instrucciones obtenidas en una secuencia y rutina de acceso/decodificación que puede decodificar la(s) instrucción(instrucciones) accedida(s) para determinar una rutina de ejecución de instrucciones nativas para emular la función de la instrucción central accedida. Otras facilidades que se definen para la arquitectura del sistema de ordenador central 5000’ se pueden emular por rutinas de facilidades de arquitectura, incluyendo tales facilidades como registros de propósito general, registros de control, traducción de dirección dinámica y caché de procesador y soporte de subsistema de I/O, por ejemplo. Las rutinas de emulación también pueden obtener ventaja de funciones disponibles en el procesador de emulación 5093 (tales como registros generales y traducción dinámica de direcciones virtuales) para mejorar el rendimiento de las rutinas de emulación. También se pueden proporcionar hardware especial y motores fuera de carga para ayudar al procesador 5093 en la emulación de la función del ordenador central 5000’.

La terminología usada en la presente memoria es para el propósito de describir realizaciones particulares solamente y no se pretende que sea limitante de la invención. Como se usa en la presente memoria, las formas singulares “un”, “uno” y “el” se pretende que incluyan las formas plurales también, a menos que el contexto lo indique claramente de

otro modo. Además se entenderá que los términos “comprende” y/o “que comprende”, cuando se usan en esta especificación, especifican la presencia de rasgos, enteros, pasos, operaciones, elementos y/o componentes indicados, pero no excluyen la presencia o adición de uno o más de otros rasgos, enteros, pasos, operaciones, elementos, componentes y/o grupos de los mismos.

Las estructuras, materiales, actos y equivalentes correspondientes de todos los medios o pasos más elementos de función en las reivindicaciones de más adelante, en su caso, se pretende que incluyan cualquier estructura, material o acto para realizar la función en combinación con otros elementos reivindicados como se reivindica específicamente. La descripción de uno o más aspectos de la presente invención se ha presentado con propósitos de ilustración y descripción, pero no se pretende que sean exhaustivos o estén limitados a la invención en la forma descrita. Muchas modificaciones y variaciones serán evidentes para los expertos en la técnica sin apartarse del alcance y espíritu de la invención. La realización se eligió y describió a fin de explicar mejor los principios de la invención y la aplicación práctica y para permitir a otros expertos en la técnica entender la invención para diversas realizaciones con diversas modificaciones que son adecuadas al uso particular contemplado.

REIVINDICACIONES

1. Un método para ejecutar una instrucción máquina en una unidad central de proceso que comprende los pasos de:

obtener, mediante un procesador, una instrucción máquina (310) para ejecución, la instrucción máquina que se define para ejecución de ordenador según una arquitectura de ordenador, la instrucción máquina que comprende:

al menos un campo de código de operación (302a, 302b) para proporcionar un código de operación, el código de operación que identifica una operación Encontrar Elemento Igual de Vector;

un campo de extensión (314) a ser usado en la designación de uno o más registros;

un primer campo de registro (304) combinado con una primera parte del campo de extensión (314) para designar un primer registro, el primer registro que comprende un primer operando, el primer operando que es un operando de vector que comprende un número predefinido de elementos;

un segundo campo de registro (306) combinado con una segunda parte del campo de extensión (314) para designar un segundo registro, el segundo registro que comprende un segundo operando, el segundo operando que es un operando de vector que comprende un número predefinido de elementos;

un tercer campo de registro (308) combinado con una tercera parte del campo de extensión (314) para designar un tercer registro, el tercer registro que comprende un tercer operando, el tercer operando que es un operando de vector que comprende un número predefinido de elementos;

un campo de máscara (310), el campo de máscara que comprende uno o más controles a ser usados durante la ejecución de la instrucción máquina; y

ejecutar la instrucción máquina, caracterizada por la ejecución que comprende:

determinar si el campo de máscara incluye un control de elemento cero fijado para indicar una búsqueda de un vector para un elemento cero (400);

basado en el campo de máscara que incluye el control de elemento cero fijado para indicar la búsqueda del vector para un elemento cero (402), buscar el segundo operando para un elemento cero (402), la búsqueda que proporciona un índice nulo (403), el índice nulo que incluye uno de un índice de un elemento cero encontrado en la búsqueda o un tamaño del vector si no se encuentra un elemento cero;

comparar cada elemento del segundo operando con cada elemento del tercer operando en paralelo (404), por igualdad, la comparación que proporciona un índice de comparación (410, 412), el índice de comparación que se basa en una dirección de la búsqueda (408) y un tamaño de los elementos del vector y que incluye uno de un índice de un elemento igual en base a la comparación que encuentra un elemento igual o un tamaño del vector en base a la comparación que encuentra elementos no iguales; y

proporcionar un resultado (416, 418), el resultado basado en si se realizó (414) la búsqueda para el elemento cero, en donde el resultado incluye uno de:

en base a no realizar la búsqueda del elemento cero (414), el resultado incluye el índice de comparación (416); o

en base a realizar la búsqueda del elemento cero (414), el resultado incluye un mínimo del índice de comparación o el índice nulo (418).

2. El método de la reivindicación 1, en donde el método además comprende:

ajustar el resultado, el ajuste que comprende realizar al menos una operación en el resultado para proporcionar un resultado ajustado, el resultado ajustado que comprende un índice de octeto de un primer octeto de un elemento cero o un elemento igual en base a la búsqueda y comparación; y

almacenar el resultado en el primer operando.

3. El método de la reivindicación 2, en donde la instrucción máquina además comprende otro campo de máscara, el otro campo de máscara que incluye un control de tamaño de elemento, el control de tamaño de elemento que especifica un tamaño de elementos en al menos uno del primer operando, el segundo operando o el tercer operando y en donde el tamaño se usa en el ajuste.

4. El método de la reivindicación 1, en donde el campo de máscara comprende un control de conjunto de códigos de condición y en donde el método comprende:

determinar si el control de conjunto de códigos de condición está fijado; y

en base al control de conjunto de códigos de condición que está fijado, fijar un código de condición en base a la ejecución de la instrucción máquina.

5. El método de la reivindicación 1, en donde la ejecución comprende determinar, en el tiempo de ejecución, una dirección para la comparación, en donde la dirección es una de izquierda a derecha o derecha a izquierda y la determinación comprende acceder por la instrucción máquina a un control de dirección para determinar la dirección.

6. El método de la reivindicación 1, en donde el segundo operando y el tercer operando comprenden N octetos y en donde la comparación comprende comparar en paralelo los N octetos del segundo operando con los N octetos del tercer operando y en donde un tamaño de un elemento comprende uno de un octeto, dos octetos o cuatro octetos.

7. El método de la reivindicación 1, en donde el índice del elemento cero comprende un índice de octeto, el índice de octeto que es un índice de un primer octeto del elemento cero.

8. El método de la reivindicación 1, en donde el índice del elemento desigual comprende un índice de octeto, el índice de octeto que es un índice de un octeto del elemento igual.

9. El método de la reivindicación 1, en donde al menos uno de la indicación de elementos no cero encontrados o la indicación de no igualdad comprende incluir en al menos uno del índice nulo o el índice de comparación un número que representa un tamaño del segundo operando.

10. Un producto de programa de ordenador que comprende un medio de almacenamiento legible por ordenador que tiene un código de programa legible por ordenador incorporado en el mismo,

en donde el código de programa legible por ordenador se ejecuta en un sistema de ordenador para realizar todos los pasos de cualquiera de las reivindicaciones 1-9 del método.

11. Un sistema de ordenador cargado con un producto de programa de ordenador según la reivindicación 10; que comprende uno o más procesadores adaptados para realizar todos los pasos de cualquiera de las reivindicaciones 1-9 del método, mediante la ejecución del código de programa legible por ordenador de dicho producto de programa de ordenador.

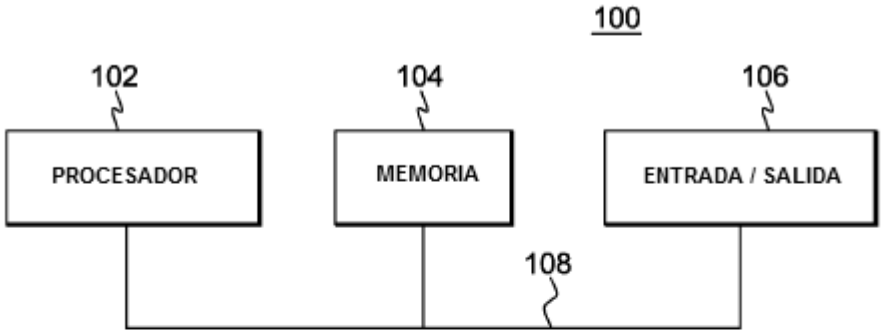

FIG. 1

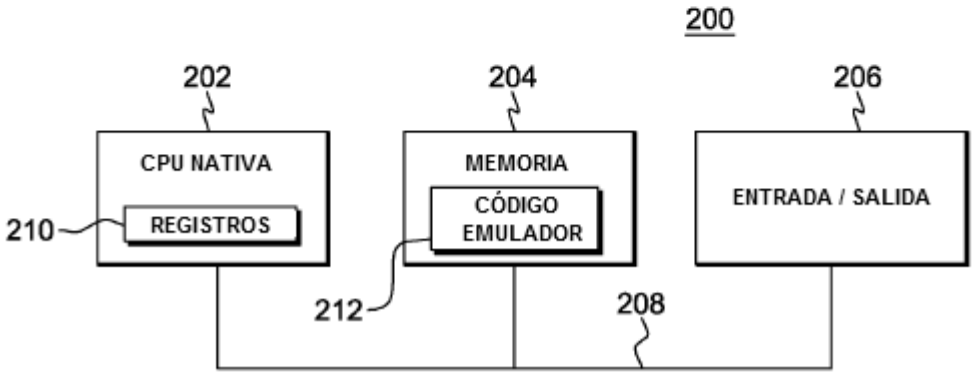

FIG. 2A

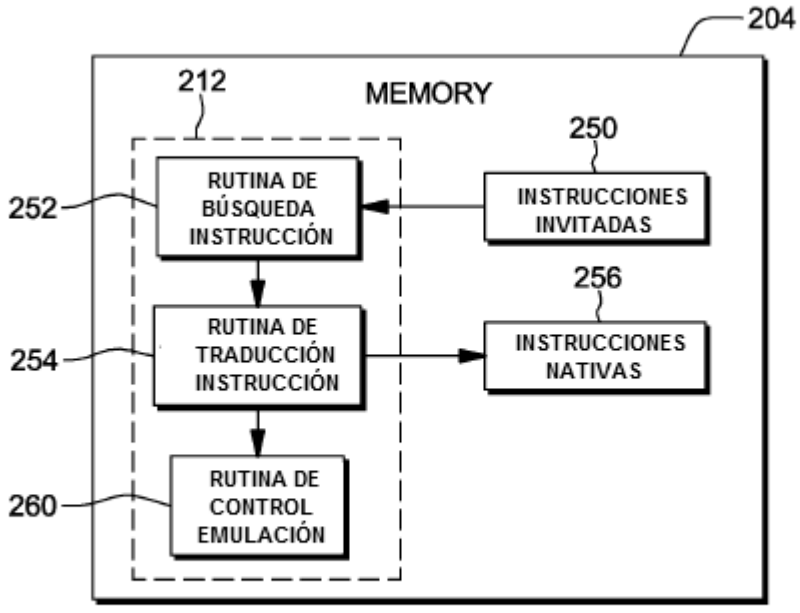

FIG. 2B

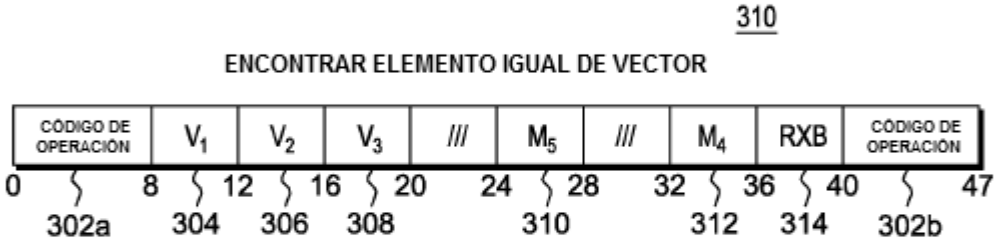
310
ENCONTRAR ELEMENTO IGUAL DE VECTOR
CÓDIGO DE OPERACIÓN
V1
V2
V3
///
M5
///
M4
RXB
CÓDIGO DE OPERACIÓN
0
8
12
16
20
24
28
32
36
40
47
302a
304
306
308
310
312
314
302b

FIG. 3

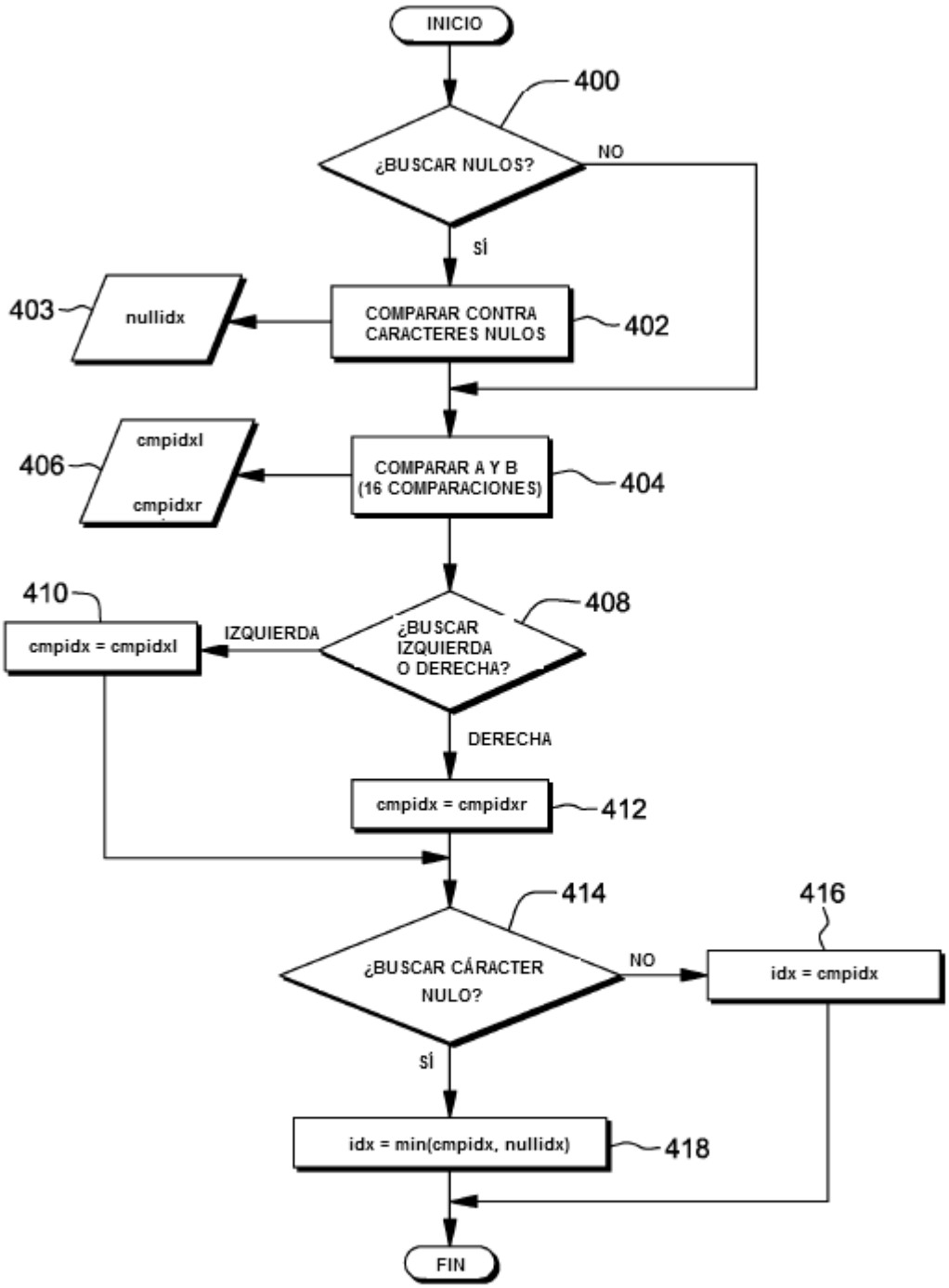
INICIO
400
¿BUSCAR NULOS?
NO
SÍ
403
nullidx
COMPARAR CONTRA CARACTERES NULOS
402
406
cmpidxl
cmpidxr
COMPARAR A Y B (16 COMPARACIONES)
404
410
cmpidx = cmpidxl
IZQUIERDA
¿BUSCAR IZQUIERDA O DERECHA?
408
DERECHA
cmpidx = cmpidxr
412
414
¿BUSCAR CÁRACTER NULO?
NO
416
idx = cmpidx
SÍ
idx = min(cmpidx, nullidx)
418
FIN

FIG. 4

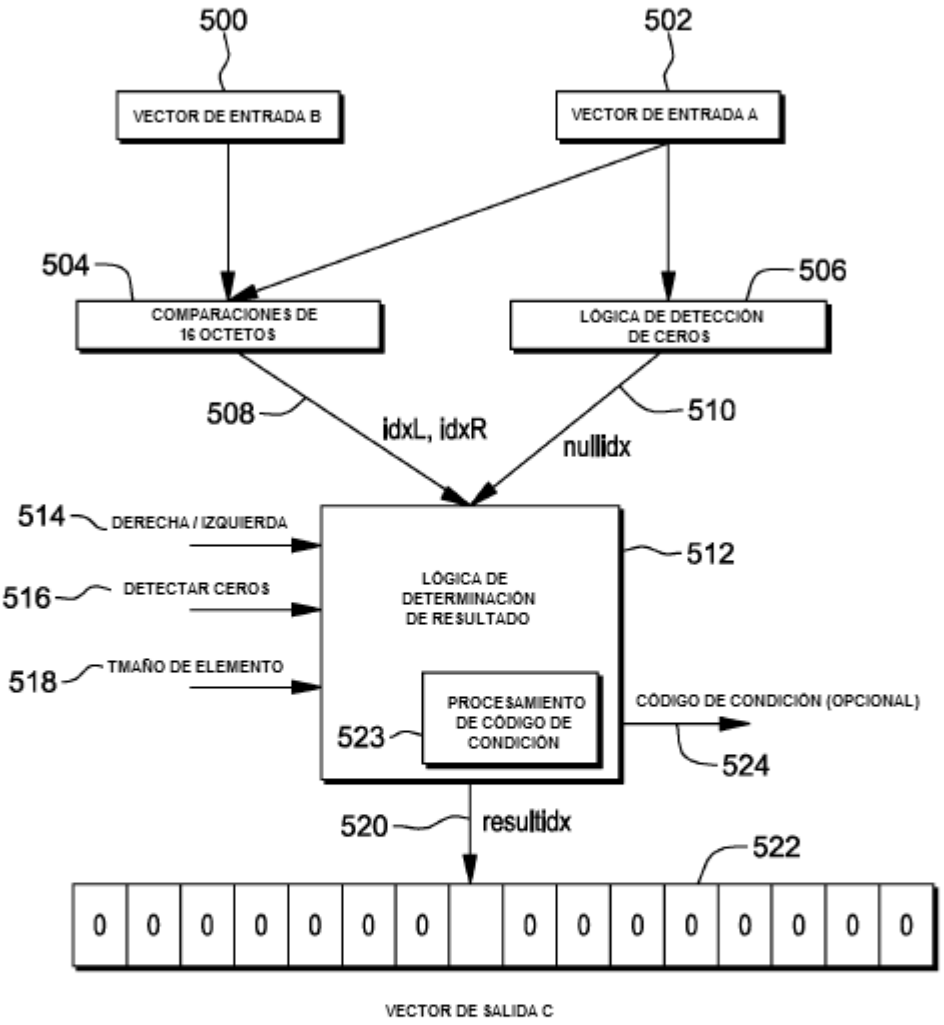
500
502
VECTOR DE ENTRADA B
VECTOR DE ENTRADA A
504
506
COMPARACIONES DE 16 OCTETOS
LÓGICA DE DETECCIÓN DE CEROS
508
idxL, idxR
510
nullidx
514
DERECHA / IZQUIERDA
512
516
DETECTAR CEROS
LÓGICA DE DETERMINACIÓN DE RESULTADO
518
TMAÑO DE ELEMENTO
PROCESAMIENTO DE CÓDIGO DE CONDICIÓN
CÓDIGO DE CONDICIÓN (OPCIONAL)
523
524
520
resultidx
522
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
VECTOR DE SALIDA C

FIG. 5

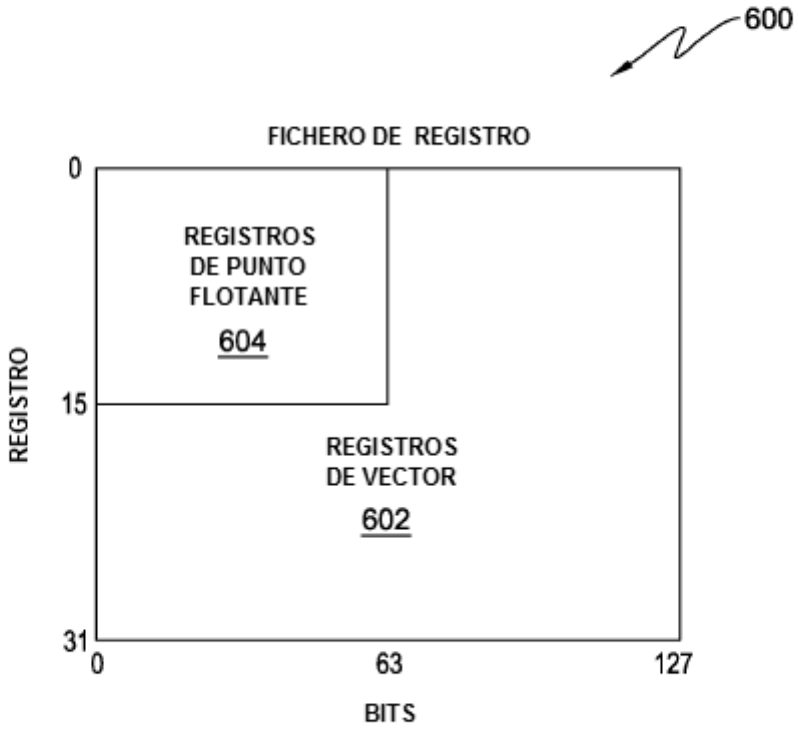
600
FICHERO DE REGISTRO
0
REGISTROS
DE PUNTO
FLOTANTE
604
15
REGISTRO
REGISTROS
DE VECTOR
602
31
0
63
127
BITS

FIG. 6

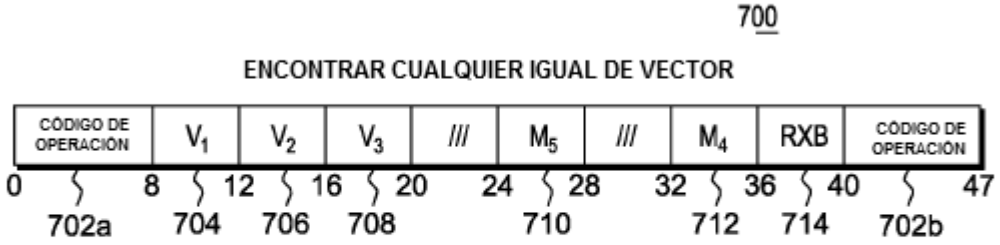
700
ENCONTRAR CUALQUIER IGUAL DE VECTOR
CÓDIGO DE OPERACIÓN
V1
V2
V3
///
M5
///
M4
RXB
CÓDIGO DE OPERACIÓN
0
8
12
16
20
24
28
32
36
40
47
702a
704
706
708
710
712
714
702b

FIG. 7

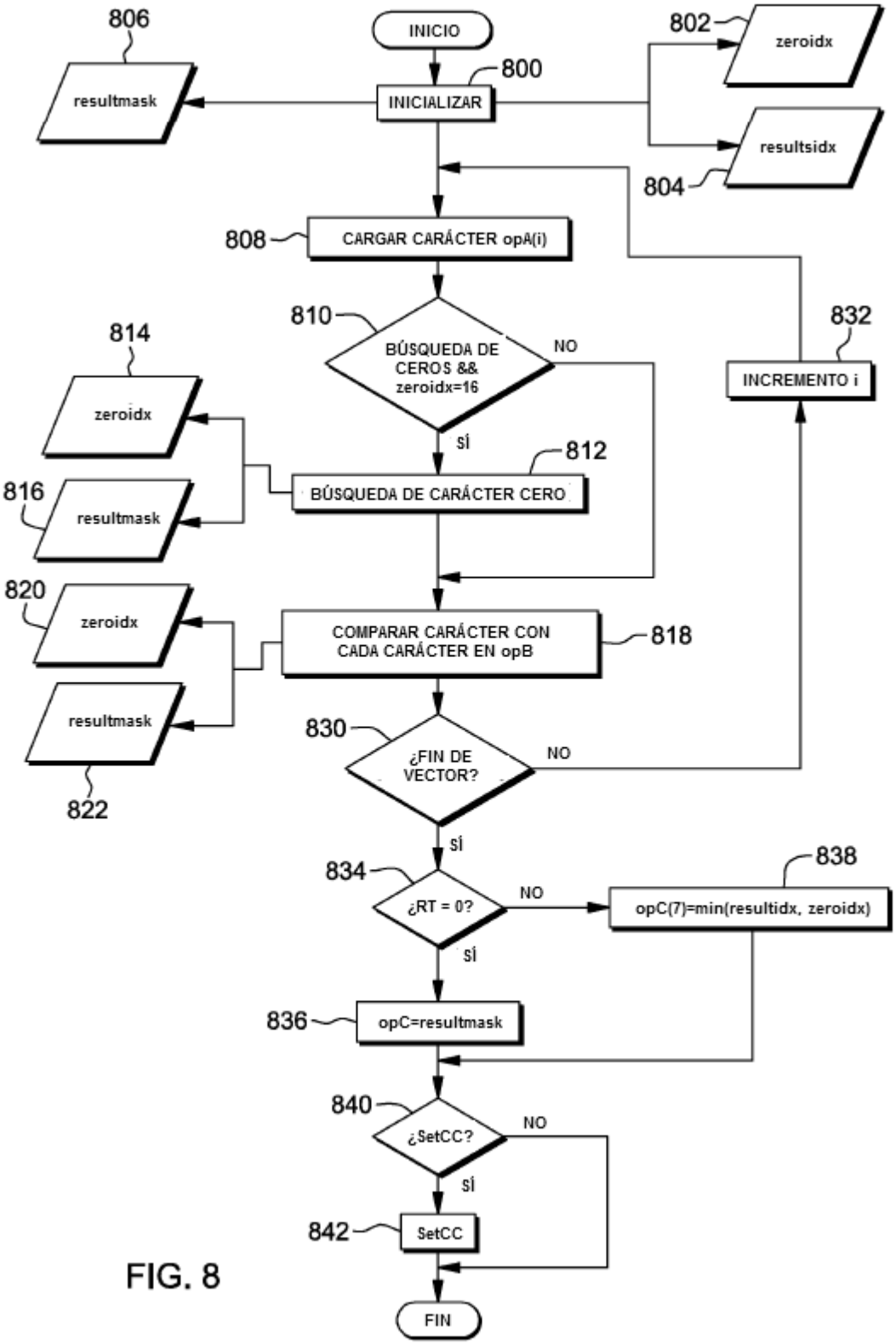
INICIO
800
INICIALIZAR
802
zeroidx
804
resultsidx
806
resultmask
808
CARGAR CARÁCTER opA(i)
810
BÚSQUEDA DE CEROS && zeroidx=16
NO
SÍ
812
BÚSQUEDA DE CARÁCTER CERO
814
zeroidx
816
resultmask
818
COMPARAR CARÁCTER CON CADA CARÁCTER EN opB
820
zeroidx
822
resultmask
830
¿FIN DE VECTOR?
NO
SÍ
832
INCREMENTO i
834
¿RT = 0?
NO
SÍ
838
opC(7)=min(resultidx, zeroidx)
836
opC=resultmask
840
¿SetCC?
NO
SÍ
842
SetCC
FIN

FIG. 8


PRODUCTO DE PROGRAMA DE ORDENADOR
900
904
LÓGICA DE CÓDIGO DE PROGRAMA
MEDIO DE ALMACENAMIENTO LEGIBLE POR ORDENADOR
902

FIG. 9

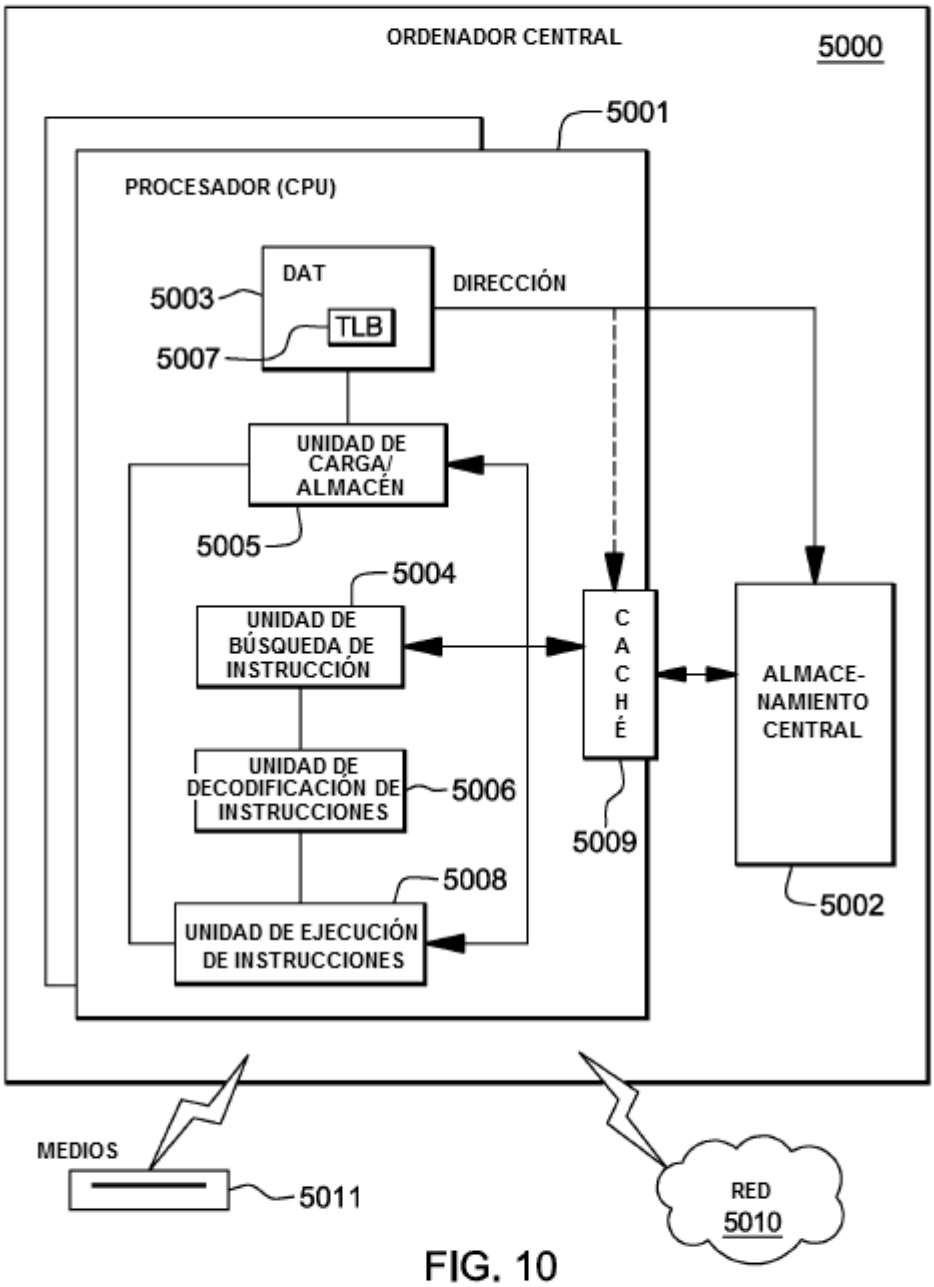
ORDENADOR CENTRAL
5000
5001
PROCESADOR (CPU)
DAT
5003
TLB
5007
DIRECCIÓN
UNIDAD DE CARGA/ ALMACÉN
5005
5004
UNIDAD DE BÚSQUEDA DE INSTRUCCIÓN
UNIDAD DE DECODIFICACIÓN DE INSTRUCCIONES
5006
5008
UNIDAD DE EJECUCIÓN DE INSTRUCCIONES
CACHÉ
5009
ALMACE-NAMIENTO CENTRAL
5002
MEDIOS
5011
RED
5010

FIG. 10

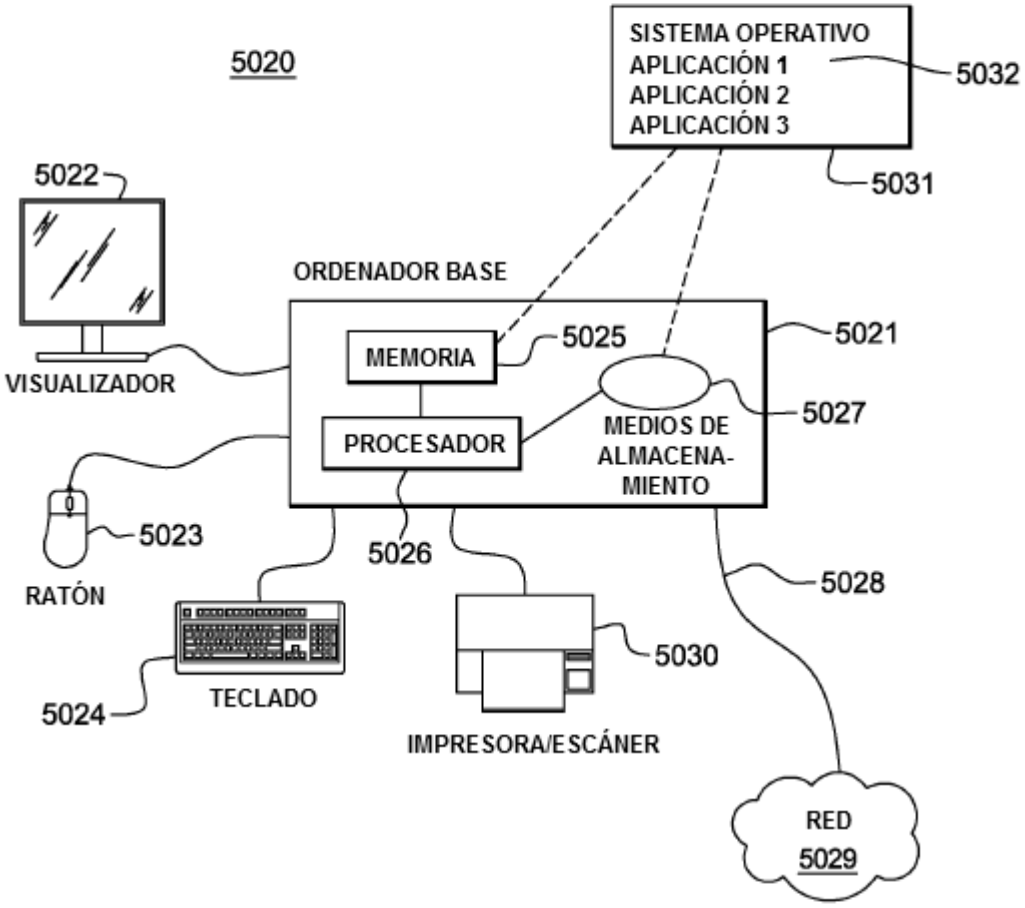
5020
SISTEMA OPERATIVO
APLICACIÓN 1
APLICACIÓN 2
APLICACIÓN 3
5032
5031
5022
ORDENADOR BASE
MEMORIA
5025
5021
PROCESADOR
MEDIOS DE ALMACENA-MIENTO
5027
VISUALIZADOR
5023
RATÓN
5026
5028
5024
TECLADO
5030
IMPRESORA/ESCÁNER
RED
5029

FIG. 11

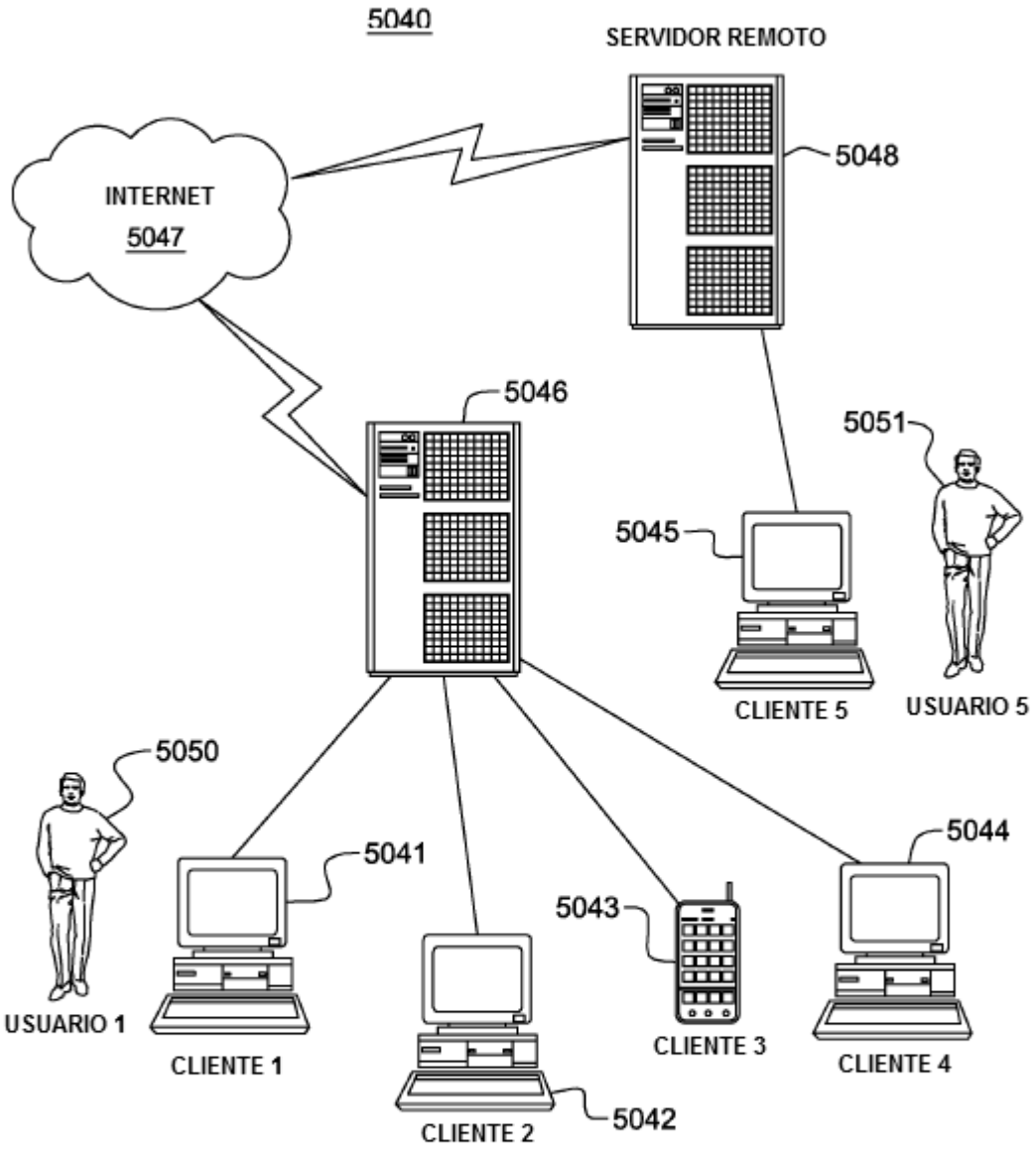
5040
SERVIDOR REMOTO
5048
INTERNET
5047
5046
5051
5045
CLIENTE 5
USUARIO 5
5050
5041
5044
5043
USUARIO 1
CLIENTE 1
CLIENTE 3
CLIENTE 4
CLIENTE 2
5042

FIG. 12

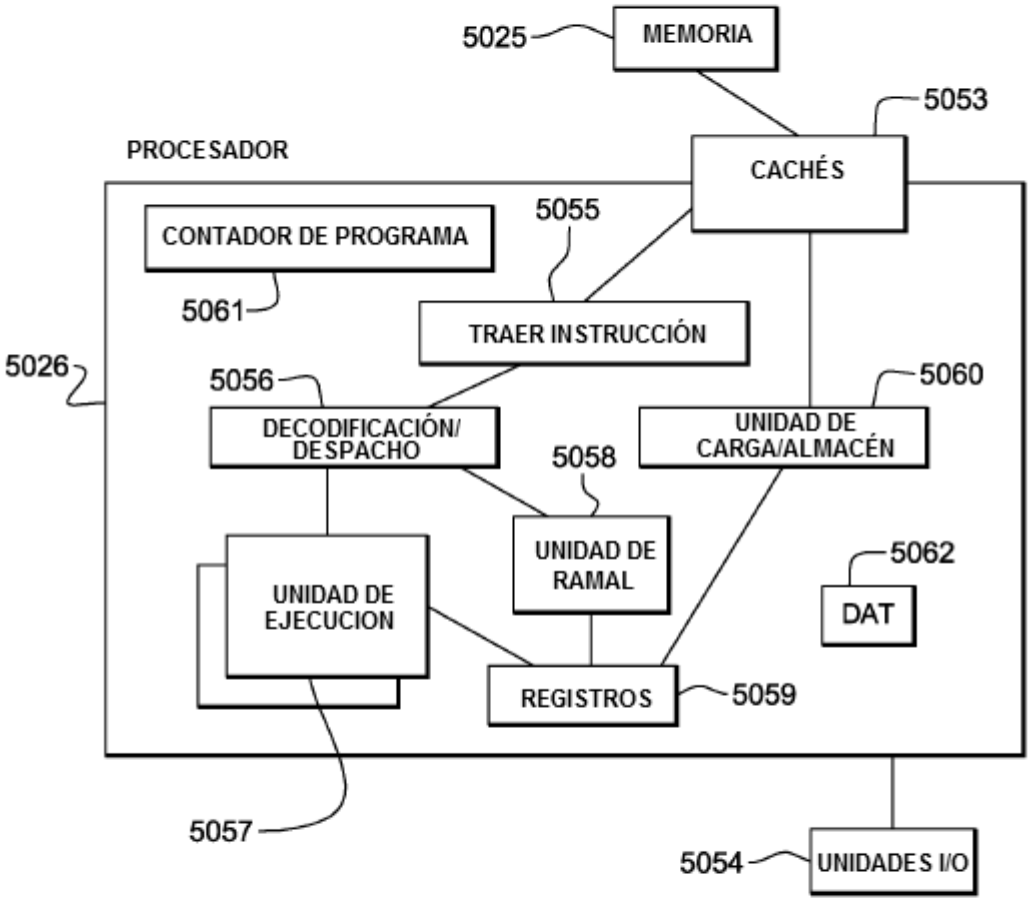
5025
MEMORIA
5053
PROCESADOR
CACHÉS
CONTADOR DE PROGRAMA
5055
5061
TRAER INSTRUCCIÓN
5026
5056
5060
DECODIFICACIÓN/
DESPACHO
UNIDAD DE
CARGA/ALMACÉN
5058
UNIDAD DE
RAMAL
5062
UNIDAD DE
EJECUCION
DAT
REGISTROS
5059
5057
5054
UNIDADES I/O

FIG. 13

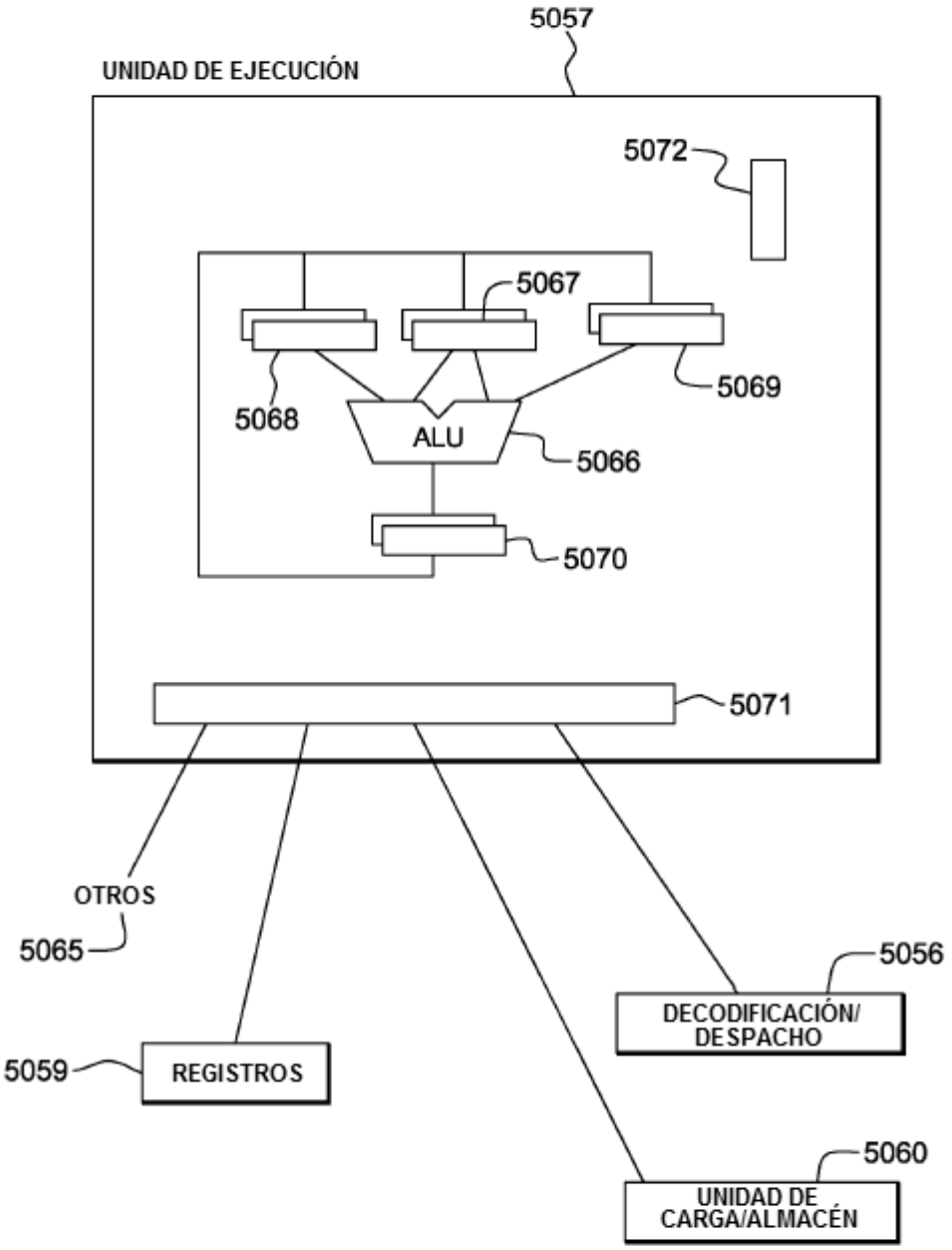
5057
UNIDAD DE EJECUCIÓN
5072
5067
5068
5069
ALU
5066
5070
5071
OTROS
5065
5056
DECODIFICACIÓN/
DESPACHO
5059
REGISTROS
5060
UNIDAD DE
CARGA/ALMACÉN

FIG. 14A

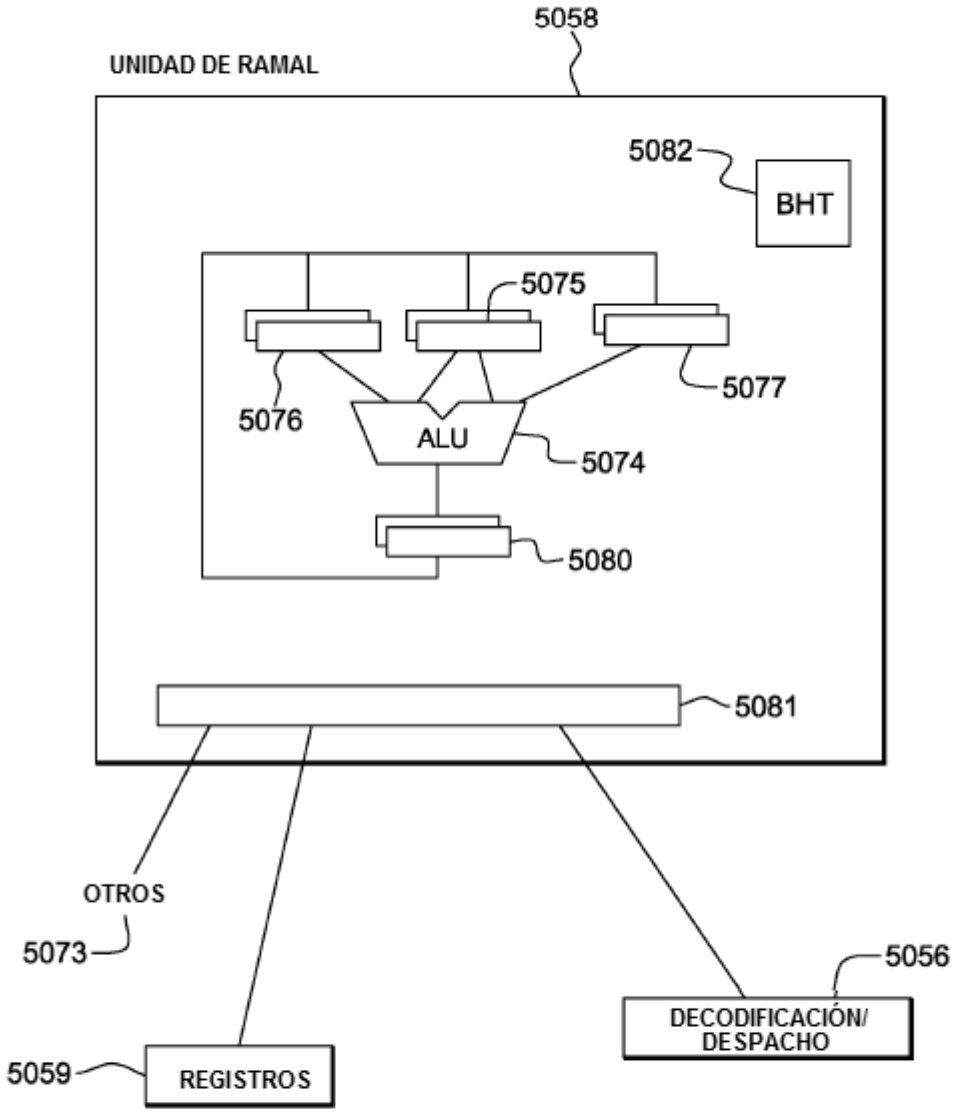
5058
UNIDAD DE RAMAL
5082
BHT
5075
5076
5077
ALU
5074
5080
5081
OTROS
5073
5056
DECODIFICACIÓN/
DESPACHO
5059
REGISTROS

FIG. 14B

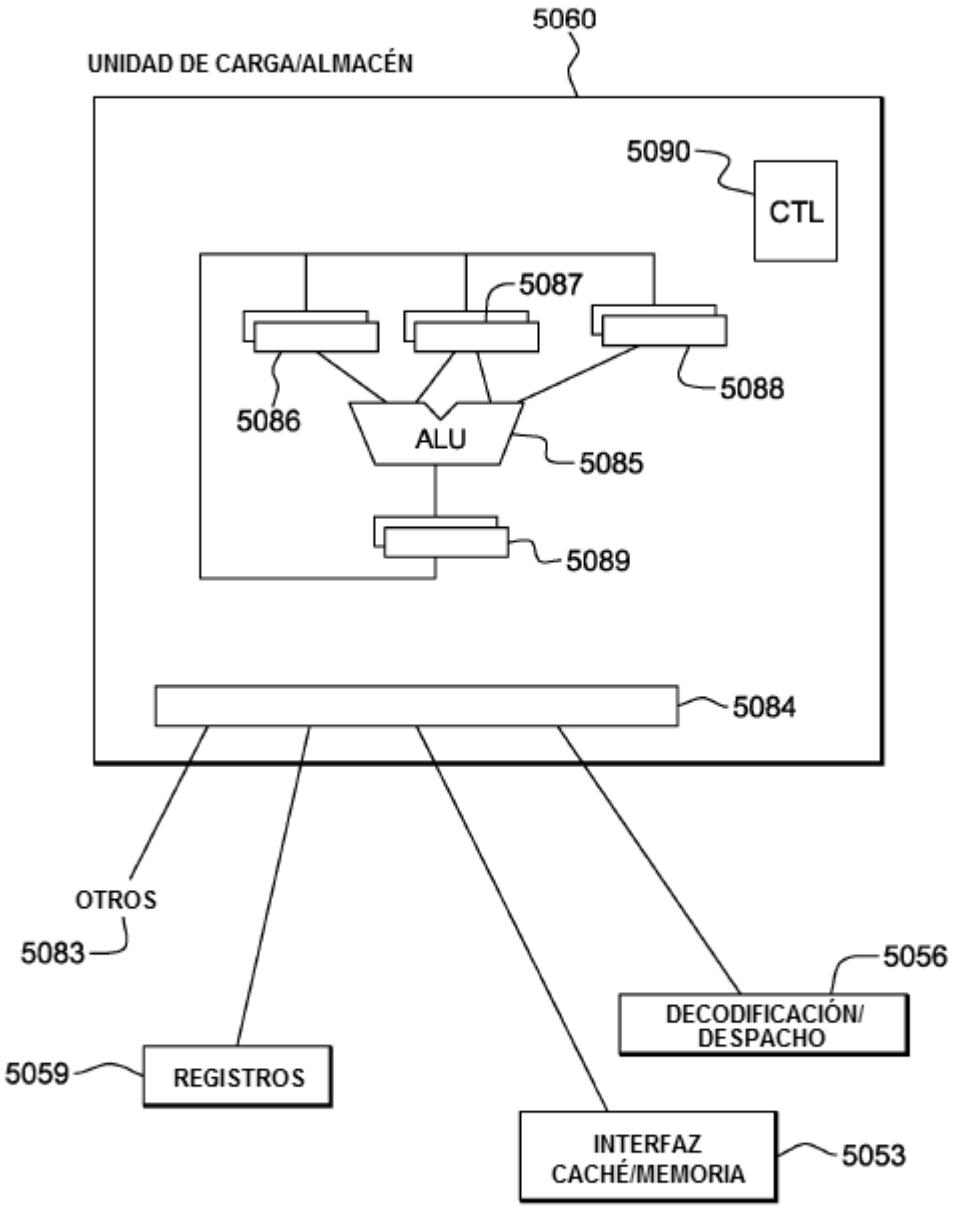
5060
UNIDAD DE CARGA/ALMACÉN
5090
CTL
5087
5086
5088
ALU
5085
5089
5084
OTROS
5083
5056
DECODIFICACIÓN/
DESPACHO
5059
REGISTROS
INTERFAZ
CACHÉ/MEMORIA
5053

FIG. 14C

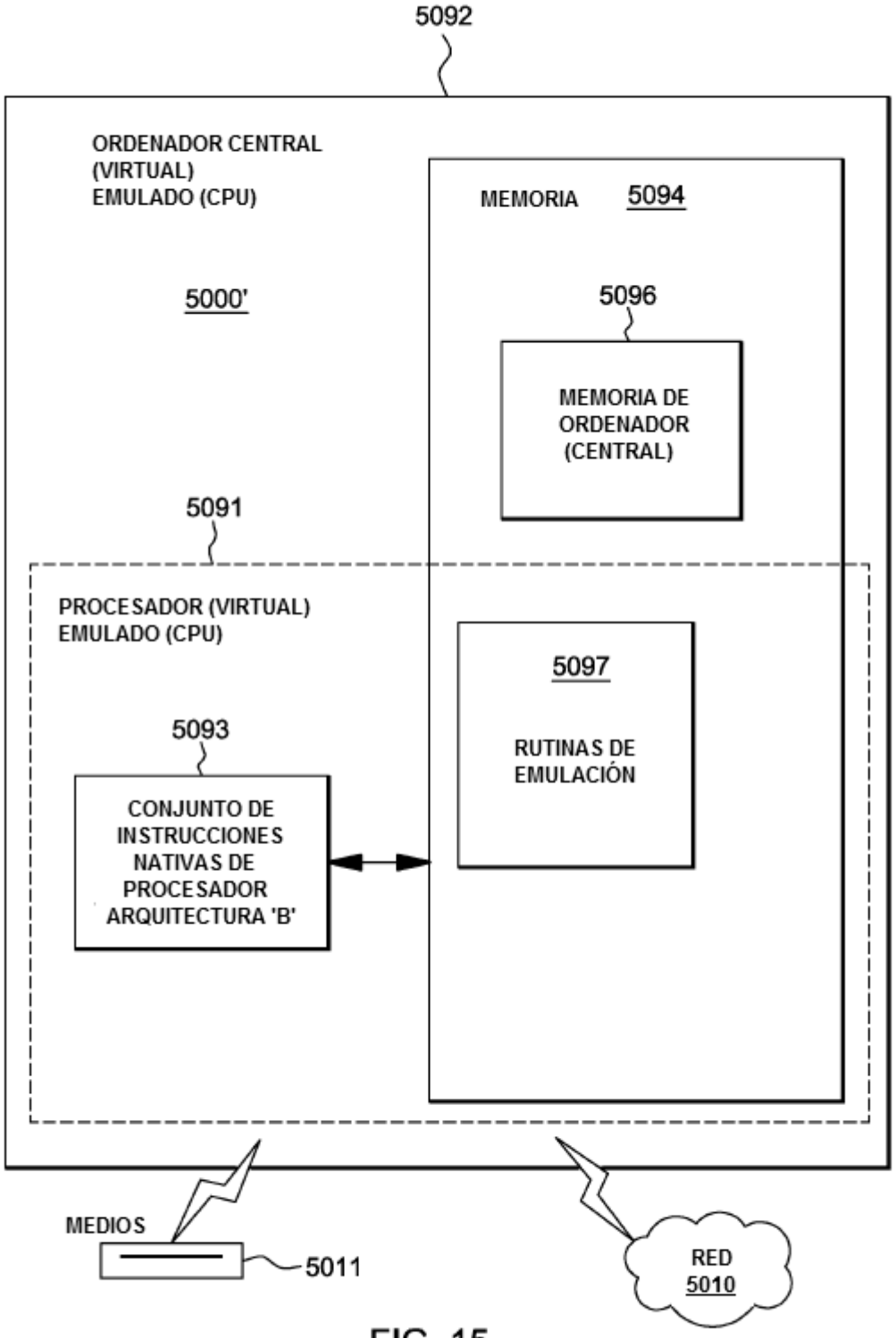
5092
ORDENADOR CENTRAL
(VIRTUAL)
EMULADO (CPU)
MEMORIA 5094
5000'
5096
MEMORIA DE
ORDENADOR
(CENTRAL)
5091
PROCESADOR (VIRTUAL)
EMULADO (CPU)
5097
RUTINAS DE
EMULACIÓN
5093
CONJUNTO DE
INSTRUCCIONES
NATIVAS DE
PROCESADOR
ARQUITECTURA 'B'
MEDIOS
5011
RED
5010

FIG. 15