(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 675 512**

(51) Int. CI.:

G06F 9/38 (2008.01)
G06F 9/30 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **15.11.2012 PCT/IB2012/056438**

(87) Fecha y número de publicación internacional: **19.09.2013 WO13136145**

(96) Fecha de presentación y número de la solicitud europea: **15.11.2012 E 12871181 (9)**

(97) Fecha y número de publicación de la concesión europea: **30.05.2018 EP 2769382**

(54) Título: **Instrucción para calcular la distancia a un límite de memoria específico**

(30) Prioridad:

15.03.2012 US 201213421451

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
11.07.2018

(73) Titular/es:

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

(72) Inventor/es:

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

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Instrucción para calcular la distancia a un límite de memoria específico

Antecedentes

La invención se refiere, en general, a procesamiento de datos, y en particular, a procesamiento que incluye límites de memoria.

El procesamiento de datos incluye varios tipos de procesamiento, incluyendo el procesamiento de textos u otros tipos de procesamiento. Durante el procesamiento de datos, se requiere con frecuencia realizar un tratamiento especial cuando los datos que van a ser procesados están próximos a un límite de memoria específico. Las instrucciones y/o técnicas actuales asociadas a procesamiento cerca de un límite de memoria tienden a ser ineficaces o problemáticas. La Patente de los Estados Unidos número US 6918010 presenta un método y un sistema para precarga de datos. La publicación de solicitud de Patente de los Estados Unidos número US 2008/077733 divulga un aparato de transferencia de datos que tiene un controlador configurado para lectura de datos en un área de dirección secuencial predeterminada en unidades de un primer conteo de bytes. La Patente de los Estados Unidos número US 7565510 se refiere a un microprocesador y a un método de control del mismo. La publicación de solicitud de Patente de los Estados Unidos número US 2007/106883 se refiere a la transmisión de un bloque de memoria con cualquier alineación de fuente en registros de propósito general (GPRs) a modo de datos alineados, usando una instrucción de carga de transmisión.

Sumario

Se abordan las deficiencias de la técnica anterior y se proporcionan ventajas mediante la provisión de un producto de programa informático para ejecutar una instrucción de máquina. Según un aspecto de la invención, se proporciona un producto de programa informático conforme a la reivindicación 1. Según otro aspecto de la invención, se proporciona un sistema informático para ejecutar una instrucción de máquina en una unidad central de procesamiento conforme a la reivindicación 10. Según otro aspecto de la invención, se proporciona un método de ejecución de una instrucción de máquina en una unidad central de procesamiento conforme a la reivindicación 17.

También se describen y reivindican en la presente memoria métodos y sistemas relativos uno o más aspectos de la presente invención. Además, también se describen servicios relacionados con uno o más aspectos de la presente invención y pueden ser reivindicados en la presente memoria.

Las características y ventajas adicionales se alcanzan mediante las técnicas de la presente invención. Otras realizaciones y aspectos de la invención se describen en detalle en la presente memoria y se consideran parte de la invención reivindicada.

Breve descripción de los dibujos

Ahora se van a describir realizaciones de la presente invención, a título de ejemplo únicamente, con referencia a los dibujos que se acompañan en los que:

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

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

La Figura 2B representa detalles adicionales de la memoria de la Figura 2A, conforme a un aspecto de la presente invención;

La Figura 3 representa una realización de un formato de una instrucción de Conteo de Carga hasta Límite de Bloque, conforme a un aspecto de la presente invención;

La Figura 4 representa una realización de una lógica asociada a la instrucción de Conteo de Carga hasta Límite de Bloque, conforme a un aspecto de la presente invención;

La Figura 5A representa un ejemplo de al menos una porción de un bloque de memoria para el que se proporciona un conteo, conforme a un aspecto de la presente invención;

La Figura 5B representa un ejemplo de un registro de propósito general que incluye el conteo, conforme a un aspecto de la presente invención;

La Figura 6 representa un ejemplo de un archivo de registro, conforme a un aspecto de la presente invención;

La Figura 7 representa una realización de un producto de programa informático que incorpora uno o más aspectos de la presente invención;

La Figura 8 representa una realización de un sistema informático anfitrión para incorporar y usar uno o más aspectos de la presente invención;

La Figura 9 representa un ejemplo adicional de un sistema informático para incorporar y usar uno o más aspectos de la presente invención;

La Figura 10 representa otro ejemplo de un sistema informático que comprende una red informática para incorporar y usar uno o más aspectos de la presente invención;

La Figura 11 representa una realización de varios elementos de un sistema informático para incorporar y usar uno o más aspectos de la presente invención;

La Figura 12A representa una realización de la unidad de ejecución del sistema informático de la Figura 11 para incorporar y usar uno o más aspectos de la presente invención;

La Figura 12B representa una realización de la unidad de bifurcación del sistema informático de la Figura 11 para incorporar y usar uno o más aspectos de la presente invención;

La Figura 12C representa una realización de la unidad de carga/almacenaje del sistema informático de la Figura 11 para incorporar y usar uno o más aspectos de la presente invención, y

La Figura 13 representa una realización de un sistema informático anfitrión emulado para incorporar y usar uno o más aspectos de la presente invención.

Descripción detallada

Conforme a un aspecto de la presente invención, se proporciona una funcionalidad para determinar una distancia hasta un límite de memoria específico desde una posición especificada (por ejemplo, una dirección de memoria). El límite de memoria es, por ejemplo, el final de un bloque de memoria principal (mencionada también en la presente descripción como almacenaje principal). Un bloque de memoria principal es cualquier bloque de memoria de un tamaño específico. El tamaño específico se menciona también como límite del bloque. El bloque de memoria principal incluye datos, tales como datos de los caracteres, datos de número entero, o cualquier otro tipo de datos.

Los datos de caracteres incluyen, aunque sin limitación, los caracteres alfabéticos, en cualquier lenguaje; dígitos numéricos; puntuación, y/u otros símbolos. Los datos de caracteres pueden ser o no cadenas de datos. Asociados a los datos de caracteres están los estándares, ejemplos de los cuales incluyen, aunque sin limitación, el ASCII (Código Estándar Americano para Intercambio de Información); Unicode que incluye, aunque sin limitación, UTF (Formato de Transformación de Unicode) 8; UTF 16; etc.

En una realización particular, se proporciona una instrucción de Conteo de Carga hasta Límite de Bloque (LCBB) que proporciona el número de bytes de datos hasta un límite de memoria específico desde una dirección especificada en la memoria. En el cálculo del número de bytes (mencionado en la presente memoria como conteo), se usa un tamaño de límite. El tamaño de límite puede ser especificado explícitamente por la instrucción (por ejemplo, un valor de variable en el texto de la instrucción, un valor de texto de instrucción fijo codificado en el opcode, un límite basado en el registro especificado en la instrucción, etc.); o el tamaño de límite puede ser determinado dinámicamente por la máquina. Por ejemplo, la instrucción especifica el tipo de límite, tal como una página o un límite caché, y la máquina determina la línea de caché o el tamaño de página, ya sea mediante un valor previamente especificado o ya sea dinámicamente en base a la información (por ejemplo, mediante búsqueda, por ejemplo, en una Tabla de Traducción-Búfer Separado para determinar el tamaño de la página).

Según una realización, la instrucción de Conteo de Carga hasta Límite de Bloque proporciona el número de bytes de datos que pueden ser, o haber sido, cargados desde la memoria, por ejemplo, en un registro sin cruzar un límite específico de la memoria. El registro puede ser un registro vectorial o cualquier otro tipo de registro.

Un registro vectorial (mencionado también como vector) incluye uno o más elementos, y un elemento de un registro vectorial (también mencionado como vector) es de uno, dos o cuatro bytes de longitud, por ejemplo. Además, un operando vectorial 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 vectorial no necesita ser SIMD, y/o puede incluir un elemento.

Se describe una realización de un entorno informático para incorporar y usar uno o más aspectos de la presente invención con referencia a la Figura 1. Un entorno informático 100 incluye, por ejemplo, un procesador 102 (por ejemplo, una unidad central de proceso), una memoria 104 (por ejemplo, memoria principal), y uno o más dispositivos de entrada salida (E/S) y/o interfaces 106 acoplados entre sí por medio de, por ejemplo, uno o más buses 108 y/u otras conexiones.

En un ejemplo, el procesador 102 está basado en la z/Architecture comercializada por International Business Machines Corporation, y forma parte de un servidor, tal como el servidor de Sistema z, el cual está también

comercializado por International Business Machines Corporation e implementa la z/Architecture. Una realización de la z/Architecture ha sido descrita en una publicación de IBM® titulada “Principios de Operación de la z/Architecture”, Publicación de IBM® núm. SA22-7832-08, Novena Edición, Agosto de 2010.

En un ejemplo, el procesador ejecuta un sistema operativo, tal como z/OS, también comercializado por International Business Machines Corporation. IBM®, Z/ARCHITECTURE® y Z/OS® son marcas registradas de International Business Machines Corporation, Armonk, New York, USA. Otros nombres usados en la presente memoria pueden ser marcas registradas, marcas o nombres de producto de International Business Machines Corporation o de otras compañías.

En un ejemplo adicional, el procesador 102 está basado en Power Architecture comercializada por International Business Machines Corporation. Una realización de la Power Architecture está descrita en “Power ISA™, Versión 2.06 Revisión B”, International Business Machines Corporation, 23 de Julio de 2010.

POWER ARCHITECTURE® es una marca registrada de International Business Machines Corporation

En otro ejemplo adicional, el procesador 102 está basado en una arquitectura Intel comercializado por Intel Corporation. Un ejemplo de arquitectura Intel ha sido descrito en “Intel® 64 y IA-32 Manual del Desarrollador de Arquitecturas: Vol. 2B, Referencia del Conjunto de Instrucciones, A-L”, número de orden 253666-041US, Diciembre de 2011, y en “Intel® 64 y IA-32 Manual del Desarrollador de Arquitecturas: Vol. 2B, Referencia del Conjunto de Instrucciones, 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 Figura 2A. En este ejemplo, un entorno informático 200 incluye, por ejemplo, una unidad de procesamiento central nativa 202, una memoria 204, y uno o más dispositivos de entrada/salida y/o interfaces 206 acoplados entre sí por medio de, por ejemplo, uno o más buses 208 y/u otras conexiones. Como ejemplos, el entorno informático 200 puede incluir un procesador PowerPC, un servidor pSeries o un servidor xSeries comercializados por International Business Machines Corporation, Armonk, New York; un HP Superdome con procesadores Itanium II de Intel comercializados por Hewlett Packard Co., Palo Alto, California, y/u otras máquinas basadas en arquitecturas comercializadas por International Business Machines Corporation, Hewlett Packard, Intel, Oracle, u otros.

La unidad de procesamiento central nativa 202 incluye uno o más registros nativos 210, tal 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 instante particular en el tiempo.

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

Detalles adicionales con relación al código emulador 212 se describen con referencia a la Figura 2B. Instrucciones huésped 250 comprenden instrucciones de software (por ejemplo, instrucciones de máquina) que han sido desarrolladas para ser ejecutadas en una arquitectura distinta a la de la CPU nativa 202. Por ejemplo, las instrucciones huésped 250 pueden haber sido diseñadas para que se ejecuten sobre un procesador de z/Architecture 102, pero en cambio, están siendo emuladas sobre la CPU nativa 202, la cual puede ser, por ejemplo, un procesador Itanium II de Intel. En un ejemplo, el código emulador 212 incluye una unidad 252 de precarga de instrucción para obtener una o más instrucciones huésped 250 desde la memoria 204, y para proporcionar opcionalmente almacenaje intermedio local para las instrucciones obtenidas. También incluye una rutina 254 de traducción de instrucción para determinar el tipo de instrucción huésped que se ha obtenido y para traducir la instrucción huésped en una o más instrucciones nativas 256 correspondientes. Esta traducción incluye, por ejemplo, identificar la función que va a ser llevada a cabo por la instrucción huésped y elegir la(s) instrucción(es) nativa(s) para llevar a cabo 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 hacer que la CPU nativa 202 ejecute una rutina de instrucciones nativas que emulen una o más instrucciones huésped previamente obtenidas y, a la conclusión de esa ejecución, devolver el control a la rutina de precarga de la instrucción para emular la obtención de la siguiente instrucción huésped o de un grupo de instrucciones huésped. La ejecución de las instrucciones nativas 256 puede incluir cargar datos en un registro desde la memoria 204; almacenar datos de nuevo en la memoria desde un registro; o, realizar algún tipo de operación aritmética o lógica, según se determine mediante la rutina de traducción.

Cada rutina está implementada, por ejemplo, en software, el cual está almacenado en memoria y se ejecuta mediante la unidad de procesamiento central nativa 202. En otros ejemplos, se implementa una o más de las operaciones o rutinas en firmware, hardware, software o alguna combinación de los mismos. Los registros del procesador emulado pueden ser emulados usando registros 210 de la CPU nativa o usando posiciones en la memoria 204. En realizaciones, las instrucciones huésped 250, las instrucciones nativas 256 y el código emulador 212 pueden residir en la misma memoria o pueden ser estar distribuidos entre diferentes dispositivos de memoria.

Según se usa en la presente descripción, firmware incluye, por ejemplo, el microcódigo, el milicódigo y/o el macrocódigo del procesador. Éste incluye, por ejemplo, las instrucciones de nivel de hardware y/o las estructuras de datos usadas en la implementación del código máquina de nivel más alto. En una realización, éste incluye, por ejemplo, el código propietario que se proporciona típicamente como microcódigo que incluye software de confianza o como microcódigo específico del hardware subyacente, y controla el acceso del sistema operativo al hardware del sistema.

En un ejemplo, una instrucción huésped 250 que se obtiene, se traduce y se ejecuta, es la instrucción que se describe en la presente memoria. La instrucción, la cual es de una arquitectura (por ejemplo, la z/Architecture), se precarga desde la memoria, se traduce y se representa a modo de secuencia de instrucciones nativas 256 de otra arquitectura (por ejemplo, PowerPC, pSeries, xSeries, Intel, etc.). Estas instrucciones nativas se ejecutan a continuación.

Según se describe en la presente memoria, las instrucciones de Conteo de Carga hasta Límite de Bloque pueden ser implementadas como parte de varias arquitecturas incluyendo, aunque sin limitación, la z/Architecture, Power, Intel, etc. Aunque una realización descrita en la presente memoria se destina a z/Architecture, la instrucción y uno o más aspectos de la presente invención pueden basarse en otras muchas arquitecturas. La z/Architecture es solamente un ejemplo.

Una realización de una instrucción de Conteo de Carga hasta Límite de Bloque ha sido representada en la Figura 3. En un ejemplo, la instrucción 300 de Conteo de Carga hasta Límite de Bloque incluye campos de opcode 302a (por ejemplo, los bits 0-7), 302b (por ejemplo, los bits 40-47) que indican una operación de Conteo de Carga hasta Límite de Bloque; un campo de registro 304 (por ejemplo, los bits 8-11) usado para designar un registro de propósito general (R_1); un campo de índice (X_2) 306 (por ejemplo, los bits 12-15); un campo de base (B_2) 308 (por ejemplo, los bits 16-19); un campo de desplazamiento (D_2) 310 (por ejemplo, los bits 20-31); y un campo de máscara (M_3) 312 (por ejemplo, los bits 32-35). Cada uno de los campos 304-312, en un ejemplo, está separado y es independiente del (de los) campo(s) de opcode. Además, en una realización, éstos están separados y son independientes entre sí; sin embargo, en otras realizaciones, se puede combinar más de un campo. A continuación se describe información adicional sobre el uso de esos campos.

En un ejemplo, los bits seleccionados (por ejemplo, los dos primeros bits) del opcode designado por el campo de opcode 302a especifican la longitud y el formato de la instrucción. En este ejemplo particular, la longitud es de tres semi-palabras, y el formato es una operación de almacenaje de registro e índice con un campo de opcode extendido.

El número de subíndice asociado a un campo de la instrucción indica el operando al que se aplica el campo. Por ejemplo, el número de subíndice 1 asociado a R_1 indica el primer operando, y así sucesivamente. El operando de registro es un registro de longitud, el cual es, por ejemplo, de 128 bits.

En un ejemplo, en una instrucción de operación de almacenaje de registro e índice, los contenidos de registros generales designados mediante los campos X_2 y B_2 se añaden a los contenidos del campo D_2 para formar una segunda dirección de operando. El desplazamiento, D_2, para la instrucción de Conteo de Carga hasta Límite de Bloque, se trata como número entero sin firma de 12 bits, en un ejemplo. La segunda dirección de operando se usa para indicar una posición en la memoria principal; sin embargo, no se usa para direccionar datos, en esta realización.

El campo M_3, en una realización, especifica un código que se usa para señalar la CPU con respecto al tamaño del límite de bloque para calcular el número de posibles bytes a cargar sin cruzar un límite de memoria. Si se especifica un valor reservado, se reconoce una excepción de la especificación. Ejemplos de códigos y de valores correspondientes son como sigue:

Código	**Límite**
0	64 Bytes
1	128 Bytes
2	256 Bytes
3	512 Bytes

4 1 Kbyte

5 2 Kbyte

6 4 Kbyte

En un ejemplo adicional, el tamaño del límite no está incluido en la instrucción, pero en cambio, se determina dinámicamente mediante el procesador que ejecuta la instrucción. Por ejemplo, el campo M_3 especifica el tipo de límite (por ejemplo, línea caché, página, etc.), y en base al tipo y a una o más características del procesador (por ejemplo, tamaño de línea caché para el procesador, tamaño de página para el procesador, etc.), el procesador determina el límite. Como ejemplos, basados en el tipo, el procesador usa un tamaño fijo para el límite (por ejemplo, una línea caché o un tamaño de página fijos predefinidos para el procesador), o en base al tipo, el procesador determina el límite. Por ejemplo, si el tipo es un límite de página, el procesador busca la dirección de inicio en una Tabla de Traducción-Búfer de Instrucciones (TLB) y determina el límite de página a partir de la misma. También existen otros ejemplos. Por ejemplo, el tipo puede ser proporcionado por otro campo de la instrucción o a partir de un control externo a la instrucción.

Durante la ejecución de una realización de la instrucción de Conteo de Carga hasta Límite de Bloque, un número entero binario sin firma (por ejemplo, 64 bits) que contiene el número de bytes que es posible cargar a partir de la posición del segundo operando sin cruzar un límite de bloque específico, limitado a, por ejemplo, el tamaño de un vector que va a ser cargado (por ejemplo, 16) se sitúa en el registro de propósito general especificado en el primer operando.

Con el resultado de la ejecución de la instrucción, se establece un código de condición opcional, tal como por ejemplo:

0 – El operando uno es dieciséis

1 –

2 –

3 – El operando uno es menor de dieciséis

En el ejemplo de instrucción que antecede, la segunda dirección de operando se determina mediante el valor de registro de índice (X_2) + un valor de registro de base (B_2) + un desplazamiento (D_2); sin embargo, en otras realizaciones, se proporciona mediante un valor de registro; una dirección de instrucción + desviación específica del texto de la instrucción; un valor de registro + desplazamiento; o un valor de registro + valor de registro de índice; todo esto solamente como algunos ejemplos.

Otros detalles de una realización del procesamiento de la instrucción de Conteo de Carga hasta Límite de Bloque se describen con referencia a la Figura 4. En un ejemplo, un procesador del entorno informático se encarga de ejecutar esta lógica.

En una realización, se crea inicialmente una máscara de límite (BdyMask), la cual se usa para determinar la cercanía al límite especificado, ETAPA 400. Para crear la máscara, en un ejemplo, se toma una negación del complemento de 2 de un tamaño de límite (BdySize) 402 que crea la máscara de límite 404 (por ejemplo, BdyMask = 0-BdySize). El tamaño del límite se proporciona, en un ejemplo, mediante la instrucción (por ejemplo, el campo M_3); o en otro ejemplo, se determina por medio de la máquina, según se describe en la presente memoria.

A continuación, se calcula una dirección de inicio, la cual indica una posición en la memoria a partir de la cual debe empezar el conteo, ETAPA 410. Como ejemplos, la dirección de inicio 412 puede ser proporcionada por un valor de registro; una dirección de instrucción más una desviación específica del texto de la instrucción; un valor de registro más un desplazamiento; un valor de registro más un valor de registro de índice; o un valor de registro más un valor de registro de índice más un desplazamiento. En la instrucción proporcionada en la presente memoria, la dirección de inicio se proporciona mediante el campo X_2, el campo B_2 y el campo D_2. Es decir, los contenidos de los registros designados mediante X_2 y B_2 se suman al desplazamiento indicado mediante D_2 para proporcionar la dirección de inicio. Las formas indicadas con anterioridad para calcular una dirección de inicio son solamente ejemplos; también son posibles otros ejemplos.

A continuación, se calcula una dirección final que indique una posición en la que se debe detener el conteo, ETAPA 420. Una entrada a este cálculo es, por ejemplo, el tamaño del límite 402, la dirección de inicio 412, el tamaño de vector (vec_size) 414, y la máscara del límite 404. El tamaño de vector es el tamaño de un registro de vector seleccionado o de otro registro (por ejemplo, en bytes; por ejemplo, 16). El registro es, por ejemplo, un registro en el que se pueden cargar datos. En un ejemplo, la dirección final 422 se calcula como sigue:

EndAddress = min(StartAddress + (BdySize - (StartAddress & $\neg$BdyMask)), StartAddress + vec_size).

A continuación, se calcula el conteo, ETAPA 430. Por ejemplo, conteo = EndAddress 422 – StartAddress 412. En un ejemplo adicional, el conteo puede ser calculado a partir de la dirección inicial y sin usar la dirección final. En este ejemplo, conteo = min(16, BdySize – (StartAddress Y NO BdyMask), donde 16 es el tamaño del registro de vector (u otro registro) en bytes. En otros ejemplos, se pueden usar otros tamaños de vector.

Un ejemplo de al menos una porción de un bloque de memoria 500 ha sido representado en la Figura 5A. El bloque de memoria 500 incluye datos desde una dirección seleccionada 502 hasta un límite específico 504. Según se ha indicado, no se cuenta ni se carga ningún dato pasado el límite designado por la línea vertical de puntos 504. Las posiciones más allá del límite no son accesibles y no se adopta ninguna excepción. Además, un conteo de la distancia desde la dirección seleccionada hasta el límite especificado, determinada conforme a un aspecto de la presente invención, se almacena en un registro de propósito general 510, del que se ha representado un ejemplo en la Figura 5B. Según se muestra, en este ejemplo, el conteo es 13 hex, y por lo tanto se almacena 0D en el registro de propósito general 510 que indica que se han contado 13 bytes de datos y que pueden ser almacenados en un registro de vector.

Lo que se ha descrito con anterioridad es un ejemplo de una instrucción de conteo usada para determinar la distancia desde una dirección especificada hasta un límite especificado. Se puede proporcionar una alerta si la dirección especificada está cerca del límite, requiriendo posiblemente, de ese modo, un tratamiento especial.

Según una realización, la instrucción de Conteo de Carga hasta Límite de Bloque se utiliza para determinar cuántos bytes de datos fueron cargados en un registro, tal como un registro de vector. Este conteo puede ser usado a continuación en otro procesamiento.

En una realización, el registro que se carga y para el que se obtiene un conteo es un registro de vector, el cual forma parte de una instalación vectorial. La instalación vectorial proporciona, por ejemplo, vectores de tamaño fijo comprendidos en la gama de uno a dieciséis elementos. Cada vector incluye datos que son operados mediantes instrucciones de vector definidas en la instalación. En una realización, si un vector está formado por múltiples elementos, entonces cada elemento se procesa en paralelo con los otros elementos. La terminación de la instrucción no ocurre hasta que se hay completado el procesamiento de todos los elementos.

A modo de un ejemplo, existen 32 registros de vector y otros tipos de registros pueden mapear a un cuadrante de los registros de vector. Por ejemplo, según se ha mostrado en la Figura 6, si existe un archivo de registro 600 que incluya 32 registros de vector 602 y cada registro tiene una longitud de 128 bits, entonces 16 registros de punto flotante 604, que tienen una longitud de 64 bits, pueden superponerse a los registros de vectores. De ese modo, como ejemplo, cuando se modifica el registro de punto flotante 2, entonces el registro de vector 2 se modifica también. Otros mapeos son también posibles para otros tipos de registros.

En la presente descripción, los términos memoria, memoria principal, almacenaje y almacenaje principal se usan de manera intercambiable, a menos que se indique lo contrario explícitamente o mediante el contexto.

Los detalles de la instrucción de Conteo de Carga hasta Límite de Bloque, y algunos ejemplos de otras instrucciones proporcionadas en la instalación vectorial, se detallan en el Apéndice.

Como podrá apreciar un experto en la materia, uno o más aspectos de la presente invención pueden ser materializados a modo de sistema, método o producto de programa informático. Por consiguiente, uno o más aspectos de la presente invención pueden adoptar la forma de una realización completamente en hardware, una realización completamente en software (incluyendo firmware, software residente, micro-código, etc.) o una realización que combine aspectos de software y de hardware que en general pueda ser mencionada en su conjunto en la presente descripción como un “circuito”, “módulo” o “sistema”. Además, uno o más aspectos de la presente invención pueden adoptar la forma de un producto de programa informático materializado en uno o más medios legibles con ordenador que tengan un código de programa legible con ordenador materializado en los mismos.

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

Haciendo ahora referencia a la Figura 7, en un ejemplo, un producto de programa informático 700 incluye, por ejemplo, uno o más medios de almacenaje legibles con ordenador 702 no transitorios, para almacenar medios de

código de programa legible con ordenador o lógica 704 en los mismos para proporcionar y facilitar uno o más aspectos de la presente invención.

El código de programa materializado en un medio legible con ordenador puede ser transmitido usando un medio apropiado, incluyendo aunque sin limitación los inalámbricos, los cableados, el cable de fibra óptica, RF, etc., o cualquier combinación adecuada de los anteriores.

El código de programa informático para llevar a cabo operaciones para uno o más aspectos de la presente invención puede estar escrito en cualquier combinación de uno o más lenguajes de programación, incluyendo un lenguaje de programación orientado al objeto, tal como Java, Smalltalk, C++ o similar, y lenguajes de programación procedimentales convencionales, tal como el lenguaje de programación "C", ensamblador o lenguajes de programación similares. El código de programa puede ejecutarse completamente en el ordenador del usuario, parcialmente en el ordenador del usuario, como paquete de software autónomo, parcialmente en el ordenador del usuario y parcialmente en un ordenador remoto, o completamente en el ordenador remoto o servidor. En el último escenario, el ordenador remoto puede estar conectado al ordenador del usuario a través de cualquier tipo de red, incluyendo una red de área local (LAN) o una red de área extensa (WAN), o la conexión puede hacerse hasta un ordenador externo (por ejemplo, a través de Internet usando un Proveedor de Servicio de Internet).

En la presente memoria se describe uno o más aspectos de la presente invención con referencia a las ilustraciones de los diagramas de flujo y/o a los diagramas de bloques de métodos, aparatos (sistemas) y productos de programa informático conforme a realizaciones de la invención. Se comprenderá que cada bloque de las ilustraciones de diagramas de flujo y/o de los diagramas de bloques, y las combinaciones de bloques en las ilustraciones de diagramas de flujo y/o diagramas de bloques, pueden ser implementados mediante instrucciones de programa informático. Estas instrucciones de programa informático pueden ser proporcionadas a un procesador de un ordenador de propósito general, un ordenador de propósito especial, u otro aparato de procesamiento de datos programable para producir una máquina, de tal modo 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 funciones/actos especificados en el diagrama de flujo y/o en el bloque o los bloques del diagrama de bloques.

Estas instrucciones de programa informático pueden ser almacenadas también en un medio legible con ordenador que pueda dirigir un ordenador, otro aparato de procesamiento de datos programable, u otros dispositivos para que funcionen de una manera particular, de tal modo que las instrucciones almacenadas en el medio legible con ordenador producen un artículo de fabricación que incluye instrucciones que implementan la función/el acto especificado en el diagrama de flujo y/o en el bloque o los bloques del diagrama de bloques.

Las instrucciones de programa informático pueden ser también cargadas en un ordenador, otro aparato de procesamiento de datos programable, u otros dispositivos para hacer que se lleven a cabo una serie de etapas operativas en el ordenador, otro aparato programable u otros dispositivos para producir un proceso implementado en ordenador de tal modo que las instrucciones que se ejecuten en el ordenador u otro aparato programable proporcionen procesos para implementar las funciones/los actos especificados en el diagrama de flujo y/o en el bloque o los bloques del diagrama de bloques.

El diagrama de flujo y los diagramas de bloques de las Figuras ilustran la arquitectura, funcionalidad y operación de posibles implementaciones de sistemas, métodos y productos de programa informático conforme a varias realizaciones de uno o más aspectos de la presente invención. A este respecto, cada bloque del diagrama de flujo o de los diagramas de bloques puede representar un módulo, un segmento o una porción de código, que comprenda una o más instrucciones ejecutables para implementar la(s) función(es) lógica(s) especificada(s). Se debe apreciar también que, en algunas implementaciones alternativas, las funciones observadas en el bloque pueden ocurrir en un orden distinto al indicado en las Figuras. Por ejemplo, dos bloques mostrados de forma sucesiva pueden ser ejecutados, de hecho, de una manera sustancialmente simultánea, o los bloques pueden ser ejecutados a veces en el orden inverso, dependiendo de la funcionalidad involucrada. Se apreciará también que cada bloque de los diagramas de bloques y/o de la ilustración de diagrama de flujo, y combinaciones de bloques en los diagramas de bloques y/o en la ilustración del diagrama de flujo, pueden ser implementados mediante sistemas a base de hardware de propósito especial que realicen las funciones o actos especificados, o mediante combinaciones de hardware e instrucciones de ordenador de propósito especial.

Adicionalmente a lo anterior, uno o más aspectos de la presente invención pueden ser proporcionados, ofrecidos, desplegados, gestionados, puestos en servicio, etc., por un proveedor de servicio que ofrezca la gestión de entornos de clientes. Por ejemplo, el proveedor de servicio puede crear, mantener, soportar, etc., un código informático y/o una infraestructura de ordenador que realice uno o más aspectos de la presente invención para uno o más clientes. En cambio, el proveedor del servicio puede recibir el pago del cliente bajo una suscripción y/o un acuerdo de honorarios, como ejemplos. Adicionalmente o alternativamente, el proveedor de servicio puede recibir el pago a partir de la venta de contenidos publicitarios a uno o más terceros.

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

Según un aspecto adicional de la presente invención, se puede desplegar una infraestructura de computación que comprenda integrar un código legible con ordenador en un sistema de computación, en la que el código, en combinación con el sistema de computación, sea capaz de llevar a cabo uno o más aspectos de la invención.

Según un aspecto adicional más de la presente invención, se puede proporcionar un proceso para integrar infraestructura de computación que comprenda integrar un código legible con ordenador en un sistema de ordenador. El sistema de ordenador comprende un medio legible con ordenador, en el que el medio informático comprende uno o más aspectos de la presente invención. El código, en combinación con el sistema de ordenador, está capacitado para realizar uno o más aspectos de la presente invención.

Aunque se han descrito en lo que antecede varias realizaciones, éstas son solamente ejemplos. Por ejemplo, entornos de computación de otras arquitecturas pueden incorporar y usar uno o más aspectos de la presente invención. Además, se pueden usar vectores de otros tamaños y otros registros, y se pueden realizar cambios en la instrucción sin apartarse de la presente invención. Además, se pueden usar otras técnicas para calcular la distancia desde una dirección específica hasta un límite particular.

Además, otros tipos de entornos de computación pueden beneficiarse de uno o más aspectos de la presente invención. Como ejemplo, resulta utilizable un sistema de procesamiento de datos adecuado para almacenar y/o ejecutar un código de programa, que incluya al menos dos procesadores acoplados directamente o indirectamente a elementos de memoria a través de un bus de sistema. Los elementos de memoria incluyen, por ejemplo, memoria local empleada durante la ejecución real del código de programa, almacenaje masivo, y memoria caché, que proporcionan el almacenaje temporal de al menos algún código de programa con el fin de reducir el número de veces que el código debe ser recuperado a partir del almacenaje masivo durante la ejecución.

Dispositivos de entrada/Salida o de E/S (incluyendo, aunque sin limitación, los teclados, displays, dispositivos de puntero, DASD, cinta, CDs, DVDs, memorias USB y otros medios de memoria, etc.) pueden ser acoplados al sistema ya sea directamente o ya sea a través de controladores de E/S intermedios. También se pueden acoplar adaptadores de red al sistema para permitir que el sistema de procesamiento de datos se acople a otros sistemas de procesamiento de datos o a impresoras remotas o a dispositivos de almacenaje a través de redes intermedias privadas o públicas. Los módems, módems de cable, y tarjetas de Ethernet son sólo unos pocos de los tipos disponibles de adaptadores de red.

Con referencia a la Figura 8, se han presentado componentes representativos de un sistema 5000 de Ordenador Anfitrión para implementar uno o más aspectos de la presente invención. El ordenador anfitrión 5000 representativo comprende una o más CPUs 5001 en comunicación con memoria de ordenador (es decir, almacenaje central) 5002, así como interfaces de E/S para dispositivos 5011 de medios de almacenaje y redes 5010 para comunicar con otros ordenadores o SANs y similares. La CPU 5001 es acorde con una arquitectura que tiene una arquitectura de instrucciones establecida y una arquitectura de funcionalidad. La CPU 5001 puede tener traducción dinámica de direcciones (DAT) 5003 para transformar direcciones de programa (direcciones virtuales) en direcciones reales de memoria. Una DAT incluye típicamente un búfer de traducción de instrucciones (TLB) 5007 para almacenar en caché traducciones de modo que los accesos de este último al bloque de memoria de ordenador 5002 no requieran el retardo de traducción de la dirección. Típicamente, se emplea una caché 5009 entre la memoria de ordenador 5002 y el procesador 5001. La caché 5009 puede ser jerárquica que tenga una gran caché disponible para más de una CPU y cachés más pequeñas, más rápidas (nivel más bajo) entre la caché grande y cada CPU. En algunas implementaciones, las cachés de nivel más bajo se dividen para proporcionar cachés separadas de bajo nivel para precarga de instrucciones y accesos de datos. En una realización, se precarga una instrucción desde la memoria 5002 por medio de una unidad 5004 de precarga de instrucción a través de una caché 5009. Esta instrucción se descodifica en una unidad de descodificación de instrucciones 5006 y se envía (con otras instrucciones en algunas realizaciones) a la unidad o unidades 5008 de ejecución de instrucciones. 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 instrucción de bifurcación. La instrucción se ejecuta por medio de la unidad de ejecución, accediendo a operandos desde registros específicos de la instrucción o desde la memoria, según se necesite. Si se debe acceder a un operando (cargado o almacenado) desde la memoria 5002, una unidad 5005 de carga/almacenaje gestiona el acceso bajo el control de la instrucción que se está ejecutando. Las instrucciones pueden ser ejecutadas en circuitos de hardware o en microcódigo interno (firmware) o mediante una combinación de ambos.

Según se aprecia, un sistema de ordenador incluye información en almacenaje local (o principal), así como direccionamiento, protección y grabación de referencia y de cambios. Algunos aspectos del direccionamiento incluyen el formato de las direcciones, el concepto de espacios de dirección, los diversos tipos de direcciones, y la manera en la que se traduce un tipo de dirección a otro tipo de dirección. Algo del almacenaje principal incluye posiciones de almacenamiento asignadas permanentemente. El almacenaje principal dota al sistema de almacenaje de datos de acceso rápido directamente direccionable. Tanto los datos como los programas han de ser cargados en el almacenamiento principal (desde dispositivos de entrada) con anterioridad a que los mismos puedan ser procesados.

El almacenamiento principal puede incluir uno o más almacenajes tampón más pequeños, de acceso más rápido, denominados a veces cachés. Una caché está típicamente asociada físicamente a una CPU o a un procesador de E/S. Los efectos, excepto el rendimiento, de la construcción física y del uso de medios de almacenamiento distintos no son generalmente observables por el programa.

Se pueden mantener cachés separadas para instrucciones y para operandos de datos. La información del interior de una caché se mantiene en bytes contiguos sobre un límite integral denominado bloque de caché o línea de caché (o línea, por brevedad). Un modelo puede proporcionar una instrucción de EXTRAER ATRIBUTO DE CACHÉ que devuelva el tamaño de una línea de caché en bytes. Un modelo puede proporcionar también instrucciones de PRECARGAR DATOS y PRECARGAR LONGITUD RELATIVA DE DATOS, que efectúe la precarga de almacenamiento en la caché de datos o de instrucciones, o la liberación de datos desde la caché.

El almacenaje se observa a modo de una cadena horizontal larga de bits. Para la mayor parte de las operaciones, los accesos al almacenamiento avanzan según una secuencia de izquierda a derecha. La cadena de bits se subdivide en unidades de ocho bits. Una unidad de ocho bits se conoce como un byte, el cual es el bloque de construcción básica de todos los formatos de información. Cada posición de byte en el almacenamiento se identifica mediante un único número entero no negativo, el cual es la dirección de esa posición de byte o, simplemente, la dirección de byte. Las posiciones de byte adyacentes tienen direcciones consecutivas, empezando por 0 a la izquierda y avanzando según una secuencia de izquierda a derecha. Las direcciones son números enteros binarios sin firma y son de 24, 31 o 64 bits.

La información se transmite entre el almacenamiento y una CPU o un byte o un grupo de bytes de subsistema de canal, cada vez. A menos que se especifique lo contrario, por ejemplo en la z/Architecture, se direcciona un grupo de bytes en el almacenamiento mediante el byte más a la izquierda del grupo. El número de bytes en el grupo es implícito o bien está especificado explícitamente por la operación que va a ser realizada. Cuando se usa en una operación de CPU, un grupo de bytes se denomina campo. Dentro de cada grupo de bytes, por ejemplo en la z/Architecture, los bits están numerados mediante una secuencia de izquierda a derecha. En z/Architecture, los bits más a la izquierda son a veces mencionados como los bits de “orden alto” y los bits más a la derecha como los bits “de orden bajo”. Los números de bit no son, sin embargo, direcciones de almacenaje. Solamente los bytes pueden ser direccionados. Para operar sobre los bits individuales de un byte en el almacenamiento, se accede al byte completo. Los bits de un byte están numerados del 0 al 7, de izquierda a derecha (por ejemplo, en la z/Architecture). Los bits de una dirección pueden estar numerados como 8-31 o 40-63 para direcciones de 24 bits, o como 1-31 o 33-63 para direcciones de 31 bits; éstos se numeran como 0-63 para direcciones de 64 bits. Dentro de cualquier otro formato de longitud fija de múltiples bytes, los bits que forman el formato están numerados consecutivamente a partir del 0. A los efectos de detección de errores, y preferiblemente para corrección, se puede transmitir uno o más bits de comprobación con cada byte o con un grupo de bytes. Tales bits de comprobación son generados de forma automática por la máquina y no pueden ser controlados directamente por el programa. Las capacidades de almacenamiento se expresan en número de bytes. Cuando la longitud de un campo de operando de almacenamiento está implícita en el código de operación de una instrucción, se dice que el campo tiene una longitud fija, la cual puede ser de uno, dos, cuatro, ocho o dieciséis bytes. Los campos más grandes pueden estar implícitos para algunas instrucciones. Cuando la longitud de un campo de operando de almacenamiento no está implícita sino que se define explícitamente, se dice que el campo tiene una longitud variable. Los operandos de longitud variable pueden variar de longitud mediante incrementos de un byte (o con algunas instrucciones, en múltiplos de dos bytes u otros múltiplos). Cuando la información se dispone en almacenaje, solamente los contenidos de esas posiciones de byte que son reemplazadas están incluidos en el campo designado, incluso aunque la anchura de la trayectoria física para el almacenamiento pueda ser mayor que la longitud del campo que va a ser almacenado.

Algunas unidades de información deben estar sobre un límite integral en el almacenamiento. Se dice que un límite es integral para unidad de información cuando su dirección de almacenamiento es un múltiplo de la longitud de la unidad en bytes. Se dan nombres especiales a los campos de 2, 4, 8 y 16 bytes sobre un límite integral. Una semi-palabra es un grupo de dos bytes consecutivos sobre un límite de dos bytes, y es el bloque de construcción básico de instrucciones. Una palabra es un grupo de cuatro bytes consecutivos sobre un límite de cuatro bytes. Una doble palabra es un grupo de ocho bytes consecutivos sobre un límite de ocho bytes. Una cuádruple palabra es un grupo de 16 bytes consecutivos sobre un límite de 16 bytes. Cuando las direcciones de almacenamiento designan semi-palabras, palabras, dobles palabras y cuádruples palabras, la representación binaria de la dirección contiene uno, dos, tres o cuatro bits cero más a la derecha, respetivamente. Las instrucciones han de estar sobre límites integrales de dos bytes. Los operandos de almacenamiento de la mayor parte de las instrucciones no tienen necesidades de alineamiento de límite.

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

En una realización, la invención puede ser puesta en práctica mediante software (a veces mencionado como código interno licenciado, firmware, micro-código, mili-código, pico-código y similares, cualquiera de los cuales podría ser

acorde con uno o más aspectos de la presente invención). Con referencia a la Figura 8, se puede tener acceso a un código del programa de software que materializa uno o más aspectos de la presente invención mediante el procesador 5001 del sistema anfitrión 5000 desde dispositivos de medios de almacenamiento a largo plazo 5011, tal como una unidad de CD-ROM, una unidad de cinta o un disco duro. El código del programa de software puede estar materializado sobre cualquiera de una diversidad de medios conocidos para su uso con un sistema de procesamiento de datos, tal como un disquete, un disco duro, o un CD-ROM. El código puede estar distribuido sobre tales medios, o puede ser distribuido a los usuarios desde la memoria de ordenador 5002 o desde el almacenamiento de un sistema informático a través de una red 5010 hasta otros sistemas informáticos para su utilización por usuarios de esos otros sistemas.

El código del programa de software incluye un sistema operativo que controla la función y la interacción de los diversos componentes informáticos y uno o más programas de aplicación. El código de programa se pagina normalmente desde el dispositivo de medios de almacenaje 5011 hasta el almacenamiento de ordenador 5002 de velocidad relativamente más alta, donde está disponible para su procesamiento por parte del procesador 5001. Las técnicas y métodos para materializar el código de programa de software en la memoria, sobe medios físicos, y/o distribuir el código de software a través de redes, son bien conocidos y no van a ser explicados aquí con mayor detalle. El código de programa, cuando se ha creado y se ha almacenado en un medio tangible (incluyendo, aunque sin limitación, módulos de memoria electrónica (RAM), memoria flash, Discos Compactos (CDs), DVDs, Cinta Magnética y similares), se menciona con frecuencia como “producto de programa informático”. El medio de producto de programa informático es típicamente legible mediante un circuito de procesamiento, con preferencia en un sistema de ordenador para su ejecución mediante el circuito de procesamiento.

La Figura 9 ilustra una estación de trabajo representativa, o sistema de hardware de servidor, en el que se puede poner en práctica uno o más aspectos de la presente invención. El sistema 5020 de la Figura 9 comprende un sistema de ordenador base 5021 representativo, tal como un ordenador personal, una estación de trabajo o un servidor, que incluye dispositivos periféricos opcionales. El sistema de ordenador base 5021 incluye uno o más procesadores 5026 y un bus empleado para conectar y habilitar la comunicación entre el (los) procesador(es) 5026 y los otros componentes del sistema 5021 de acuerdo con técnicas conocidas. El bus conecta el procesador 5026 a la memoria 5025 y al almacenamiento de largo plazo 5027, el cual puede incluir un disco duro (incluyendo cualquiera de entre medios magnéticos, CD, DVD y memoria Flash, por ejemplo) o una unidad de cinta, por ejemplo. El sistema 5021 podría incluir también un adaptador de interfaz de usuario, el cual conecta el microprocesador 5026, a través del bus, a uno o más dispositivos de interfaz, tal como un teclado 5024, un ratón 5023, una impresora/escáner 5030 y/u otros dispositivos de interfaz, los cuales pueden ser cualquier dispositivo de interfaz de usuario, tal como una pantalla sensible al tacto, un teclado de entrada digitalizado, etc. El bus conecta también un dispositivo de visualización 5022, tal como una pantalla o monitor de LCD, al microprocesador 5026 a través de un adaptador de visualización.

El sistema 5021 puede comunicar con otros ordenadores o redes de ordenadores a través de un adaptador de red capacitado para comunicar 5028 con una red 5029. Ejemplos de adaptadores de red son los canales de comunicaciones, token ring, Ethernet o módems. Alternativamente, el sistema 5021 puede comunicar usando una interfaz inalámbrica, tal como una tarjeta CDPD (datos por paquetes digitales celulares). El sistema 5021 puede estar asociado a esos 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 configuración de cliente/servidor con otro ordenador, etc. Todas esas configuraciones, así como el hardware y el software de comunicaciones apropiados, son conocidos en el estado de la técnica.

La Figura 10 ilustra una red de procesamiento de datos 5040 en la que pueden ser puestos en práctica uno o más aspectos de la presente invención. La red de procesamiento de datos 5040 puede incluir una pluralidad de redes individuales, tal como una red inalámbrica y una red cableada, cada una de las cuales puede incluir una pluralidad de estaciones de trabajo 5041, 5042, 5043, 5044 individuales. Adicionalmente, como podrán apreciar los expertos en la materia, pueden estar incluidas una o más redes LANs, donde una LAN puede comprender una pluralidad de estaciones de trabajo inteligentes acopladas a un procesador anfitrión.

Haciendo aún referencia a la Figura 10, las redes pueden incluir también ordenadores centrales o servidores, tal como un ordenador de pasarela (servidor de cliente 5046) o un servidor de aplicación (servidor remoto 5048 que puede acceder a un depósito de datos y al que también se puede acceder directamente desde una estación de trabajo 5045). Un ordenador de pasarela 5046 sirve como punto de entrada a cada red individual. Se necesita una pasarela cuando se conecta un protocolo de red a otro. La pasarela 5046 puede estar acoplada preferiblemente a otra red (Internet 5047, por ejemplo) por medio de un enlace de comunicaciones. La pasarela 5046 puede estar también acoplada directamente a una o más estaciones de trabajo 5041, 5042, 5043, 5044 usando un enlace de comunicaciones. El ordenador de pasarela puede ser implementado usando un Sistema eServer™ de IBM, servidor disponible en International Business Machines Corporation.

Haciendo simultáneamente referencia a la Figura 9 y la Figura 10, el código de programación de software que puede materializar uno o más aspectos de la presente invención puede ser consultado por el procesador 5026 del sistema 5020 desde medios de almacenaje de largo plazo 5027, tal como una unidad de CD-ROM o un disco duro. El código

de programación de software puede estar materializado sobre cualquiera de una diversidad de medios conocidos para su uso con un sistema de procesamiento de datos, tal como un disquete, un disco duro, o un CD-ROM. El código puede estar distribuido sobre tales medios, o puede ser distribuido a usuarios 5050, 5051 desde la memoria o el almacenamiento de un sistema de ordenador a través de una red hasta otros sistemas de ordenador para su uso por los usuarios de esos otros sistemas.

Alternativamente, el código de programación puede ser materializado en la memoria 5025, y consultado por el procesador 5026 usando el bus del procesador. Dicho código de programación incluye un sistema operativo que controla la función y la interacción de los diversos componentes informáticos y uno o más programas de aplicación 5032. El código de programa se pagina normalmente desde medios de almacenaje 5027 hasta una memoria de alta velocidad 5025 donde está disponible para su procesamiento por el procesador 5026. Las técnicas y métodos para materializar el código de programación de software en la memoria, sobre medios físicos, y/o distribuir el código de software a través de redes, son vienen conocidos y no van a ser explicados aquí con mayor detalle. El código de programa, cuando se ha creado y se ha almacenado en un medio tangible (incluyendo, aunque sin limitación, módulos de memoria electrónica (RAM), memoria flash, Discos Compactos (CDs), DVDs, Cinta Magnética y similares), se menciona con frecuencia como “producto de programa informático”. El medio de producto de programa informático es típicamente legible mediante un circuito de procesamiento, con preferencia en un sistema de ordenador para su ejecución por el circuito de procesamiento.

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

Con referencia a la Figura 11, se ha representado un ejemplo de realización de procesador en relación con el procesador 5026. Típicamente se emplea uno o más niveles de caché 5053 para bloques de memoria tampón con el fin de mejorar el rendimiento del procesador. La caché 5053 es una memoria intermedia de alta velocidad que mantiene líneas de caché de datos de memoria que probablemente van a ser usados. Las líneas de caché típicas son de 64, 128 ó 256 bytes de datos de memoria. Se emplean con frecuencia cachés separadas para disponer en caché instrucciones con más frecuencia que para disponer datos en caché. La coherencia de caché (sincronización de copias de líneas en memoria y en las cachés), viene proporcionada con frecuencia por varios algoritmos “snoop” que son bien conocidos en el estado de la técnica. El almacenamiento de memoria principal 5025 de un sistema de procesador se menciona con frecuencia como caché. En un sistema de procesador que tiene 4 niveles de caché 5053, el almacenaje principal 5025 se menciona a veces como la caché de nivel 5 (L5) puesto que es típicamente más rápido y solamente mantiene una porción del almacenamiento no volátil (DASD, cinta, etc.) que está disponible para un sistema de ordenador. El almacenamiento principal 5025 “dispone en caché” páginas de datos paginadas en, y fuera del, almacenamiento principal 5025 por el sistema operativo.

Un contador de programa (contador de instrucciones) 5061 contiene la ruta de la dirección de la instrucción actual que va a ser ejecutada. Un contador de programa en un procesador de z/Architecture es de 64 bits y puede ser recortado hasta 31 ó 24 bits para soportar límites de direccionamiento anteriores. Un contador de programa se materializa típicamente en una PSW (palabra de estado de programa) de un ordenador de tal modo que persista durante conmutación de contexto. De ese modo, un programa en curso, que tiene un valor de contador de programa, puede ser interrumpido, por ejemplo, por el sistema operativo (conmutación de contexto desde el entorno del programa hasta el entorno del sistema operativo). La PSW del programa mantiene el valor del contador de programa mientras el programa no está activo, y el contador de programa (en la PSW) del sistema operativo se usa mientras el sistema operativo se está ejecutando. Típicamente, el contador de programa se incrementa en una cantidad igual al número de bytes de la instrucción actual. Instrucciones RISC (Cálculo del Conjunto Reducido de Instrucciones) son típicamente de longitud fija mientras que las instrucciones CISAC (Cálculo del Conjunto Complejo de Instrucciones) son típicamente de longitud variable. Las instrucciones de la z/Architecture de IBM son instrucciones CISC que tienen una longitud de 2, 4 ó 6 bytes. El contador de programa 5061 se modifica ya sea mediante una operación de conmutación de contexto o ya sea mediante una operación de toma de bifurcación de una instrucción de bifurcación, por ejemplo. En una operación de conmutación de contexto, el valor del contador de programa actual se salva en la palabra de estado de programa junto con otra información de estado acerca del programa que se está ejecutando (tal como códigos de condición), y se carga un nuevo valor de contador de programa que apunte a una instrucción de un nuevo módulo de programa que se va a ejecutar. Una operación de toma de bifurcación se efectúa con el fin de permitir que el programa tome decisiones o forme un bucle dentro del programa cargando el resultado de la instrucción de bifurcación en el contador de programa 5061.

Típicamente se emplea una unidad de precarga de instrucción 5055 para precargar instrucciones del lado del procesador 5026. La unidad de precarga, o bien precarga las “siguientes instrucciones secuenciales”, instrucciones objetivo de instrucciones de toma de bifurcación, o bien las primeras instrucciones de un programa a continuación a una conmutación de contexto. Las modernas unidades de precarga de instrucciones emplean con frecuencia técnicas de precarga para precargar especulativamente instrucciones en base a la probabilidad de que vayan a ser usadas las instrucciones precargadas. Por ejemplo, una unidad de precarga puede precargar 16 bytes de instrucción

que incluya la siguiente instrucción secuencial y bytes adicionales de instrucciones secuenciales adicionales.

Las instrucciones precargadas son ejecutadas a continuación por el procesador 5026. En una realización, la(s) instrucción(es) precargada(s) se hace(n) pasar a una unidad de envío 5056 de la unidad de precarga. La unidad de envío descodifica la(s) instrucción(es) y envía información acerca de la(s) instrucción(es) descodificadas(s) a unidades 5057, 5058, 5060 apropiadas. Una unidad de ejecución 5057 recibirá típicamente información acerca de las instrucciones aritméticas descodificadas desde la unidad de precarga de instrucción 5055, y realizará operaciones aritméticas sobre operandos conforme al opcode de la instrucción. Se proporcionan operandos a la unidad de ejecución 5057 ya sea preferiblemente desde la memoria 5025, desde la arquitectura de los registros 5059, o bien desde un campo inmediato de la instrucción que se estar ejecutando. Los resultados de la ejecución, cuando se almacenan, son almacenados ya sea en la memoria 5025, en registros 5059 o bien en otro hardware de máquina (tal como registros de control, registros de PSW y similares).

Un procesador 5026 tiene típicamente una o más unidades 5057, 5058, 5060 para ejecutar la función de la instrucción. Con referencia a la Figura 12A, una unidad de ejecución 5057 puede comunicar con registros de arquitectura general 5059, con una unidad de descodificación/envío 5056, con una unidad de almacenamiento de carga 5060, y con otras unidades de procesador 5065 a través de lógica de interfaz 5071. Una unidad de ejecución 5057 puede emplear varios circuitos de registro 5067, 5068, 5069 para mantener información sobre la que operará la unidad lógica aritmética (ALU) 5066. La ALU lleva a cabo operaciones aritméticas tales como suma, resta, multiplicación y división, así como una función lógica tal como Y, O y O-exclusiva (XOR), giro y desplazamiento. Con preferencia, la ALU soporta operaciones especializadas que son dependientes del diseño. Otros circuitos pueden proporcionar otras instalaciones de arquitectura 5072 incluyendo códigos de condición y lógica de soporte de recuperación, por ejemplo. Típicamente, el resultado de la operación de una ALU se conserva en un circuito de registro de salida 5070 que puede enviar el resultado a una diversidad de otras funciones de procesamiento. Existen muchas disposiciones de unidades de procesador, estando destinada la presente descripción solamente a proporcionar una comprensión representativa de una realización.

Una instrucción de ADD, por ejemplo, podría ser ejecutada en una unidad de ejecución 5057 que tiene una funcionalidad aritmética y lógica mientras que una instrucción de punto flotante, por ejemplo, podría ser ejecutada en una ejecución de punto flotante que tenga una capacidad de punto flotante especializada. Con preferencia, una unidad de ejecución opera sobre operandos identificados por una instrucción llevando a cabo una función definida de opcode sobre los operandos. Por ejemplo, una instrucción de ADD puede ser ejecutada por una unidad de ejecución 5057 sobre operandos encontrados en dos registros 5059 identificados por campos de registro de la instrucción.

La unidad de ejecución 5057 realiza la adición aritmética sobre dos operandos y almacena el resultado en un tercer operando donde el tercer operando puede ser un tercer registro o uno de los dos registros fuente. La unidad de ejecución utiliza con preferencia una Unidad Lógica Aritmética (ALU) 5066 que está capacitada para llevar a cabo una diversidad de funciones lógicas tal como Desplazamiento, Giro, Y, O y XOR, así como una diversidad de funciones algebraicas que incluyen cualquiera de entre sumar, restar, multiplicar, dividir. Algunas ALUs 5066 están diseñadas para operaciones escalares y algunas para punto flotante. Los datos pueden ser Gran Endian (donde el byte menos significativo está en la dirección de byte más alto) o Pequeño Endian (donde el byte menos significativo está en la dirección de byte más bajo), dependiendo de la arquitectura. La z/Architecture de IBM es Gran Endian. Los campos firmados pueden ser signo y magnitud, complemento del 1 o complemento del 2, dependiendo de la arquitectura. Un número complemento del 2 resulta ventajoso debido a que la ALU no necesita designar una capacidad de resta puesto que cualquier valor negativo o un valor positivo en el complemento del 2 solamente requiere una adición dentro de la ALU. Los números se describen normalmente en taquigrafía, donde un campo de 12 bits define una dirección de un bloque de 4.096 bytes y se describe normalmente como un bloque de 4 Kbyte (kilo-byte), por ejemplo.

Con referencia a la Figura 12B, la información de instrucción de bifurcación para ejecutar una instrucción de bifurcación se envía típicamente a una unidad de bifurcación 5058 que con frecuencia emplea un algoritmo de predicción de bifurcación tal como una tabla histórica de bifurcación 5082 para predecir el resultado de la bifurcación con anterioridad a que se completen otras operaciones condicionales. El objetivo de la instrucción de bifurcación actual será precargado y ejecutado especulativamente con anterioridad a que se completen las operaciones condicionales. Cuando las operaciones condicionales se han completado, las instrucciones de bifurcación ejecutadas especulativamente son completadas o bien descartadas en base a las condiciones de la operación condicional y al resultado especulado. Una instrucción de bifurcación típica puede probar códigos de condición y bifurcación para una dirección objetivo si los códigos de condición cumplen el requisito de bifurcación de la instrucción de bifurcación, una dirección objetivo puede ser calculada en base a varios números que incluyen los encontrados en campos de registro o en un campo inmediato de la instrucción, por ejemplo. La unidad de bifurcación 5058 puede emplear una ALU 5074 que tenga una pluralidad de circuitos de registro de entrada 5075, 5076, 5077, y un circuito de registro de salida 5080. La unidad de bifurcación 5058 puede comunicar con registros generales 5059, descodificar la unidad de envío 5056 u otros circuitos 5073, por ejemplo.

La ejecución de un grupo de instrucciones puede ser interrumpida por una diversidad de razones incluyendo una

conmutación de contexto iniciada por un sistema operativo, una excepción de programa o un error que causen una conmutación de contexto, una señal de interrupción de E/S que provoque una conmutación de contexto o una actividad multihilo de una pluralidad de programas (en un entorno multi-hilo), por ejemplo. Con preferencia, una acción de conmutación de contexto salva información de estado acerca de un programa que se está ejecutando en ese momento y a continuación carga información de estado acerca de otro programa que se esté involucrado. La información de estado puede ser salvada en registros de hardware o en una memoria, por ejemplo. La información de estado comprende, con preferencia, un valor de contador de programa que apunta a una siguiente instrucción que va a ser ejecutada, códigos de condición, información de traducción de memoria y el contenido de la arquitectura de un registro. Una actividad de conmutación de contexto puede ser ejercitada mediante circuitos de hardware, programas de aplicación, programas de sistema operativo o código de firmware (microcódigo, pico-código o un código interno licenciado (LIC)), solo o en combinación.

Un procesador accede a operandos conforme a métodos definidos de instrucción. La instrucción puede proporcionar un operando inmediato que use el valor de una porción de la instrucción, puede proporcionar uno o más campos de registro que apuntan explícitamente ya sea a registros de propósito general o ya sea a registros de propósito especial (registros de punto flotante, por ejemplo). La instrucción puede utilizar registros implícitos identificados mediante un campo de opcode como operandos. La instrucción puede utilizar posiciones de memoria para operandos. Se puede proporcionar una posición de memoria de un operando mediante un registro, un campo inmediato, o una combinación de registros y campo inmediato según se ejemplifica mediante la instalación de desplazamiento largo de z/Architecture en donde la instrucción define un registro de base, un registro de índice y un campo inmediato (campo de desplazamiento) que se suman entre sí para proporcionar la dirección del operando en la memoria, por ejemplo. La posición implica típicamente en la presente descripción una posición en la memoria principal (almacenamiento principal) a menos que se indique otra cosa.

Con referencia a la Figura 12C, un procesador accede a un almacenamiento usando una unidad de carga/ almacenamiento 5060. La unidad de carga/almacenamiento 5060 puede realizar una operación de carga obteniendo la dirección del operando objetivo en la memoria 5053, y cargando el operando en un registro 5059 u otra posición de memoria 5053, o puede realizar una operación de almacenaje mediante la obtención de la dirección del operando objetivo en la memoria 5053 y almacenando datos obtenidos desde un registro 5059 u otra posición de memoria 5053 en la posición del operando objetivo en la memoria 5053. La unidad de carga/almacenamiento 5060 puede ser especulativa y puede acceder a la memoria en una secuencia que está fuera de orden con relación a la secuencia de instrucciones, aunque no obstante la unidad de carga/almacenamiento 5060 se destina a mantener la apariencia respecto a programas en que las instrucciones se ejecutaron por orden. Una unidad de carga/almacenamiento 5060 puede comunicar con registros generales 5059, con una unidad de descodificación/envío 5056, con una interfaz de caché/memoria 5053 o con otros elementos 5083, y comprender varios circuitos de registro, ALUs 5085 y lógica de control 5090 para calcular direcciones de almacenaje y para proporcionar secuenciación de canalización para mantener las operaciones por orden. Algunas operaciones pueden estar fuera de orden pero la unidad de carga/almacenamiento proporciona funcionalidad para hacer que las operaciones fuera de orden aparezcan respecto al programa como que han sido realizadas por orden, según se conoce bien en el estado de la técnica.

Preferiblemente, las direcciones que un programa de aplicación “ve” se mencionan con frecuencia como direcciones virtuales. Las direcciones virtuales son a veces mencionadas como “direcciones lógicas” y “direcciones efectivas”. Estas direcciones virtuales son virtuales debido a que las mismas son redirigidas a una posición de memoria física por medio de una de una diversidad de tecnologías de traducción dinámica de dirección (DAT) que incluyen, aunque sin limitación, prefijar simplemente una dirección virtual con un valor de desviación, traducir la dirección virtual por medio de una o más tablas de traducción, comprendiendo las tablas de traducción, con preferencia, al menos una tabla de segmento y una tabla de página sola o en combinación, teniendo preferiblemente la tabla de segmento un puntero de entrada a la tabla de página. En la z/Architecture, se proporciona una jerarquía de traducción que incluye una primera tabla de región, una segunda tabla de región, una tercera tabla de región, una tabla de segmento y una tabla de página opcional. El rendimiento de la traducción de dirección se mejora con frecuencia utilizando un búfer de traducción de instrucción (LTB) que comprende entradas que mapean una dirección virtual respecto a una posición de memoria física asociada. Las entradas se crean cuando la DAT traduce una dirección virtual usando las tablas de traducción. El uso posterior de la dirección virtual puede utilizar entonces la entrada de la TLB rápida en vez de los accesos de la tabla de traducción secuencial lenta. El contenido de la TLB puede ser gestionado mediante una diversidad de algoritmos de sustitución que incluyen LRU (menos recientemente usados).

En caso de que el procesador sea un procesador de un sistema multi-procesador, cada procesador tiene la responsabilidad de mantener recursos compartidos, tal como E/S, cachés, TLBs y memoria, interconectados por coherencia. Típicamente, se usarán tecnologías “snoop” en el mantenimiento de coherencia de caché. En un entorno snoop, cada línea de caché puede estar marcada como que está en uno cualquiera de entre un estado compartido, un estado exclusivo, un estado modificado, un estado inválido y similares, con el fin de facilitar el intercambio.

Unidades de E/S 5054 (Figura 11) dotan al procesador de medios para conexión a dispositivos periféricos incluyendo cinta, disco, impresoras, displays y redes, por ejemplo. Las unidades de E/S se presentan con frecuencia al programa informático mediante unidades de software. En ordenadores centrales, tales como el Sistema z® de

IBM®, los adaptadores de canal y los adaptadores de sistema abierto son unidades de E/S del ordenador principal que proporcionan las comunicaciones entre el sistema operativo y los dispositivos periféricos.

Además, otros tipos de entornos informáticos pueden beneficiarse de uno o más aspectos de la presente invención. Como ejemplo, un entorno puede incluir un emulador (por ejemplo, software u otros mecanismos de emulación), en el que se emule una arquitectura particular (incluyendo, por ejemplo, ejecución de instrucción, arquitectura de funciones, tal como traducción de dirección, y arquitectura de registros) o se emule un subconjunto de los mismos (por ejemplo, en un sistema de ordenador nativo que tiene un procesador y una memoria). En un entorno de ese tipo, una o más funciones de emulación del emulador pueden implementar uno o más aspectos de la presente invención, incluso aunque un ordenador que ejecute el emulador pueda tener una arquitectura diferente a las capacidades que son emuladas. Como ejemplo, en modo emulación, la instrucción u operación específica que se está emulando, se descodifica, y se construye una función de emulación apropiada para implementar la instrucción o la operación individual.

En un entorno de emulación, un ordenador anfitrión incluye, por ejemplo, una memoria para almacenar instrucciones y datos; una unidad de precarga de instrucciones para precargar instrucciones desde una memoria y para proporcionar, opcionalmente, memoria intermedia local para la instrucción precargada; una unidad de descodificación de instrucción para recibir las instrucciones precargadas y para determinar el tipo de instrucciones que han sido precargadas; 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 nuevo en la memoria desde un registro; o llevar a cabo algún tipo de operación aritmética o lógica, según se determine mediante la unidad de descodificación. En un ejemplo, cada unidad se implementa en software. Por ejemplo, las operaciones que son realizadas por las unidades se implementan como una o más subrutinas dentro del software emulador.

Más en particular, en un ordenador central, se usa arquitectura de instrucciones de máquina mediante programadores, normalmente en la actualidad programadores “C”, con frecuencia a modo de aplicación compiladora. Estas instrucciones almacenadas en el medio de almacenamiento pueden ser ejecutadas de forma nativa en un Servidor IBM® de z/Architecture, o alternativamente en máquinas que ejecuten otras arquitecturas. Éstas pueden ser emuladas en los servidores de ordenador principal de IBM® existentes y futuros, y en otras máquinas de IBM® (por ejemplo, servidores de Power Systems y Servidores de System x®). Éstas pueden ser ejecutadas en máquinas que gestionan Linux sobre una amplia diversidad de máquinas que usan hardware fabricado por IBM®, Intel®, AMD® y otros. Además de la ejecución sobre ese hardware bajo una z/Archirtecture, se puede usar también Linux como máquinas que usan emulación mediante Hércules, UMX, o FSI (Fundamental Software, Inc.), donde la ejecución está generalmente en un modo de emulación. En modo de emulación, el software de emulación se ejecuta mediante un procesador nativo para emular la arquitectura de un procesador emulado.

El procesador nativo ejecuta típicamente software de emulación que comprende ya sea firmware o bien un sistema operativo nativo para realizar emulación del procesador emulado. El software de emulación es responsable de precargar y ejecutar instrucciones de la arquitectura del procesador emulado. El software de emulación mantiene un contador de programa emulado para mantener el curso de los límites de instrucción. El software de emulación puede precargar una o más instrucciones de máquina emuladas en un instante, y convertir la una o más instrucciones de máquina emuladas en un grupo correspondiente de instrucciones de máquina nativa para su ejecución por el procesador nativo. Las instrucciones convertidas pueden ser dispuestas en caché de tal modo que se pueda realizar una conversión más rápida. No obstante, el software de emulación se destina a mantener las reglas de arquitectura de la arquitectura de procesador emulado con el fin de asegurar que los sistemas operativos y las aplicaciones escritas para el procesador emulado, funcionan correctamente. Además, el software de emulación está destinado a proporcionar recursos identificados por la arquitectura de procesador emulado incluyendo, aunque sin limitación, registros de control, registros de propósito general, registros de punto flotante, función de traducción dinámica de dirección incluyendo tablas de segmento y tablas de página por ejemplo, mecanismos de interrupción, mecanismos de conmutación de contexto, relojes con la Hora del Día (TOD) y arquitectura de interfaces para subsistemas de E/S de tal modo que un sistema operativo o un programa de aplicación diseñados para que se ejecuten en el procesador emulado, puedan ser gestionados en el procesador nativo que tenga el software de emulación.

Una instrucción específica que se está emulando, se descodifica, y se llama a una subrutina para que realice la función de la instrucción individual. Una función de software de emulación que emula una función del procesador emulado, se implementa por ejemplo en una subrutina o un controlador “C”, o en algún otro método de provisión de un controlador para el hardware específico tal y como estará al alcance de los expertos en la materia tras la comprensión de la descripción de la realización preferida. Varias patentes de emulación de software y de hardware que incluyen, pero sin limitación, la Carta de Patente U.S. número 5.551.013, titulada “Multiprocesador para Emulación de Hardware”, de Beausoleil et al.; y la Carta de Patente U.S. número 6.009.261, titulada “Procesamiento de Rutinas Objetivo Almacenadas para Emular Instrucciones Incompatibles sobre un Procesador Objetivo”, de Scalzi et al; y la Carta de Patente U.S. número 5.574.873, titulada “Descodificación de Instrucción Huésped para Acceder Directamente a Rutinas de Emulación que Emulan las Instrucciones Huésped”, de Davidian et al.; y la Carta de Patente U.S. número 6.308.255 titulada “Bus de Multiprocesamiento Simétrico y Conjunto de Chip Usado para Soporte de Coprocesador que Permite que se Ejecute un Código No-Nativo se Ejecute en un Sistema”, de Gorishek et al.; y la Carta de Patente U.S. número 6.463.582, titulada “Traductor de Código Objeto de Optimización Dinámica

para Emulación de Arquitectura y Método de Traducción de Código Objeto de Optimización Dinámica", de Lethin et al.; y la Carta de Patente U.S. número 5.790.825, titulada "Método para Emular Instrucciones Huésped en un Ordenador Anfitrión Mediante Re-Compilación Dinámica de Instrucciones Anfitrión", de Eric Traut, y muchas otras, ilustran una diversidad de formas conocidas para conseguir la emulación de una arquitectura de formato de instrucción para una máquina diferente, para una máquina objetivo disponible para los expertos en la materia.

En la Figura 13, se proporciona un ejemplo de un sistema de ordenador anfitrión emulado 5092 que emula un sistema de ordenador anfitrión 5000' de una arquitectura de anfitrión. En el sistema de ordenador anfitrión emulado 5092, el procesador anfitrión (CPU) 5091 es un procesador anfitrión emulado (o procesador anfitrión virtual) y comprende un procesador de emulación 5093 que tiene una arquitectura de conjunto de instrucciones nativas diferente a la del procesador 5091 del ordenador anfitrión 5000'. El sistema de ordenador anfitrión emulado 5092 tiene memoria 5094 accesible para el procesador de emulación 5093. En el ejemplo de realización, la memoria 5094 se particiona en una porción de memoria de ordenador anfitrión 5096 y una porción de rutinas de emulación 5097. La memoria de ordenador anfitrión 5096 está disponible para programas del ordenador anfitrión emulado 5092 conforme a la arquitectura del ordenador anfitrión. El procesador de emulación 5093 ejecuta instrucciones nativas sobre una arquitectura de un conjunto de instrucciones de una arquitectura distinta a la del procesador emulado 5091, las instrucciones nativas obtenidas desde la memoria de rutinas de emulación 5097, y puede acceder a una instrucción anfitrión para ejecución desde un programa en la memoria de ordenador anfitrión 5096 empleando una o más instrucción(es) obtenida(s) en una secuencia y de acceso/rutina de descodificación que puede descodificar la(s) instrucción(es) anfitrión a la(s) que ha accedido para determinar una rutina de ejecución de instrucción nativa para emular la función de la instrucción anfitrión consultada. Otras instalaciones que se definen respecto a la arquitectura del sistema de ordenador anfitrión 5000', pueden ser emuladas mediante la arquitectura de rutinas de las instalaciones, incluyendo tales instalaciones como registros de propósito general, registros de control, traducción dinámica de dirección y soporte de subsistema de E/S y caché de procesador, por ejemplo. Las rutinas de emulación pueden también tener la ventaja de funciones disponibles en el procesador de emulación 5093 (tal como registros generales y traducción dinámica de direcciones virtuales) para mejorar el rendimiento de las rutinas de emulación. También se puede proporcionar hardware especial y motores de descarga para ayudar al procesador 5093 en la emulación de la función del ordenador anfitrión 5000'.

La terminología usada en la presente descripción tiene la finalidad de describir realizaciones particulares solamente y no se pretende que sea limitativa de la invención. Según se utiliza en la presente memoria, las formas en singular "un", "una" y "el/la" está previsto que incluyan también las formas en plural, a menos que el contexto indique claramente otra cosa. Se comprenderá además que los términos "comprende" y/o "comprendiendo", cuando se usan en la presente descripción, especifican la presencia de características establecidas, números enteros, etapas, operaciones, elementos, y/o componentes, pero no excluyen la presencia o la adición de una o más de otras características, números enteros, etapas, operaciones, elementos, componentes y/o grupos de los mismos.

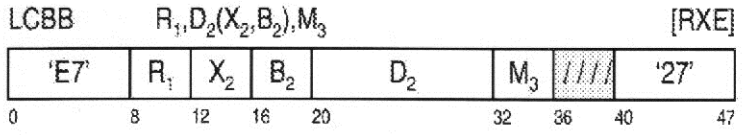
Las estructuras correspondientes, materiales, actos y equivalentes de todos los medios o etapa más los elementos de función en las reivindicaciones que siguen, si aparecen, está previsto que incluyan cualquier estructura, material o acto para llevar a cabo la función en combinación con otros elementos reivindicados según se reivindica específicamente. La descripción de uno o más aspectos de la presente invención ha sido presentada con fines de ilustración y de descripción, pero no se pretende que sea exhaustiva ni esté limitada a la invención en la forma que se ha descrito. Muchas modificaciones y variaciones resultarán evidentes para el experto en la materia sin apartarse del alcance de la invención. La realización ha sido elegida y descrita a efectos de explicar mejor los principios de la invención y su aplicación práctica, y para permitir que otros expertos en la materia comprendan la invención para diversas realizaciones con diversas modificaciones según sea adecuado para el uso particular contemplado.

Apéndice

Instrucciones de Instalación Vectorial

A menos que se especifique otra cosa, todos los operandos son operandos de registro vectorial. Una “V” en la sintaxis de ensamblador designa un operando vectorial.

Conteo de Carga hasta Límite de Bloque



Un número entero binario sin firma de 32 bits que contiene el número de bytes posibles para su cargar desde la segunda posición de operando sin cruzar un límite de bloque especificado, limitado a dieciséis, se coloca en el primer operando.

El desplazamiento se trata como un número entero sin firma de 12 bits.

La segunda dirección de operando no se usa para datos de dirección.

El campo M_3 especifica un código que se usa para señalar la CPU como el tamaño del límite de bloque para calcular el número de bytes posibles cargados. Si se especifica un valor reservado, entonces se reconoce una excepción de especificación.

Límite de Código

0 64 Bytes

1 128 Bytes

2 256 Bytes

3 512 Bytes

4 1K Bytes

5 2K Bytes

6 4K Bytes

7-15 Reservado

Código de Condición Resultante

0 El operando uno es dieciséis

1 --

2 --

3 El operando uno es menor de dieciséis

Código de Condición Resultante

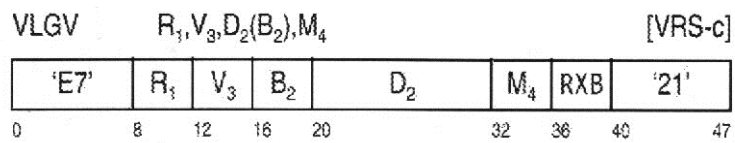
Excepciones del Programa

* Operación si la instalación de extensión vectorial no está instalada

* Especificación

Nota de Programación: Se espera que el Conteo de Carga hasta Límite de Bloque se use junto con Carga Vectorial hasta Límite de Bloque para determinar el número de bytes que se han cargados.

Carga Vectorial GR desde el Elemento VR



El elemento del tercer operando de tamaño especificado por el valor de ES en el campo M4 e indexado por la

segunda dirección de operando, se sitúa en la primera posición de operando. El tercer operando es un registro vectorial. El primer operando es un registro general. Si el índice especificado por la segunda dirección de operando es mayor que el elemento de número más alto en el tercer operando, del tamaño de elemento especificado, los datos en el primer operando son impredecibles.

Si el elemento de registro vectorial es más pequeño que una doble palabra, el elemento está alineado a la derecha en el registro general de 64 bits y los ceros rellenan los bits restantes.

La segunda dirección de operando no se usa para direccionar datos; en cambio se usan los 12 bits más a la derecha de la dirección para especificar el índice de un elemento dentro del segundo operando.

El campo M_4 especifica el control de tamaño del elemento (ES). El control del ES especifica el tamaño de los elementos en los operandos de registro vectorial. Si se especifica un valor reservado, se reconoce una excepción de la especificación.

0 - Byte

1 – Semi-palabra

2 – Palabra

3 – Doble palabra

4-15 – Reservado sin cambio

Código de Condición Resultante: El código se mantiene sin cambio

Excepciones del Programa:

* Datos con DXC FE, Registro Vectorial

* Operación si la instalación de extensión vectorial no está instalada

* Especificación (Valor de ES reservado)

* Restricción de Transacciones

Mnemotécnicos Extendidos

VLGVB $R_1,V_3,D_2(B2)$ VLGV R1,V3,D2(B2),0

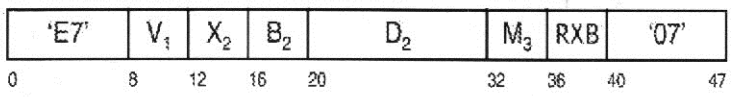
VLGVH R1,V3,D2(B2) VLGV R1,V3,D2(B2),1

VLGVF R1,V3,D2(B2) VLGV R1,V3,D2(B2),2

VLGVG R1,V3,D2(B2) VLGV R1,V3,D2(B2),3

Carga Vectorial hasta Límite de Bloque

VLBB $V_1,D_2(X_2,B_2),M_3$ [VRX]



El primer operando se carga empezando en el elemento de byte indexado con cero, con bytes procedentes del segundo operando. Si se encuentra una condición de límite, el resto del primer operando es impredecible. Las excepciones de acceso no son reconocidas sobre bytes no cargados.

El desplazamiento para VLBB se trata como número entero sin firma de 12 bits.

El campo M_3 especifica un código que se usa para señalar la CPU como el tamaño de límite de bloque a cargar. Si se especifica un valor reservado, se reconoce una excepción de especificación.

Límite de Código

0 64 Bytes

1 128 Bytes

2 256 Bytes

3 512 Bytes

4 1K Bytes

5 2K Bytes

6 4K Bytes

7-15 Reservado

Código de Condición Resultante: El código permanece sin cambio

Excepciones de Programa:

* Acceso (precarga, operando 2)

* Datos con DXC FE, Registro Vectorial

* Operación si la instalación de extensión vectorial no está instalada

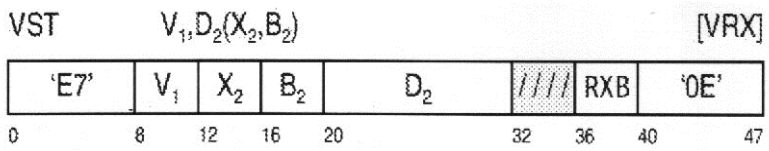
* Especificación (Código de Límite de Bloque Reservado)

* Limitación de Transacciones

Notas de Programación

1. En determinadas circunstancias, los datos pueden ser cargados pasado el límite del bloque. Sin embargo, esto ocurrirá solamente si no hay excepciones de acceso sobre esos datos.

Almacenamiento Vectorial



El valor de 128 bits en el primer operando se almacena en la posición de almacenaje especificada por el segundo operando. El desplazamiento para VST se trata como un número entero sin firma de 12 bits.

Código de Condición Resultante: El código se mantiene sin cambio

Excepciones de Programa

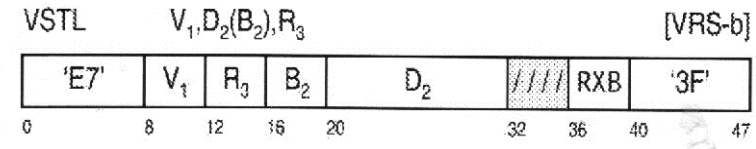
* Acceso (almacenamiento, operando 2)

* Datos con DXC FE, Registro Vectorial

* Operación si la instalación de extensión vectorial no está instalada

* Limitación de Transacciones

Almacenamiento de Vector con Longitud



Avanzando de izquierda a derecha, los bytes del primer operando se almacenan en la segunda posición de operando. El tercer operando especificado del registro general contiene un número entero sin firma de 32 bits que contiene un valor que representa el byte indexado más alto a almacenar. Si el tercer operando contiene un valor más alto o igual que el índice de byte más alto del vector, se almacenan todos los bytes del primer operando.

Las excepciones de acceso se reconocen solamente sobre bytes almacenados.

Este desplazamiento para Almacenamiento de Vector con Longitud se trata como un número entero sin firma de 12 bits.

Código de Condición Resultante: El código de condición permanece sin cambio

Excepciones de Programa

* Acceso (almacenamiento, operando 2)

* Datos con DXC FE, Registro Vectorial

* Operación si la instalación de extensión vectorial no está instalada

* Limitación de Transacciones

Descripción de RXB

Todas las instrucciones vectoriales tienen un campo en los bits 36-40 de la instrucción etiquetada como RXB. El campo contiene los bits más significativos para todos los operandos designados del registro vectorial. Los bits para designación de registro no especificados por la instrucción, se reservan y deben ser establecidos como cero; en otro caso, el programa puede que no opere de manera compatible en el futuro. El bit más significativo está concatenado a la izquierda de la designación de registro de cuatro bits para crear la designación de registro vectorial de cinco bits.

Los bits se definen como sigue:

0. Bit más significativo para la designación del registro vectorial en los bits 8-11 de la instrucción.

1. Bit más significativo para la designación del registro vectorial en los bits 12-15 de la instrucción

2. Bit más significativo para la designación del registro vectorial en los bits 16-19 de la instrucción.

3. Bit más significativo para la designación del registro vectorial en los bits 32-35 de la instrucción.

Control de Habilitación Vectorial

Los registros vectoriales y las instrucciones se pueden usar solamente si tanto el control de habilitación vectorial (bit 46) como el control de registro de AFP (bit 45) en el cero del registro de control se establecen en uno. Si la instalación vectorial está instalada y se ejecuta una instrucción vectorial sin el conjunto de bits de habilitación, se reconoce una excepción de datos con DXC FEX hex. Si la instalación de vector no está instalada, se reconoce una excepción de operación.

REIVINDICACIONES

1.- Un producto de programa informático para ejecutar una instrucción de máquina en una unidad central de procesamiento, comprendiendo el producto de programa informático: un medio de almacenamiento legible con ordenador, legible por medio de un circuito de procesamiento, e instrucciones de almacenamiento para su ejecución por el circuito de procesamiento para llevar a cabo un método que comprende:

obtener, mediante el procesador, una instrucción de máquina para su ejecución, estando la instrucción de máquina definida para su ejecución con ordenador conforme a una arquitectura de ordenador, comprendiendo la instrucción de máquina (300):

al menos un campo de opcode (302a) para proporcionar un opcode;

un campo de registro (304) para ser usado a efectos de designar un registro (R_1), comprendiendo el registro un primer operando, y

al menos un campo (306) para indicar una posición de un segundo operando, comprendiendo el segundo operando una dirección de inicio del bloque de memoria principal,

y caracterizado porque el opcode identifica una operación de Conteo de Carga hasta Límite de Bloque, siendo la operación de Conteo de Carga hasta Límite de Bloque para calcular una distancia desde una posición en la memoria principal hasta un límite de un bloque de memoria principal,

y porque el método comprende además ejecutar la instrucción de máquina para realizar la operación de Conteo de Carga hasta Límite de Bloque, comprendiendo la ejecución:

determinar la distancia desde la posición del segundo operando hasta el límite del bloque de memoria principal, comprendiendo la determinación de la distancia contar mediante el procesador el número de bytes desde la posición del segundo operando hasta el límite del bloque de memoria, y

disponer un valor que representa la distancia en el primer operando, siendo este valor un resultado de ejecutar la operación de Conteo de Carga hasta Límite de Bloque.

2.- El producto de programa informático de la reivindicación 1, en donde la instrucción de máquina comprende además un campo de máscara, donde el campo de máscara especifica el límite.

3.- El producto de programa informático de la reivindicación 2, en donde el límite de bloque es un límite de entre una pluralidad de límites especificables por el campo de máscara.

4.- El producto de programa informático de cualquier reivindicación anterior, en donde la ejecución comprende además determinar dinámicamente el límite, usando la determinación dinámica un tipo específico de límite y una o más características del procesador.

5.- El producto de programa informático de la reivindicación 1, en donde la determinación de la distancia comprende una dirección final en la que el conteo ha de detenerse.

6.- El producto de programa informático de la reivindicación 5, en donde la determinación de la dirección final comprende calcular la dirección final como sigue:

dirección final = mínimo de (dirección de inicio + (tamaño del límite – (dirección de inicio Y NO máscara de límite)), dirección de inicio + tamaño del registro), en donde el tamaño del límite es el límite, la máscara del límite es igual a 0 – tamaño del límite, y el tamaño del registro es una longitud de un registro seleccionado.

7.- El producto de programa informático de la reivindicación 5, en donde la determinación de la distancia comprende calcular el valor restando la dirección de inicio de la dirección final.

8.- El producto de programa informático de la reivindicación 2, en donde la determinación de la distancia comprende calcular el valor como sigue:

valor = mínimo de (tamaño del registro, tamaño del límite - (posición del segundo operando Y máscara del límite)), en donde el tamaño del registro es una longitud de un registro seleccionado, el tamaño del límite es el límite, y la máscara del límite es igual a 0 – tamaño del límite.

9.- El producto de programa informático de cualquier reivindicación anterior, en donde el al menos un campo comprende un campo de desplazamiento, un campo de base y un campo de índice, teniendo el campo de base y el campo de índice para localizar registros generales contenidos para ser sumados a los contenidos del campo de desplazamiento para formar una dirección del segundo operando, proporcionando la dirección del segundo operando la posición del segundo operando.

10.- Un sistema de ordenador (100) para:

ejecutar una instrucción de máquina en una unidad central de procesamiento (102), comprendiendo el sistema de ordenador:

una memoria, y

un procesador en comunicación con la memoria, en donde el sistema de ordenador está configurado para llevar a cabo un método, comprendiendo el método:

obtener, mediante un procesador, una instrucción de máquina (300) para su ejecución, estando la instrucción de máquina definida para su ejecución con ordenador conforme a una arquitectura de ordenador, comprendiendo la instrucción de máquina (300):

al menos un campo de opcode (302a) para proporcionar un opcode;

un campo de registro (304) para ser usado a efectos de designar un registro (R_1), comprendiendo el registro un primer operando, y

al menos un campo (306) para indicar una posición de un segundo operando, comprendiendo el segundo operando una dirección de inicio del bloque de memoria principal,

y caracterizado porque el opcode identifica una operación de Conteo de Carga hasta Límite de Bloque, siendo la operación de Conteo de Carga hasta Límite de Bloque para calcular una distancia desde una posición en la memoria principal hasta un límite de un bloque de memoria principal,

y porque el método comprende además:

ejecutar la instrucción de máquina, comprendiendo la ejecución:

determinar la distancia desde la posición del segundo operando hasta el límite del bloque de memoria principal, comprendiendo la determinación de la distancia contar mediante el procesador el número de bytes desde la posición del segundo operando hasta el límite del bloque de memoria, y

disponer un valor que representa la distancia en el primer operando, siendo este valor un resultado de ejecutar la operación de Conteo de Carga hasta Límite de Bloque.

11.- El sistema de ordenador de la reivindicación 10, en donde la instrucción de máquina comprende además un campo de máscara, donde el campo de máscara especifica el límite.

12.- El sistema de ordenador de la reivindicación 10 o la reivindicación 11, en donde la ejecución comprende además determinar dinámicamente el límite, usando la determinación dinámica un tipo específico de límite y una o más características del procesador.

13.- El sistema de ordenador de la reivindicación 12, en donde la determinación de la distancia comprende determinar una dirección final en la que el conteo ha de detenerse.

14.- El sistema de ordenador de la reivindicación 13, en donde la determinación de la dirección final comprende calcular la dirección final como sigue:

dirección final = mínimo de (dirección de inicio + (tamaño del límite – (dirección de inicio Y NO máscara del límite)), dirección de inicio + tamaño del registro), en donde el tamaño del límite es el límite, la máscara del límite es igual a 0 – tamaño del límite, y el tamaño del registro es una longitud de un registro seleccionado.

15.- El sistema de ordenador de la reivindicación 13, en donde la determinación de la distancia comprende calcular el valor restando la dirección de inicio de la dirección final.

16.- El sistema de ordenador de cualquiera de las reivindicaciones 12 a 15, en donde la determinación de la distancia comprende calcular el valor como sigue:

valor = mínimo de (tamaño del registro, tamaño del límite – (posición del segundo operando Y máscara del límite)), en donde el tamaño del registro es una longitud de un registro seleccionado, el tamaño del límite es el límite, y la máscara del límite es igual a 0 – tamaño del límite.

17.- Un método de ejecución de una instrucción de máquina en una unidad central de procesamiento, comprendiendo el método:

obtener, mediante un procesador, una instrucción de máquina (300) para su ejecución, estando la instrucción de máquina definida para su ejecución con ordenador conforme a una arquitectura de ordenador, comprendiendo la

instrucción de máquina (300):

al menos un campo de opcode (302a) para proporcionar un opcode;

un campo de registro (304) para ser usado a efectos de designar un registro (R_1), comprendiendo el registro un primer operando, y

al menos un campo (306) para indicar una posición de un segundo operando, comprendiendo el segundo operando una dirección de inicio del bloque de memoria principal,

y caracterizado porque el opcode identifica una operación de Conteo de Carga hasta Límite de Bloque, siendo la operación de Conteo de Carga hasta Límite de Bloque para calcular una distancia desde una posición en la memoria principal hasta un límite de un bloque de memoria principal,

y porque el método comprende además ejecutar la instrucción de máquina, comprendiendo la ejecución:

determinar la distancia desde la posición del segundo operando hasta el límite del bloque de memoria principal, comprendiendo la determinación de la distancia contar mediante el procesador el número de bytes desde la posición del segundo operando hasta el límite del bloque de memoria, y

disponer un valor que representa la distancia en el primer operando, siendo este valor un resultado de ejecutar la operación de Conteo de Carga hasta Límite de Bloque.

18.- El método de la reivindicación 17, en donde la posición del segundo operando es una dirección de inicio en la memoria desde la que han de ser contados los datos, y en donde la determinación de la distancia comprende:

determinar una dirección final en la que el conteo ha de detenerse, y

calcular el valor restando la dirección de inicio de la dirección final.

19.- El método de la reivindicación 17 o la reivindicación 18, en donde la determinación de la distancia comprende calcular el valor como sigue:

valor = mínimo de (tamaño del registro, tamaño del límite – (posición del segundo operando Y máscara del límite)), en donde el tamaño del registro es una longitud de un registro seleccionado, el tamaño del límite es el límite, y la máscara del límite es igual a 0 – tamaño del límite.

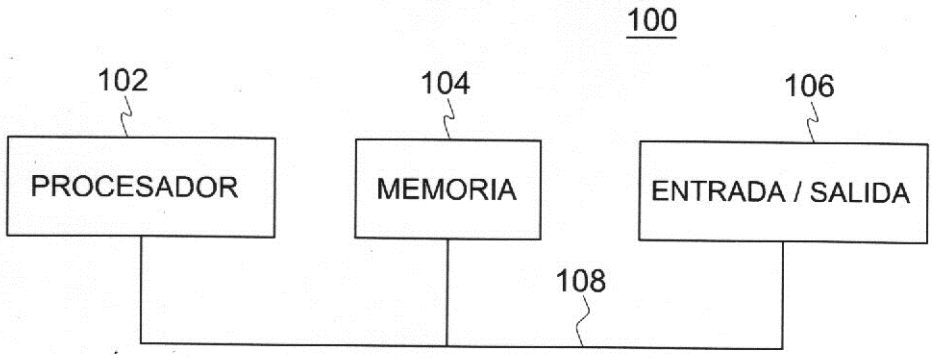


FIG. 1

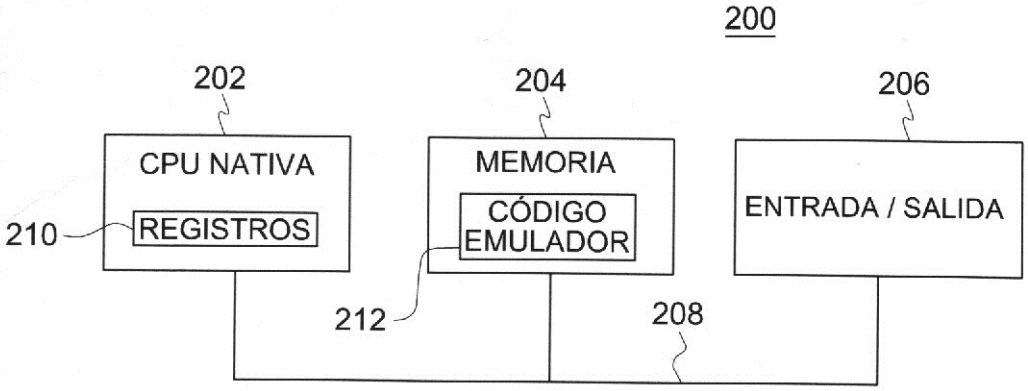


FIG. 2A

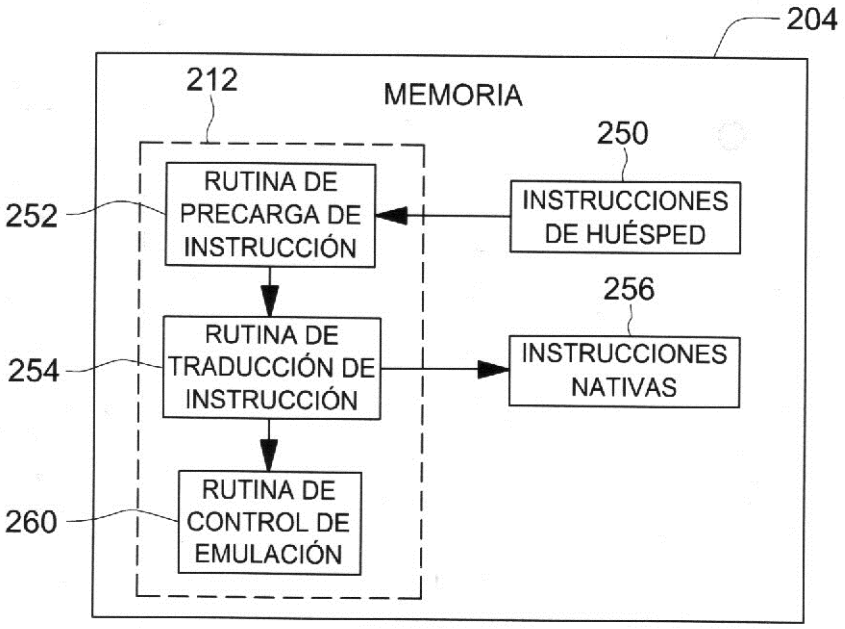


FIG. 2B

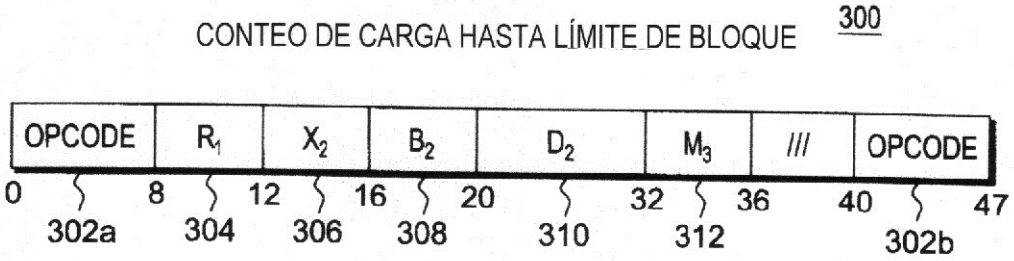

CONTEO DE CARGA HASTA LÍMITE DE BLOQUE
300
OPCODE
R1
X2
B2
D2
M3
///
OPCODE
0
8
12
16
20
32
36
40
47
302a
304
306
308
310
312
302b

FIG. 3

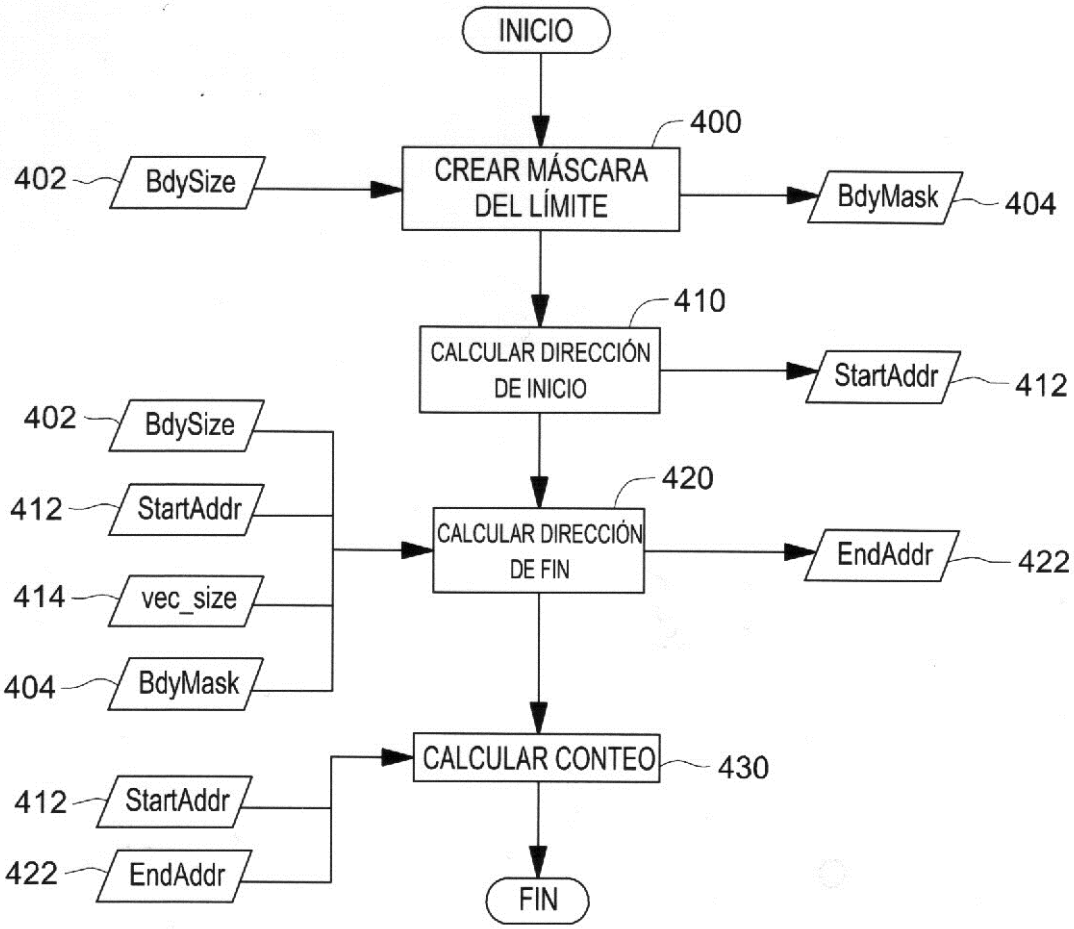

INICIO
400
402
BdySize
CREAR MÁSCARA DEL LÍMITE
BdyMask
404
410
CALCULAR DIRECCIÓN DE INICIO
StartAddr
412
402
BdySize
412
StartAddr
420
CALCULAR DIRECCIÓN DE FIN
EndAddr
422
414
vec_size
404
BdyMask
CALCULAR CONTEO
430
412
StartAddr
422
EndAddr
FIN

FIG. 4

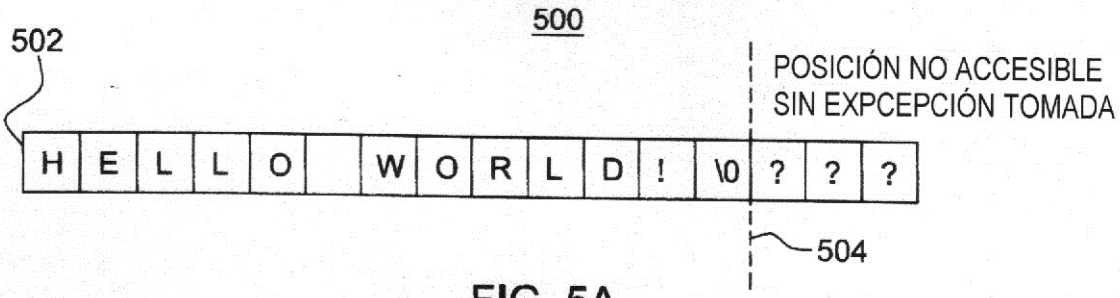


FIG. 5A

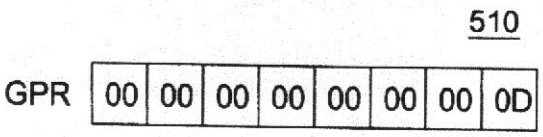


FIG. 5B

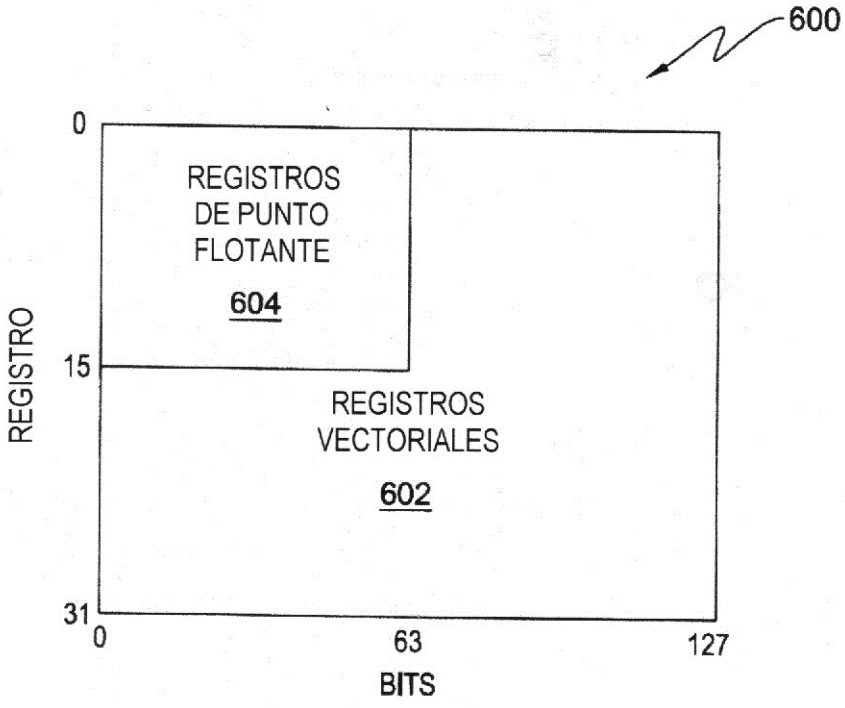


FIG. 6


PRODUCTO
DE PROGRAMA
INFORMÁTICO
700
704
LÓGICA DE CÓDIGO
DE PROGRAMA
MEDIO DE
ALMACENAJE
LEGIBLE CON
ORDENADOR
702

FIG. 7

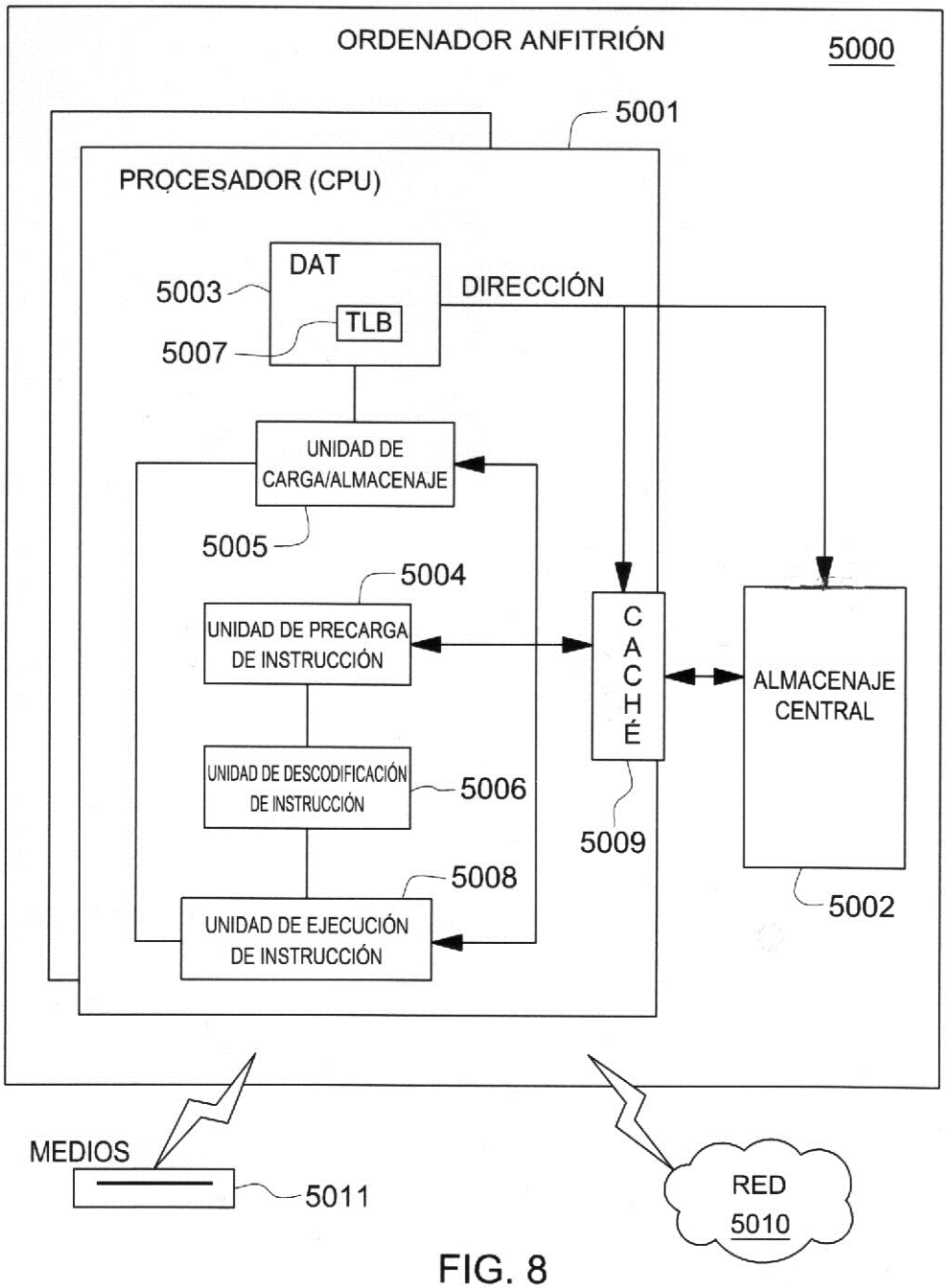

ORDENADOR ANFITRIÓN
5000
5001
PROCESADOR (CPU)
DAT
5003
TLB
5007
DIRECCIÓN
UNIDAD DE CARGA/ALMACENAJE
5005
5004
UNIDAD DE PRECARGA DE INSTRUCCIÓN
UNIDAD DE DESCODIFICACIÓN DE INSTRUCCIÓN
5006
5008
UNIDAD DE EJECUCIÓN DE INSTRUCCIÓN
CACHÉ
5009
ALMACENAJE CENTRAL
5002
MEDIOS
5011
RED
5010

FIG. 8

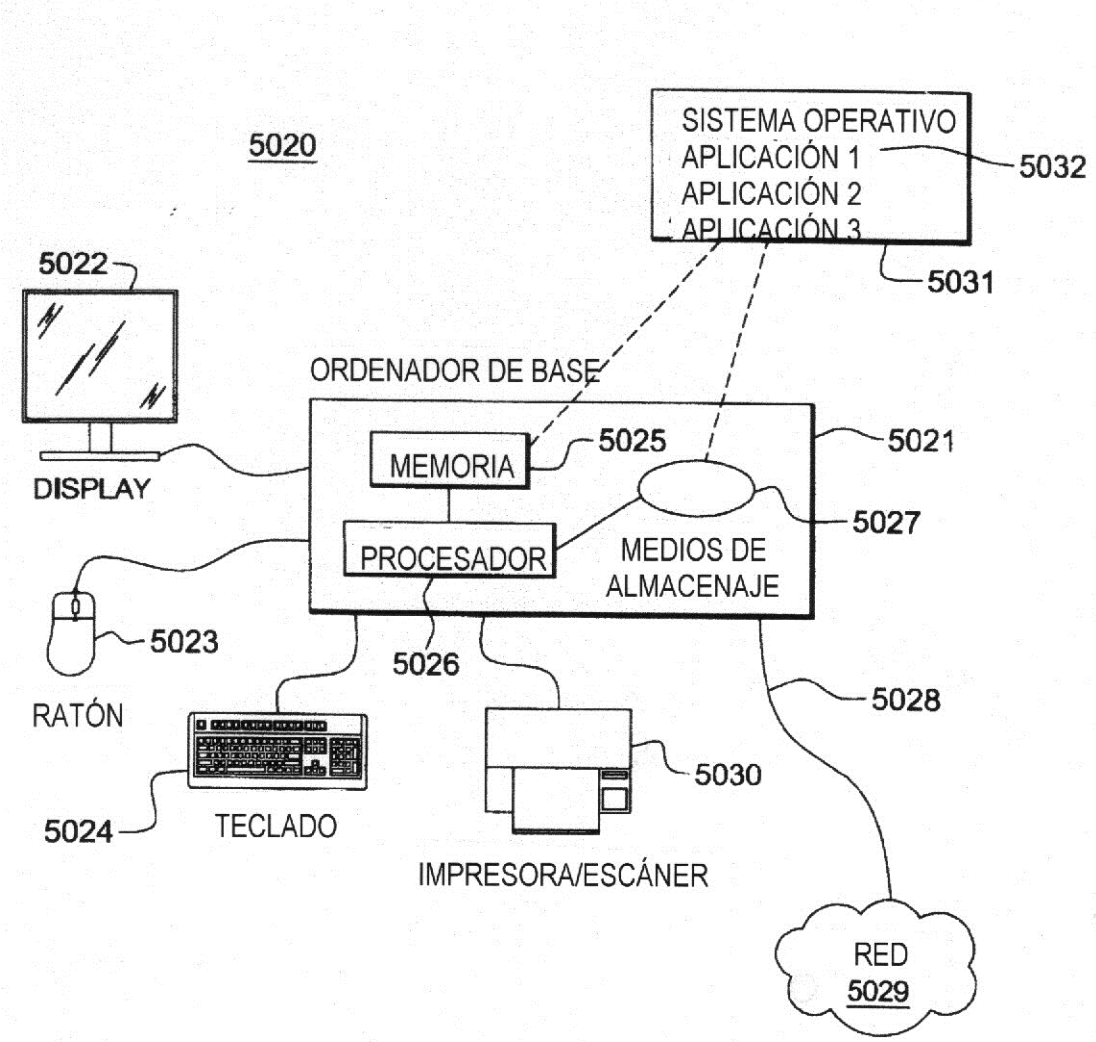

5020
SISTEMA OPERATIVO
APLICACIÓN 1
APLICACIÓN 2
APLICACIÓN 3
5032
5031
5022
ORDENADOR DE BASE
MEMORIA
5025
5021
DISPLAY
5027
PROCESADOR
MEDIOS DE ALMACENAJE
5023
5026
5028
RATÓN
5030
5024
TECLADO
IMPRESORA/ESCÁNER
RED
5029

FIG. 9

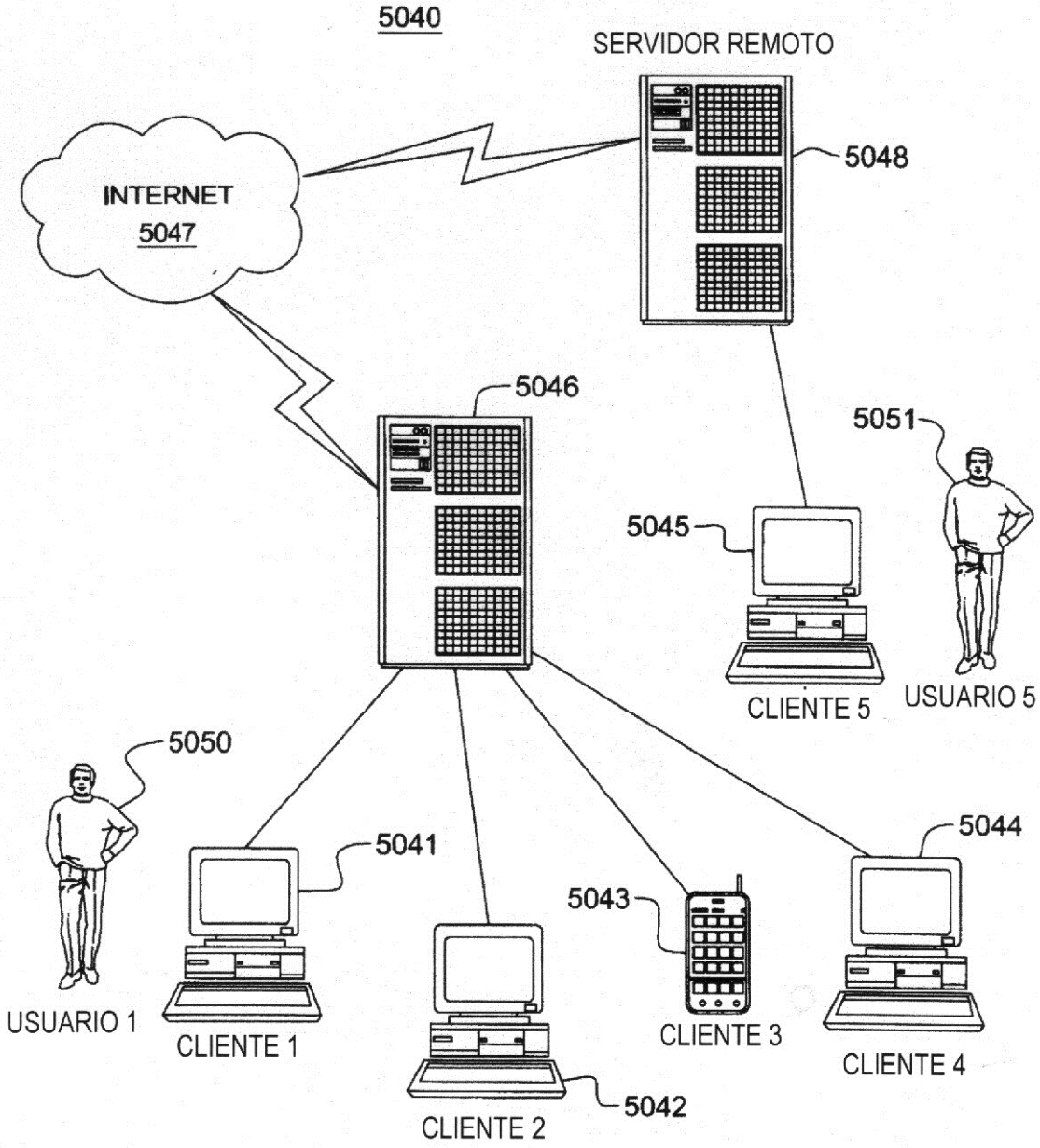

5040
SERVIDOR REMOTO
5048
INTERNET
5047
5046
5051
5045
CLIENTE 5
USUARIO 5
5050
5041
5044
5043
USUARIO 1
CLIENTE 1
CLIENTE 3
CLIENTE 4
CLIENTE 2
5042

FIG. 10

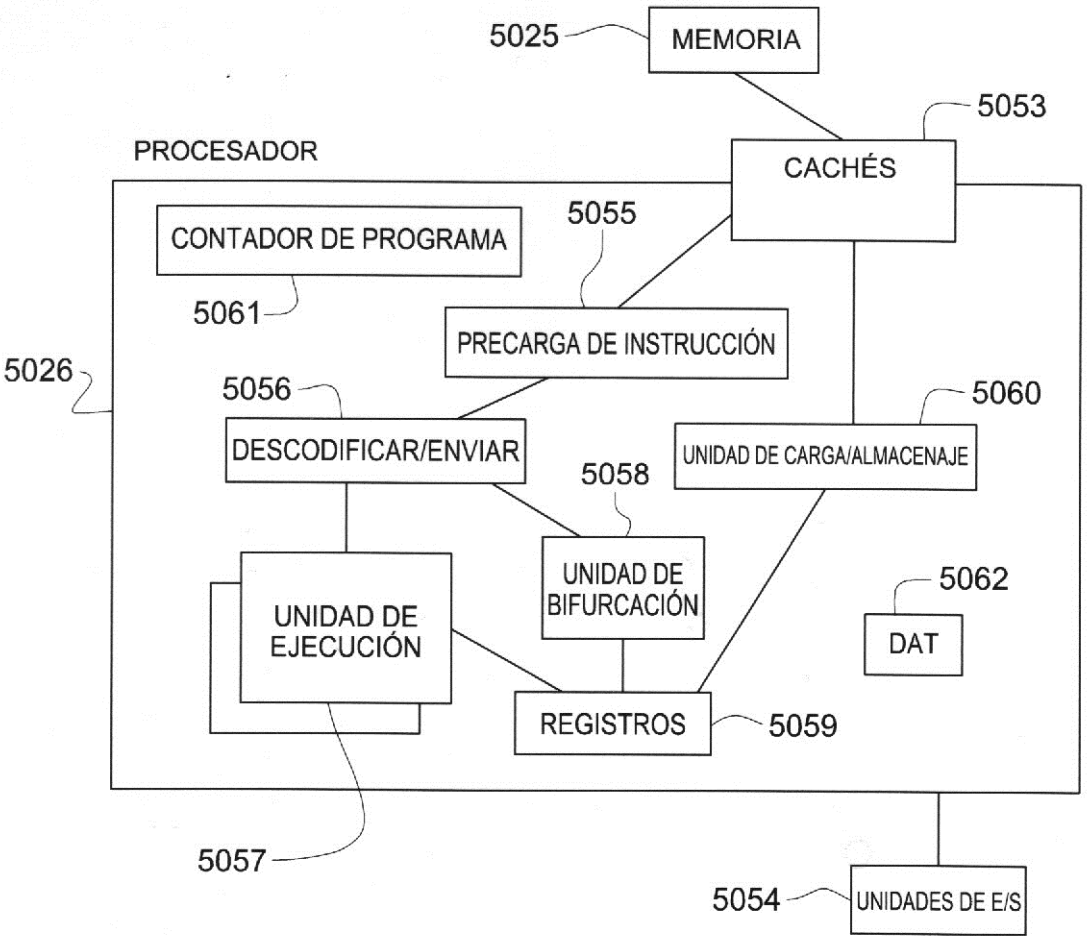

5025
MEMORIA
5053
PROCESADOR
CACHÉS
CONTADOR DE PROGRAMA
5055
5061
PRECARGA DE INSTRUCCIÓN
5026
5056
5060
DESCODIFICAR/ENVIAR
UNIDAD DE CARGA/ALMACENAJE
5058
UNIDAD DE BIFURCACIÓN
5062
UNIDAD DE EJECUCIÓN
DAT
REGISTROS
5059
5057
5054
UNIDADES DE E/S

FIG. 11

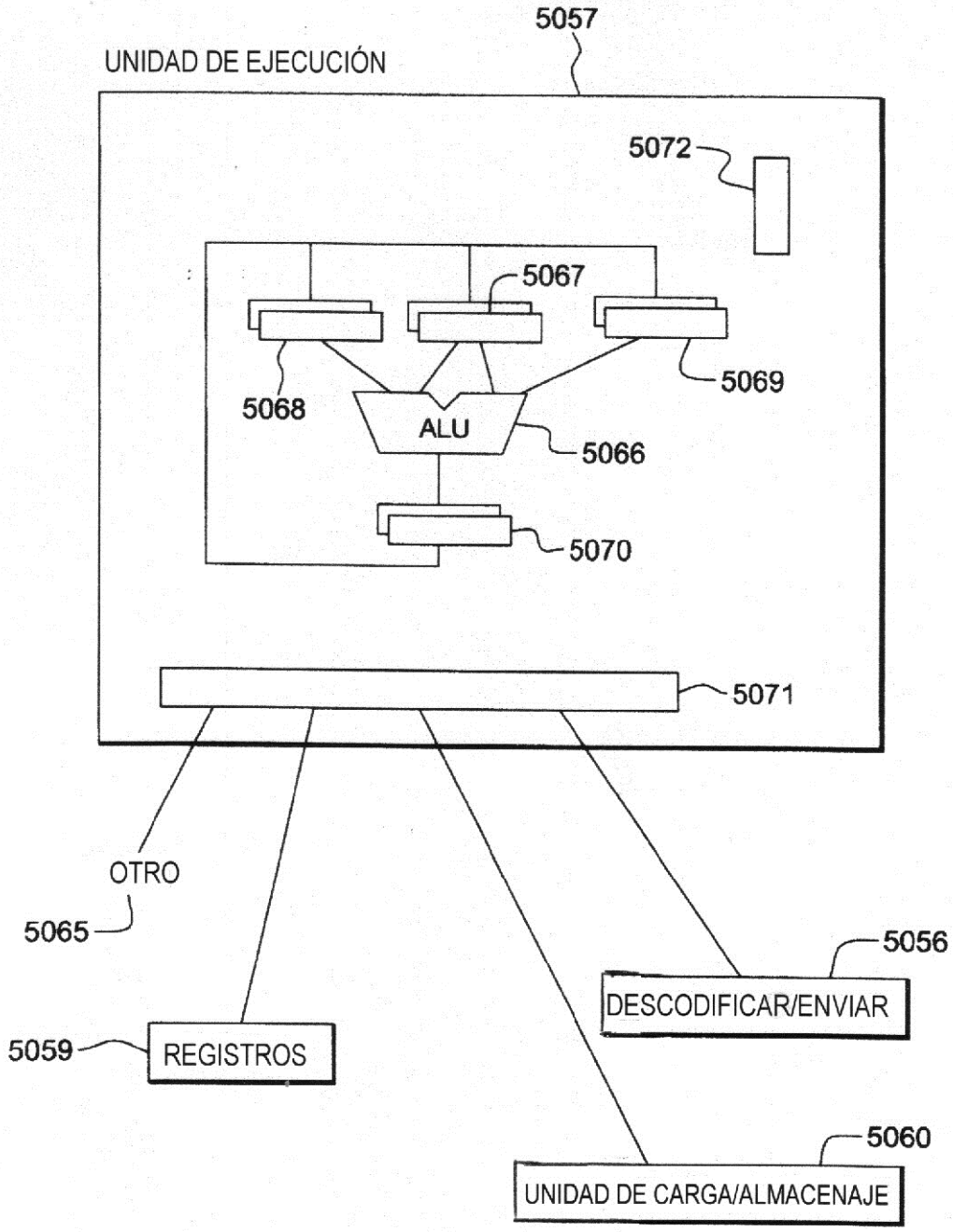

5057
UNIDAD DE EJECUCIÓN
5072
5067
5068
5069
ALU
5066
5070
5071
OTRO
5065
5056
DESCODIFICAR/ENVIAR
5059
REGISTROS
5060
UNIDAD DE CARGA/ALMACENAJE

FIG. 12A

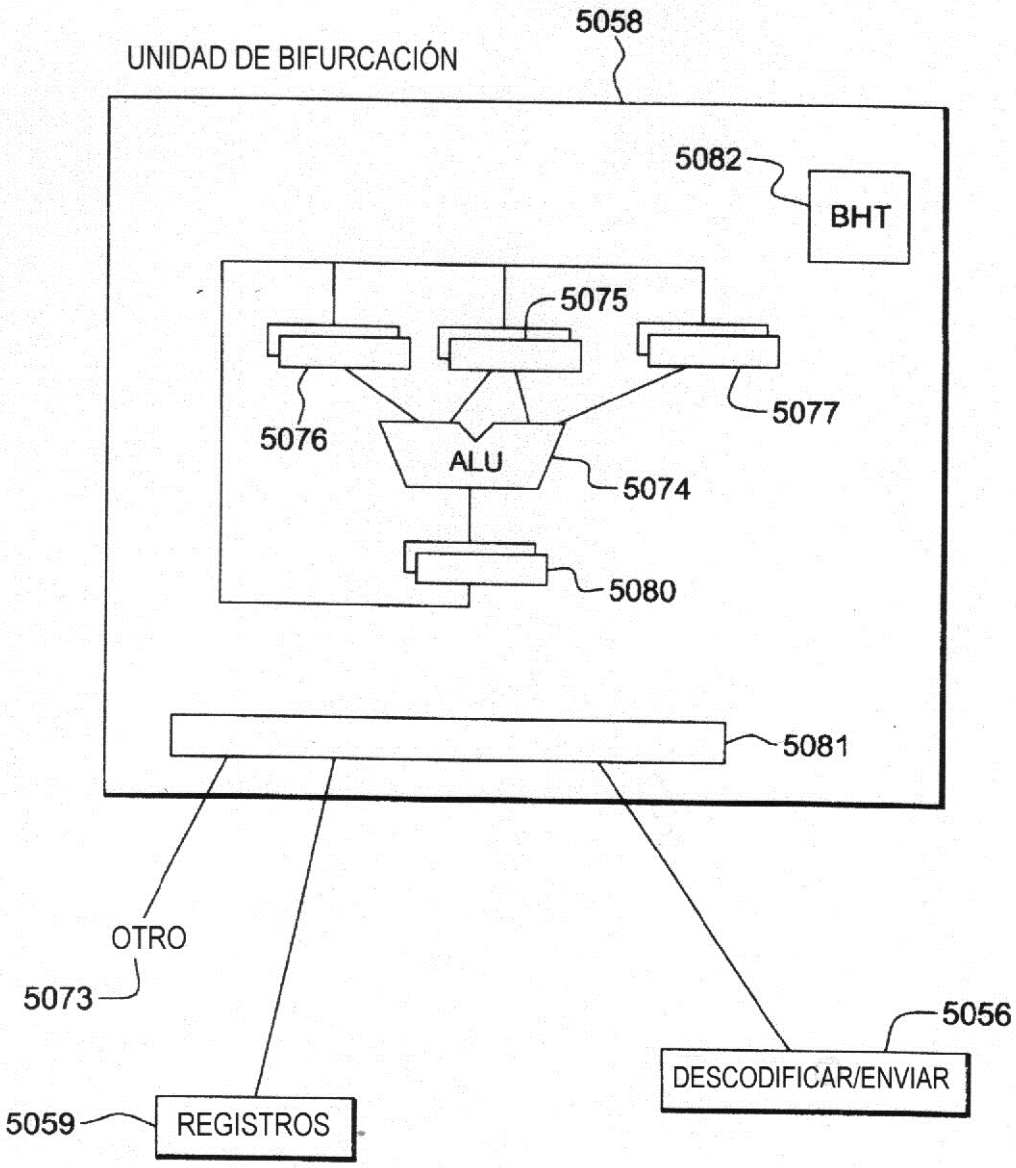

UNIDAD DE BIFURCACIÓN
5058
5082
BHT
5075
5076
5077
ALU
5074
5080
5081
OTRO
5073
5056
DESCODIFICAR/ENVIAR
5059
REGISTROS

FIG. 12B

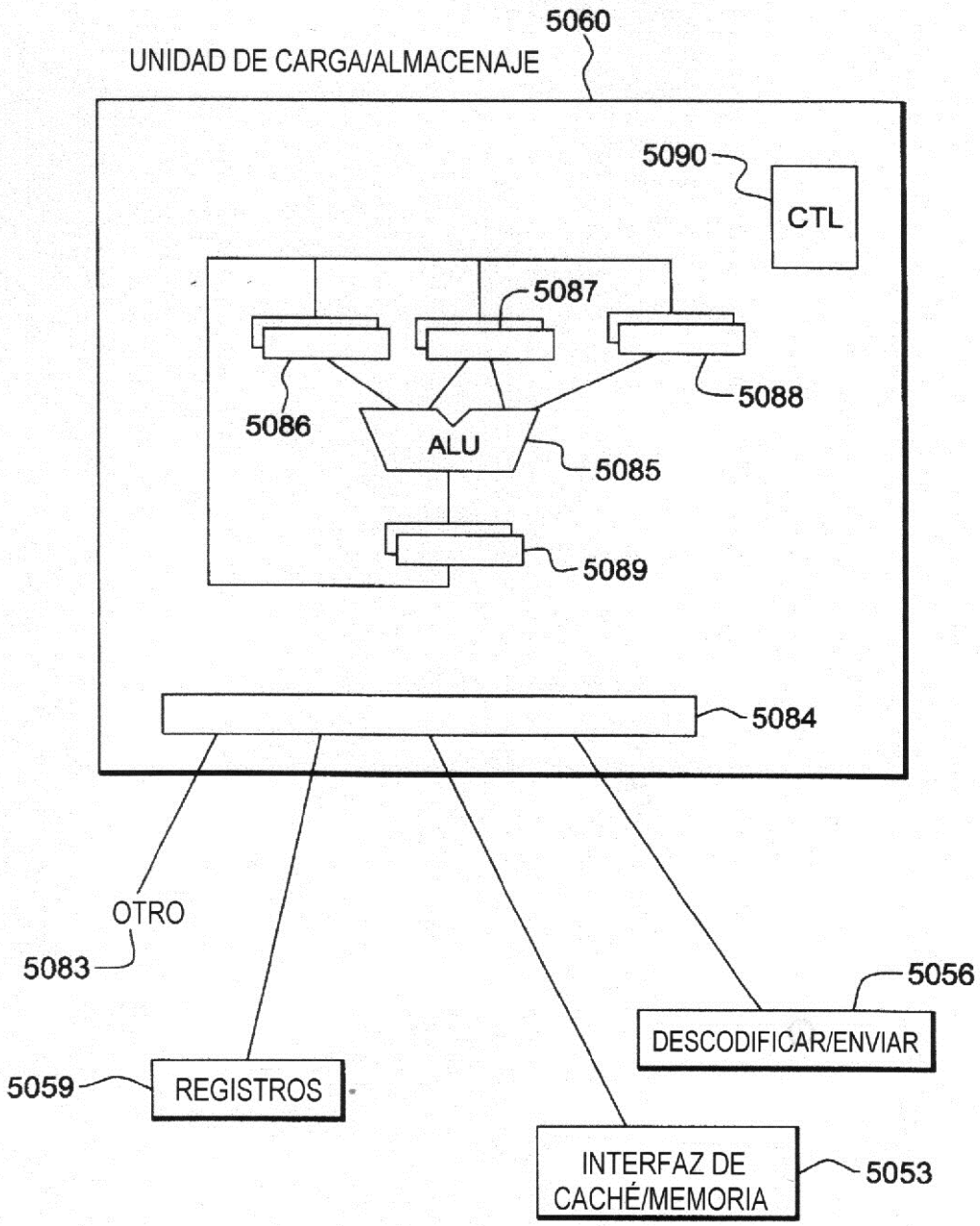

5060
UNIDAD DE CARGA/ALMACENAJE
5090
CTL
5087
5086
ALU
5088
5085
5089
5084
OTRO
5083
5056
DESCODIFICAR/ENVIAR
5059
REGISTROS
INTERFAZ DE
CACHÉ/MEMORIA
5053

FIG. 12C

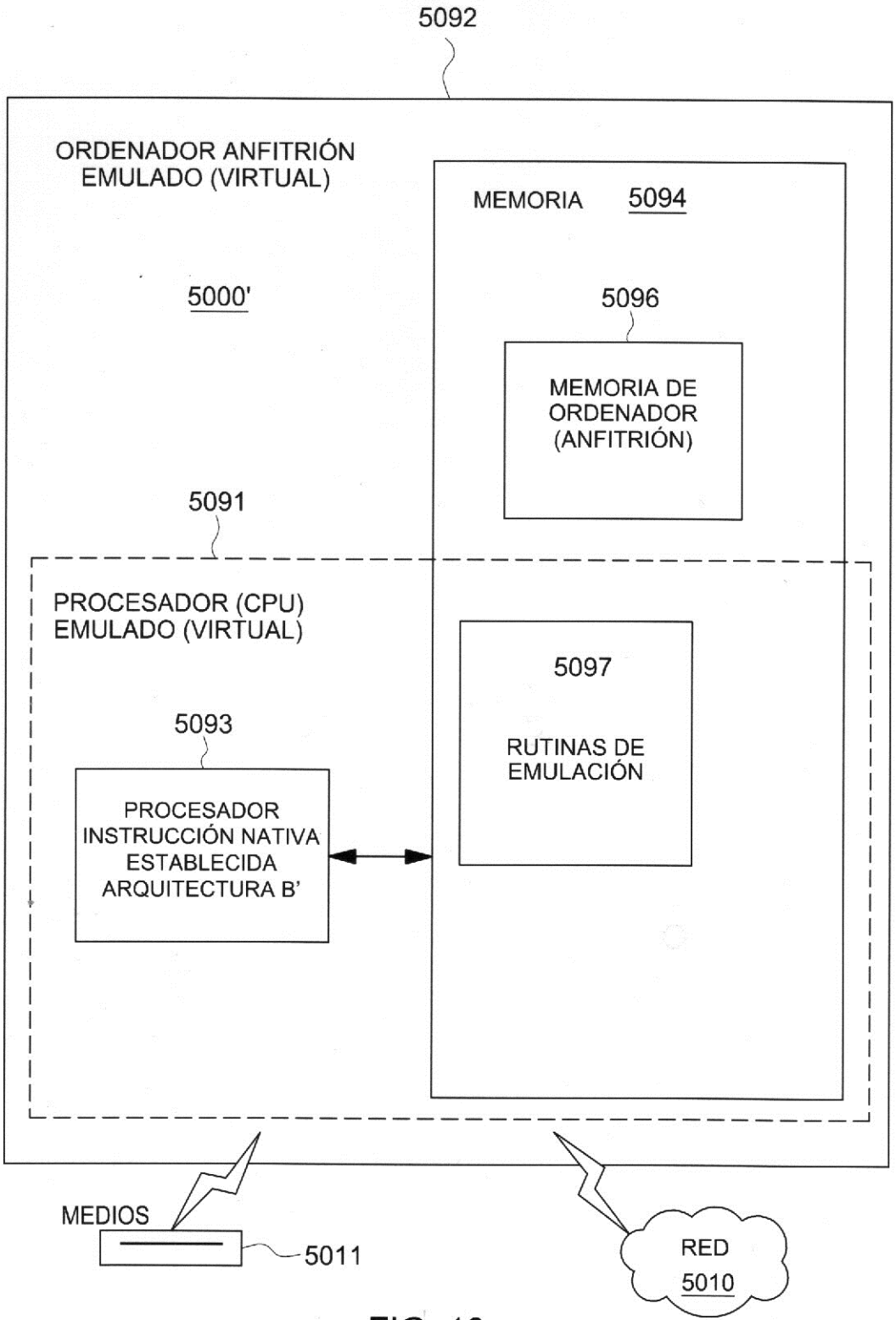

5092
ORDENADOR ANFITRIÓN EMULADO (VIRTUAL)
MEMORIA 5094
5000'
5096
MEMORIA DE ORDENADOR (ANFITRIÓN)
5091
PROCESADOR (CPU) EMULADO (VIRTUAL)
5097
RUTINAS DE EMULACIÓN
5093
PROCESADOR INSTRUCCIÓN NATIVA ESTABLECIDA ARQUITECTURA B'
MEDIOS
5011
RED
5010

FIG. 13