(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 997 191**

(51) Int. Cl.:

G06F 9/30 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **08.10.2019 E 21217772 (9)**

(97) Fecha y número de publicación de la concesión europea: **04.09.2024 EP 4002105**

(54) Título: **Sistemas y métodos para ejecutar instrucciones de producto escalar de matrices de punto flotante de 16 bits**

(30) Prioridad:

09.11.2018 US 201816186387

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
14.02.2025

(73) Titular/es:

INTEL CORPORATION (100.00%)
2200 Mission College Boulevard
Santa Clara, CA 95054, US

(72) Inventor/es:

HEINECKE, ALEXANDER F.;
VALENTINE, ROBERT;
CHARNEY, MARK J.;
SADE, RAANAN;
ADELMAN, MENACHEM;
SPERBER, ZEEV;
GRADSTEIN, AMIT y
RUBANOVICH, SIMON

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Sistemas y métodos para ejecutar instrucciones de producto escalar de matrices de punto flotante de 16 bits

CAMPO DE LA INVENCIÓN

El campo de la invención se refiere en general a la arquitectura de procesadores de computadoras y, más específicamente, con sistemas y métodos para ejecutar instrucciones de producto escalar de matrices de punto flotante de 16 bits.

TRASFONDO

Las matrices son cada vez más importantes en muchas tareas informáticas, tales como el aprendizaje automático y otros procesamientos de datos masivos. El aprendizaje profundo es una clase de algoritmos de aprendizaje automático. Las arquitecturas de aprendizaje profundo, como las redes neuronales profundas, se han aplicado en campos que incluyen visión por ordenador, reconocimiento de voz, procesamiento del lenguaje natural, reconocimiento de audio, filtrado de redes sociales, traducción automática, bioinformática y diseño de fármacos.

La inferencia y el entrenamiento, dos herramientas usadas para el aprendizaje profundo, tienden hacia una aritmética de baja precisión. Maximizar el rendimiento de los algoritmos y cálculos de aprendizaje profundo puede ayudar a satisfacer las necesidades de los procesadores de aprendizaje profundo, por ejemplo, aquellos que realizan un aprendizaje profundo en un centro de datos.

La multiplicación matriz-matriz (también conocida como GEMM, por sus siglas en inglés, o multiplicación general de matrices) es una operación de cálculo intensivo común en los procesadores modernos. El hardware especial para la multiplicación de matrices (por ejemplo, GEMM) es una buena opción para mejorar el pico de cálculo (y la eficiencia energética) de ciertas aplicaciones, como el aprendizaje profundo.

Algunas de estas aplicaciones, incluido el aprendizaje profundo, pueden operar en elementos de datos de entrada con relativamente pocos bits sin perder precisión, siempre que los elementos de salida tengan suficientes bits (es decir, más que las entradas).

El documento US 2018/321938 A1 se refiere a la aceleración de una operación de multiplicación y acumulación de matrices (MMA). Por ejemplo, un procesador incluye una ruta de datos configurada para ejecutar la operación MMA para generar una pluralidad de elementos de una matriz de resultados en una salida de la ruta de datos. Cada elemento de la matriz de resultados se genera calculando al menos un producto escalar de pares correspondientes de vectores asociados con operandos de matriz especificados en una instrucción para la operación MMA. Una operación de producto escalar incluye los pasos de: generar una pluralidad de productos parciales multiplicando cada elemento de un primer vector con un elemento correspondiente de un segundo vector; alinear la pluralidad de productos parciales basándose en los exponentes asociados con cada elemento del primer vector y cada elemento del segundo vector; y acumular la pluralidad de productos parciales alineados en una cola de resultados utilizando al menos un sumador.

SUMARIO

La presente invención está definida en las reivindicaciones independientes. Las reivindicaciones dependientes definen realizaciones de las mismas.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

La presente invención se ilustra a modo de ejemplo y sin limitación en las figuras de los dibujos adjuntos, en los que referencias similares indican elementos similares y en los que:

La **Figura 1A** ilustra una realización de teselas configuradas;

La **Figura 1B** ilustra una realización de teselas configuradas;

La **Figura 2** ilustra varios ejemplos de almacenamiento de matrices;

La **Figura 3** ilustra una realización de un sistema que utiliza un acelerador de operaciones de matrices (tesela);

Las **Figuras 4 y 5** muestran diferentes realizaciones de cómo se comparte la memoria usando un acelerador de operaciones de matrices;

La **Figura** 6 ilustra una realización de la operación de multiplicación y acumulación de matrices utilizando teselas ("TMMA");

La **Figura 7** ilustra una realización de un subconjunto de la ejecución de una iteración de una instrucción de multiplicación acumulación fusionada en cadena;

La **Figura 8** ilustra una realización de un subconjunto de la ejecución de una iteración de una instrucción de multiplicación acumulación fusionada en cadena;

La **Figura 9** ilustra una realización de un subconjunto de la ejecución de una iteración de una instrucción de multiplicación acumulación fusionada en cadena;

La **Figura 10** ilustra una realización de un subconjunto de la ejecución de una iteración de instrucción de multiplicación acumulación fusionada en cadena;

La **Figura 11** ilustra implementaciones de SIMD con tamaño de potencia de dos en las que los acumuladores usan tamaños de entrada que son mayores que las entradas a los multiplicadores de acuerdo con una realización;

La **Figura 12** ilustra una realización de un sistema que utiliza una circuitería de operaciones de matriz;

La **Figura 13** ilustra una realización de una canalización central de procesador que soporta operaciones de matrices usando teselas;

La **Figura 14** ilustra una realización de una canalización central de procesador que soporta operaciones de matrices usando teselas;

La **Figura 15** ilustra un ejemplo de una matriz expresada en formato de fila principal y formato de columna principal;

La **Figura 16** ilustra un ejemplo de uso de matrices (teselas);

La **Figura 17** ilustra una realización de un método de uso de matrices (teselas);

La **Figura 18** ilustra el soporte para la configuración del uso de teselas de acuerdo con una realización;

La **Figura 19** ilustra una realización de una descripción de las matrices (teselas) que se van a soportar;

Las **Figuras 20(A)-(D)** ilustrar ejemplos de registro o registros;

La **Figura 21** es un diagrama de bloques que ilustra el uso de una instrucción TILE16BDP para acelerar la multiplicación de matrices, de acuerdo con algunas realizaciones;

La **Figura 22A** es un pseudocódigo que ilustra la ejecución de una instrucción TILE16BDP de acuerdo con algunas realizaciones;

La **Figura 22B** es un pseudocódigo que ilustra la ejecución de una instrucción TILE16BDP de acuerdo con algunas realizaciones;

La **Figura 22C** es un pseudocódigo que ilustra funciones auxiliares para su uso por el pseudocódigo de las **Figuras 22A y 22B**, de acuerdo con algunas realizaciones;

La **Figura 23** ilustra una realización de un procesador que ejecuta un flujo para procesar una instrucción TILE16BDP;

La **Figura 24** es un diagrama de bloques que ilustra un formato de una instrucción TILE16BDP de acuerdo con algunas realizaciones;

Las **Figuras 25A-25B** son diagramas de bloques que ilustran un formato de instrucción compatible con vectores genérico y plantillas de instrucciones del mismo de acuerdo con realizaciones;

La **Figura 25A** es un diagrama de bloques que ilustra un formato de instrucción compatible con vectores genérico y plantillas de instrucciones de clase A del mismo de acuerdo con realizaciones;

La **Figura 25B** es un diagrama de bloques que ilustra el formato de instrucción compatible con vectores genérico y las plantillas de instrucciones de clase B del mismo de acuerdo con las realizaciones;

La **Figura 26A** es un diagrama de bloques que ilustra un formato de instrucción compatible con vectores específico ejemplar de acuerdo con realizaciones;

La **Figura 26B** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico que conforman el campo opcode completo de acuerdo con una realización;

La **Figura 26C** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico que conforman el campo de índice de registro de acuerdo con una realización;

La **Figura 26D** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico que conforman el campo de operación de aumento de acuerdo con una realización;

La **Figura 27** es un diagrama de bloques de una arquitectura de registros de acuerdo con una realización;

La **Figura 28A** es un diagrama de bloques que ilustra tanto un canal en orden ejemplar al igual que un canal de ejecución/emisión fuera de orden de cambio de nombre de registro ejemplar de acuerdo con realizaciones;

La **Figura 28B** es un diagrama de bloques que ilustra tanto una realización ejemplar de un núcleo de arquitectura en orden al igual que un núcleo de arquitectura ejemplar de ejecución/emisión fuera de orden de cambio de nombre de registro que se incluirá en un procesador de acuerdo con realizaciones;

Las **Figuras 29A-B** ilustran un diagrama de bloques de una arquitectura de núcleo en orden ejemplar más específica, cuyo núcleo sería uno de varios bloques lógicos (incluidos otros núcleos del mismo tipo y/o tipos diferentes) en un chip;

La **Figura 29A** es un diagrama de bloques de un solo núcleo de procesador, junto con su conexión a la red de interconexión en chip y con su subconjunto local de la caché de Nivel 2 (L2), de acuerdo con las realizaciones;

La **Figura 29B** es una vista ampliada de parte del núcleo de procesador en la **Figura 29A** de acuerdo con las realizaciones;

La **Figura 30** es un diagrama de bloques de un procesador que puede tener más de un núcleo, puede tener un controlador de memoria integrado y puede tener gráficos integrados de acuerdo con realizaciones;

Las **Figuras 31-34** son diagramas de bloques de arquitecturas informáticas ilustrativas;

La **Figura 31** muestra un diagrama de bloques de un sistema de acuerdo con una realización de la presente invención;

La **Figura 32** es un diagrama de bloques de un primer sistema ilustrativo más específico de acuerdo con una realización de la presente invención;

La **Figura 33** es un diagrama de bloques de un segundo sistema ilustrativo más específico de acuerdo con una realización de la presente invención;

La **Figura 34** es un diagrama de bloques de un sistema en un chip (SoC) de acuerdo con una realización de la presente invención; y

La **Figura 35** es un diagrama de bloques que contrasta el uso de un convertidor de instrucciones de software para convertir instrucciones binarias en un conjunto de instrucciones de origen a instrucciones binarias en un conjunto de instrucciones de destino de acuerdo con realizaciones.

DESCRIPCIÓN DETALLADA

En la siguiente descripción, se exponen numerosos detalles específicos. Sin embargo, se entiende que las realizaciones pueden practicarse sin estos detalles específicos. En otros casos, no se han mostrado en detalle circuitos, estructuras y técnicas bien conocidos para no complicar la comprensión de esta descripción.

Las referencias en la memoria descriptiva a "una realización", "una realización ilustrativa", etc., indican que la realización descrita puede incluir un rasgo, estructura o característica particular, pero cada realización puede no necesariamente incluir el rasgo, estructura, o característica particular. Además, tales expresiones no se refieren necesariamente a la misma realización. Además, cuando se describe un rasgo, estructura o característica particular en relación con una realización, se sugiere que está dentro del conocimiento de un experto en la materia afectar dicho rasgo, estructura o característica en relación con otras realizaciones, explícitamente descritas o no.

En muchos procesadores convencionales, el manejo de matrices es una tarea difícil y/o intensiva en cuanto a instrucciones. Por ejemplo, las filas de una matriz podrían colocarse en una pluralidad de registros de datos empaquetados (por ejemplo, SIMD o vectoriales) y, a continuación, operarse individualmente. Por ejemplo, sumar dos matrices de 8x2 puede requerir una carga o reunirlas en cuatro registros de datos empaquetados dependiendo de los tamaños de los datos. A continuación, se realiza una primera suma de registros de datos empaquetados correspondientes a una primera fila de cada matriz y se realiza una segunda suma de registros de datos empaquetados correspondientes a una segunda fila de cada matriz. A continuación, los registros de datos empaquetados resultantes se devuelven a la memoria. Si bien para matrices pequeñas esta situación puede ser aceptable, a menudo no lo es con matrices más grandes.

DISCUSIÓN

En el presente documento se describen mecanismos para soportar operaciones de matrices en hardware informático tales como unidades centrales de procesamiento (CPU), unidades de procesamiento gráfico (GPU) y aceleradores. Las operaciones de matrices utilizan estructuras de datos bidimensionales (2-D) que representan una o más regiones empaquetadas de memoria tales como registros. A lo largo de esta descripción, estas estructuras de datos 2-D se denominan teselas. Obsérvese que, una matriz puede ser más pequeña que una tesela (usar menos que toda una tesela) o utilizar una pluralidad de teselas (la matriz es más grande que el tamaño de cualquier tesela). A lo largo de la descripción, se utiliza un lenguaje matricial (de teselas) para indicar operaciones realizadas utilizando teselas que impactan una matriz; por lo general, no es relevante si esa matriz es más grande que cualquiera de las teselas o no.

Cada tesela puede ser objeto de diferentes operaciones, como las que se detallan en el presente documento e incluyen, entre otras: multiplicación de matrices (tesela), suma de teselas, resta de teselas, diagonal de tesela, cero de tesela, transformación de tesela, producto escalar de tesela, difusión de teselas, difusión de filas de teselas, difusión de columnas de teselas, multiplicación de teselas, multiplicación y acumulación de teselas, movimiento de teselas, etc. Además, se puede utilizar soporte para operadores tales como el uso de una escala y/o sesgo con estas operaciones o en soporte de aplicaciones no numéricas en el futuro, por ejemplo, "memoria local" de OpenCL, compresión/descompresión de datos, etc. También se describen en el presente documento instrucciones para realizar instrucciones de producto escalar de teselas de 16 bits (TILE16BDP) de matrices (teselas).

Las porciones de almacenamiento (tales como la memoria (no volátil y volátil), registros, caché, etc.) están dispuestas en teselas de diferentes dimensiones horizontales y verticales. Por ejemplo, una tesela puede tener una dimensión horizontal de 4 (por ejemplo, cuatro filas de una matriz) y una dimensión vertical de 8 (por ejemplo, 8 columnas de la matriz). Típicamente, la dimensión horizontal está relacionada con los tamaños de los elementos (por ejemplo, 2, 4, 8, 16, 32, 64, 128 bits, etc.). Se pueden soportar múltiples tipos de datos (punto flotante de precisión simple, punto flotante de precisión doble, entero, etc.).

USO EJEMPLAR DE TESELAS CONFIGURADAS

En algunas realizaciones, se pueden configurar los parámetros de teselas. Por ejemplo, una tesela dada se puede configurar para proporcionar opciones de tesela. Las opciones de teselas ilustrativas incluyen, pero sin limitación: un número de filas de la tesela, un número de columnas de la tesela, si la tesela es VÁLIDA y si la tesela consiste en un PAR de teselas con el mismo tamaño.

La **Figura 1A** ilustra una realización de teselas configuradas. Como se muestra, 4 kB de la memoria de aplicación 102 tienen almacenados en ella 4 teselas de 1 kB, tesela t0 104, tesela t1 106, tesela t2 108 y tesela t3 110. En este ejemplo, las 4 teselas no consisten en pares y cada una tiene elementos dispuestos en filas y columnas. Las teselas t0 104 y t1 106 tienen K filas y N columnas de elementos de 4 bytes (por ejemplo, datos de precisión sencilla), donde K es igual a 8 y N=32. Las teselas t2 108 y t3 110 tienen K filas y N/2 columnas de elementos de 8 bytes (por ejemplo, datos de precisión doble). Como los operandos de doble precisión tienen el doble de ancho que los de precisión sencilla, esta configuración es consistente con una paleta, usada para proporcionar opciones de teselas, que proporciona al menos 4 nombres con un almacenamiento total de al menos 4 kB. En operación, las teselas se pueden cargar desde y almacenar en la memoria usando operaciones de carga y almacenamiento. Dependiendo del esquema de codificación de instrucciones usado, varía la cantidad de memoria de aplicación disponible, así como el tamaño, número y configuración de las teselas disponibles.

La **Figura 1B** ilustra una realización de teselas configuradas. Como se muestra, 4 kB de memoria de aplicación 122 tienen almacenados en los mismos 2 pares de teselas de 1 kB, siendo el primer par la tesela t4L 124 y la tesela t4R 126, y siendo el segundo par la tesela t5L 128 y la tesela t5R 130. Como se muestra, los pares de teselas se dividen en una tesela izquierda y una tesela derecha. En otras realizaciones, el par de teselas se divide en una tesela par y una tesela impar. En este ejemplo, cada una de las 4 teselas tiene elementos dispuestos en filas y columnas. Las teselas t4L 124 y t4R 126 tienen K filas y N columnas de elementos de 4 bytes (por ejemplo, datos de punto flotante de precisión simple), donde K es igual a 8 y N es igual a 32. Las teselas t5L 128 y t5R 130 tienen K filas y N/2 columnas de elementos de 8 bytes (por ejemplo, datos de punto flotante de doble precisión). Como los operandos de doble precisión tienen el doble de de ancho que los de precisión sencilla, esta configuración es consistente con una paleta, usada para proporcionar opciones de teselas, que proporciona al menos 2 nombres con un almacenamiento total de

al menos 4 kB. Las cuatro teselas de la **Figura 1A** utilizan 4 nombres, cada uno de los cuales nombra una tesela de 1 kB, mientras que los 2 pares de teselas de la **Figura 18** pueden utilizar 2 nombres para especificar las teselas emparejadas. En algunas realizaciones, las instrucciones de tesela aceptan el nombre de una tesela emparejada como operando. En operación, las teselas se pueden cargar desde y almacenar en la memoria usando operaciones de carga y almacenamiento. Dependiendo del esquema de codificación de instrucciones usado, varía la cantidad de memoria de aplicación disponible, así como el tamaño, número y configuración de las teselas disponibles.

En algunas realizaciones, los parámetros de las teselas son definibles. Por ejemplo, se usa una "paleta" para proporcionar opciones de teselas. Las opciones ilustrativas incluyen, pero sin limitación: el número de nombres de teselas, el número de bytes en una fila de almacenamiento, el número de filas y columnas en una tesela, etc. Por ejemplo, una "altura" máxima (número de filas) de una tesela se puede definir como:

Máximo de filas de teselas = almacenamiento diseñado / (el número de nombres de paletas * el número de bytes por fila).

Como tal, una aplicación se puede escribir de manera que un uso fijo de nombres pueda aprovechar diferentes tamaños de almacenamiento en todas las implementaciones.

La configuración de las teselas se realiza mediante una instrucción de configuración de matriz (tesela) ("TILECONFIG"), donde se define un uso particular de la tesela en una paleta seleccionada. Esta declaración incluye el número de nombres de teselas que se van a usar, el número solicitado de filas y columnas por nombre (tesela) y, en algunas realizaciones, el tipo de datos solicitado de cada tesela. En algunas realizaciones, se realizan comprobaciones de consistencia durante la ejecución de una instrucción TILECONFIG para determinar que coincide con las restricciones de la entrada de la paleta.

TIPOS EJEMPLARES DE ALMACENAMIENTO DE TESELAS

La **Figura 2** ilustra varios ejemplos de almacenamiento de matrices. En (A), una tesela se almacena en la memoria. Como se muestra, cada "fila" consiste en cuatro elementos de datos empaquetados. Para pasar a la siguiente "fila", se usa un valor de paso. Obsérvese que las filas pueden almacenarse consecutivamente en la memoria. Los accesos a la memoria con paso permiten el acceso de una fila a la siguiente cuando el almacenamiento de teselas no mapea el ancho de fila de la matriz de memoria subyacente.

Las cargas de teselas desde la memoria y los almacenes en la memoria típicamente son accesos con pasos desde la memoria de la aplicación a filas empaquetadas de datos. Las instrucciones TILELOAD y TILESTORE de ejemplo, u otras referencias de instrucciones a la memoria de la aplicación como un operando TILE en instrucciones de carga-operación, son, en algunas realizaciones, reiniciables para manejar (hasta) 2*filas de fallas de página, excepciones de punto flotante sin máscara y/o interrupciones por instrucción.

En (B), una matriz se almacena en una tesela compuesta por una pluralidad de registros, tales como registros de datos empaquetados (datos múltiples de instrucción única (SIMD) o registros vectoriales). En este ejemplo, la tesela se superpone en tres registros físicos. Típicamente, se usan registros consecutivos, aunque no tiene por qué ser así.

En (C), una matriz se almacena en un mosaico en un almacenamiento sin registro accesible a un circuito de acumulación múltiple con fusibles (FMA) utilizado en las operaciones de mosaico. Este almacenamiento puede estar dentro de un FMA o adyacente a él. Además, en algunas realizaciones, que se analizan a continuación, el almacenamiento puede ser para un elemento de datos y no para una fila o tesela completa.

Los parámetros admitidos para la arquitectura TMMA se informan a través de CPUID. En algunas realizaciones, la lista de información incluye una altura máxima y una dimensión de SIMD máxima. Configurar la arquitectura de TMMA requiere especificar las dimensiones de cada tesela, el tamaño de elemento para cada tesela y el identificador de paleta. Esta configuración se realiza ejecutando la instrucción TILECONFIG.

La ejecución con éxito de una instrucción TILECONFIG habilita a los operadores TILE posteriores. Una instrucción TILERELEASEALL borra la configuración de la tesela y deshabilita las operaciones TILE (hasta que se ejecute la siguiente instrucción TILECONFIG). En algunas realizaciones, XSAVE, XSTORE, etc. se usan en la conmutación de contexto usando teselas. En algunas realizaciones, se utilizan 2 bits XCR0 en XSAVE, uno para metadatos TILECONFIG y un bit correspondiente a los datos de carga útil de la tesela real.

TILECONFIG no únicamente configura el uso de teselas, sino que también establece una variable de estado que indica que el programa se encuentra en una región de código con teselas configuradas. Una implementación puede enumerar restricciones sobre otras instrucciones que se pueden usar con una región de tesela, tal como no usar un conjunto de registros existente, etc.

Típicamente, salir de una región de tesela se realiza con la instrucción TILERELEASEALL. No requiere parámetros e invalida rápidamente todas las teselas (lo que indica que los datos ya no necesitan guardarse ni restaurarse) y borra el estado interno correspondiente a estar en una región de tesela.

En algunas realizaciones, las operaciones de tesela pondrán a cero cualquier fila y columna más allá de las dimensiones especificadas por la configuración de tesela. Por ejemplo, las operaciones de tesela pondrán a cero los datos más allá del número configurado de columnas (teniendo en cuenta el tamaño de los elementos) a medida que se escribe cada fila. Por ejemplo, con filas de 64 bytes y un mosaico configurado con 10 filas y 12 columnas, una operación que escriba elementos FP32 escribiría cada una de las primeras 10 filas con 12 * 4 bytes con datos de salida/resultado y pondría a cero los 4 * 4 bytes restantes en cada fila. Las operaciones en tesela también ponen a cero por completo cualquier fila después de las primeras 10 filas configuradas. Cuando se utiliza mosaico de 1K con filas de 64 bytes, habrá 16 filas, por lo que en este ejemplo, las últimas 6 filas también se pondrán a cero.

En algunas realizaciones, una instrucción de restauración de contexto (por ejemplo, XRSTOR), al cargar datos, impone que los datos más allá de las filas configuradas para una tesela se mantendrán como cero. Si no hay una configuración válida, todas las filas se ponen a cero. XRSTOR de datos de teselas puede cargar basura en las columnas más allá de las configuradas. No debería ser posible que XRSTOR borre más allá del número de columnas configuradas porque no hay un ancho de elemento asociado con la configuración de la tesela.

El guardado de contexto (por ejemplo, XSAVE) expone toda el área de almacenamiento TILE al escribirla en la memoria. Si XRSTOR cargó datos basura en la parte más a la derecha de una tesela, XSAVE guardará esos datos. XSAVE pondrá ceros en las filas más allá del número especificado para cada tesela.

En algunas realizaciones, las instrucciones de las teselas se pueden reiniciar. Las operaciones que acceden a la memoria permiten reiniciar después de fallos de página. Las instrucciones computacionales que tratan con operaciones de punto flotante también permiten excepciones de punto flotante sin máscara, con el enmascaramiento de las excepciones controladas por un registro de control y/o de estado.

Para soportar instrucciones de reinicio después de estos eventos, las instrucciones almacenan información en los registros de inicio que se detallan a continuación.

SISTEMAS DE OPERACIÓN DE MATRICES (TESELAS)

SOPORTE DE HARDWARE EJEMPLAR

La **Figura 3** ilustra una realización de un sistema que utiliza un acelerador de operaciones de matriz (mosaico). En esta ilustración, un procesador anfitrión/sistema de procesamiento 301 comunica comandos 311 (por ejemplo, operaciones de manipulación de matrices tales como operaciones aritméticas o de manipulación de matrices, u operaciones de carga y almacenamiento) a un acelerador de operaciones de matrices 307. Sin embargo, esto se muestra de esta manera solo por motivos de análisis. Como se detalla más adelante, este acelerador 307 puede ser parte de un núcleo de procesamiento. Típicamente, los comandos 311 que son instrucciones del operador de manipulación de teselas se referirán a las teselas como formato registro-registro ("reg-reg") o registro-memoria ("reg-mem"). Otros comandos tales como TILESTORE, TILELOAD, TILECONFIG, etc., no realizan operaciones de datos en una tesela. Los comandos pueden ser instrucciones decodificadas (por ejemplo, microoperaciones) o macroinstrucciones para que las maneje el acelerador 307.

En este ejemplo, una interfaz de memoria coherente 303 está acoplada al procesador/sistema de procesamiento anfitrión 301 y al acelerador de operaciones de matrices 307 de manera que puedan compartir memoria. Las **Figuras 4 y 5** muestran diferentes realizaciones de cómo se comparte la memoria usando un acelerador de operaciones de matrices. Como se muestra en la **Figura 4,** el procesador de anfitrión 401 y el circuito de acelerador de operaciones de matrices 405 comparten la misma memoria 403. **Figura 5** ilustra una realización en la que el procesador principal 501 y el acelerador de operaciones de matrices 505 no comparten memoria, pero pueden acceder a la memoria de cada uno. Por ejemplo, el procesador 501 puede acceder a la memoria de tesela 507 y utilizar su memoria de anfitrión 503 de forma normal. De manera similar, el acelerador de operaciones de matrices 505 puede acceder a la memoria anfitrión 503, pero más típicamente usa su propia memoria 507. Obsérvese que estas memorias pueden ser de diferentes tipos.

En algunas realizaciones, las teselas se soportan mediante una superposición sobre registros físicos. Por ejemplo, una tesela puede utilizar 16 registros de 1.024 bits, 32 registros de 512 bits, etc., dependiendo de la implementación. En algunas realizaciones, las operaciones de matrices utilizan estructuras de datos bidimensionales (2-D) que representan una o más regiones empaquetadas de memoria, tales como registros. A lo largo de esta descripción, estas estructuras de datos 2-D se denominan teselas o registros de teselas.

En algunas realizaciones, el acelerador de operaciones de matrices 307 incluye una pluralidad de FMA 309 acoplados a memorias intermedias de datos 305 (en algunas implementaciones, uno o más de estas memorias intermedias 305 se almacenan en los FMA de la cuadrícula como se muestra). Las memorias intermedias de datos 305 que almacenan

de forma intermedia teselas cargadas desde la memoria y/o teselas que se van a almacenar en la memoria (por ejemplo, usando una instrucción de tileload o de tilestore). Las memorias intermedias de datos pueden ser, por ejemplo, una pluralidad de registros. Típicamente, estas FMA están dispuestas como una cuadrícula de FMA 309 en cadena que pueden leer y escribir teselas. En este ejemplo, el acelerador de operaciones de matrices 307 ha de realizar una operación de multiplicación matricial usando las teselas T0, T1 y T2. Al menos una de las teselas está alojada en la cuadrícula de FMA 309. En algunas realizaciones, todas las teselas en una operación se almacenan en la cuadrícula de FMA 309. En otras realizaciones, únicamente un subconjunto se almacena en la cuadrícula de FMA 309. Como se muestra, T1 está alojada y T0 y T2 no lo están. Obsérvese que A, B y C se refieren a las matrices de estas teselas que pueden o no ocupar todo el espacio de la tesela.

La **Figura 6** ilustra una realización de la operación de multiplicación acumulación de matrices usando teselas ("TMMA").

El número de filas en la matriz (TILE A 601) coincide con el número de FMA en serie (en cadena) que comprenden la latencia del cálculo. Una implementación es libre de recircular sobre una cuadrícula de menor altura, pero el cálculo sigue siendo el mismo.

El vector de origen/destino proviene de una tesela de N filas (TESELA C 605) y la cuadrícula de FMA 611 realiza N operaciones vector-matriz que dan como resultado una instrucción completa que realiza una multiplicación matricial de teselas. La tesela B 603 es el otro origen de vector y proporciona términos de "difusión" a las FMA en cada etapa.

En operación, en algunas realizaciones, los elementos de la matriz B (almacenados en una tesela B 603) se distribuyen a través de la cuadrícula rectangular de los FMA. La matriz B (almacenada en la tesela A 601) tiene sus elementos de una fila transformadas para que coincidan con la dimensión columnar de la cuadrícula rectangular de los FMA. En cada FMA de la cuadrícula, un elemento de A y B se multiplica y se suma al sumando entrante (desde arriba en la Figura) y la suma saliente se pasa a la siguiente fila de FMA (o la salida final).

La latencia de una única etapa es proporcional a K (altura de fila de la matriz B) y las TMMA típicamente tienen suficientes filas de origen-destino (ya sea en una única tesela o a través de teselas) para ocultar esa latencia. Una implementación también puede dividir la dimensión de SIMD (elemento de datos empaquetados) M (altura de fila de la matriz A) a través de etapas de tiempo, pero esto simplemente cambia la constante por la que se multiplica K. Cuando un programa especifica un K menor que el máximo enumerado por TMACC, una implementación es libre de implementarlo con "enmascaramiento" o "salidas anticipadas".

La latencia de una TMMA completa es proporcional a N*K. La tasa de repetición es proporcional a N. El número de las MAC por instrucción TMMA es N*K*M.

La **Figura 7** ilustra una realización de un subconjunto de la ejecución de una iteración de una instrucción de multiplicación acumulación fusionada en cadena. En particular, esto ilustra el circuito de ejecución de una iteración de la posición de un elemento de datos empaquetado del destino. En esta realización, la multiplicación acumulación fusionada en cadena opera en orígenes con signo en los que el acumulador tiene el doble del tamaño de los datos de entrada.

Un primer origen con signo (origen 1 701) y un segundo origen con signo (origen 2 703) tiene cada uno cuatro elementos de datos empaquetados. Cada uno de estos elementos de datos empaquetados almacena datos con signo, tales como datos de punto flotante. Un tercer origen con signo (origen 3 709) tiene dos elementos de datos empaquetados, cada uno de los cuales almacena datos con signo. Los tamaños del primer y segundo orígenes con signo 701 y 703 son la mitad que los del tercer origen con signo (valor inicial o resultado anterior) 709. Por ejemplo, el primer y segundo origen con signo 701 y 703 podrían tener elementos de datos empaquetados de 32 bits (por ejemplo, punto flotante de precisión simple) mientras que el tercer origen con signo 709 podría tener elementos de datos empaquetados de 64 bits (por ejemplo, punto flotante de precisión doble).

En esta ilustración, únicamente se muestran las dos posiciones de elementos de datos empaquetados más significativas del primer y segundo orígenes con signo 701 y 703 y la posición de elementos de datos empaquetados más significativa del tercer origen con signo 709. Por supuesto, también se procesarían las demás posiciones de elementos de datos empaquetados.

Como se ilustra, los elementos de datos empaquetados se procesan en pares. Por ejemplo, los datos de las posiciones de elementos de datos empaquetados más significativas del primer y segundo orígenes con signo 701 y 703 se multiplican usando un circuito de multiplicación 705, y los datos de las posiciones de los segundos elementos de datos empaquetados más significativos del primer y segundo orígenes con signo 701 y 703 se multiplican usando un circuito de multiplicación 707. En algunas realizaciones, estos circuitos multiplicadores 705 y 707 se reutilizan para otras posiciones de elementos de datos empaquetados. En otras realizaciones, se usan circuitos multiplicadores adicionales para que los elementos de datos empaquetados se procesen en paralelo. En algunos contextos, la ejecución paralela se realiza usando carriles que son del tamaño del tercer origen con signo 709. Los resultados de cada una de las multiplicaciones se suman usando circuitería de suma 711.

El resultado de la suma de los resultados de las multiplicaciones se suma a los datos de la posición de elemento de datos empaquetados más significativo del origen con signo 3 709 (usando un sumador diferente 713 o el mismo sumador 711).

Finalmente, el resultado de la segunda adición se almacena en el destino con signo 715 en una posición de elemento de datos empaquetados que corresponde a la posición de elemento de datos empaquetados usada desde el tercer origen con signo 709 o se pasa a la siguiente iteración, si la hay. En algunas realizaciones, se aplica una máscara de escritura a este almacenamiento de modo que si se establece una máscara de escritura (bit) correspondiente, se realiza el almacenamiento y, si no se establece, no se realiza el almacenamiento.

La **Figura 8** ilustra una realización de un subconjunto de la ejecución de una iteración de una instrucción de multiplicación acumulación fusionada en cadena. En particular, esto ilustra el circuito de ejecución de una iteración de la posición de un elemento de datos empaquetado del destino. En esta realización, la multiplicación acumulación fusionada en cadena opera en orígenes con signo en los que el acumulador tiene el doble del tamaño de los datos de entrada.

Un primer origen con signo (origen 1801) y un segundo origen con signo (origen 2803) tiene cada uno cuatro elementos de datos empaquetados. Cada uno de estos elementos de datos empaquetados almacena datos con signo, tales como datos de números enteros. Un tercer origen con signo (origen 3 809) tiene dos elementos de datos empaquetados, cada uno de los cuales almacena datos con signo. Los tamaños del primer y segundo orígenes con signo 801 y 803 son la mitad que los del tercer origen con signo 809. Por ejemplo, el primer y segundo origen con signo 801 y 803 podrían tener elementos de datos empaquetados de 32 bits (por ejemplo, punto flotante de precisión simple), el tercer origen con signo 809 podría tener elementos de datos empaquetados de 64 bits (por ejemplo, punto flotante de precisión doble).

En esta ilustración, únicamente se muestran las dos posiciones de elementos de datos empaquetados más significativas del primer y segundo orígenes con signo 801 y 803 y la posición de elementos de datos empaquetados más significativa del tercer origen con signo 809. Por supuesto, también se procesarían las demás posiciones de elementos de datos empaquetados.

Como se ilustra, los elementos de datos empaquetados se procesan en pares. Por ejemplo, los datos de las posiciones de elementos de datos empaquetados más significativos de los orígenes con signo primero y segundo 801 y 803 se multiplican utilizando un circuito de multiplicación 805, y los datos de las posiciones de los segundos elementos de datos empaquetados más significativos de los orígenes con signo primero y segundo 801 y 803 se multiplican utilizando un circuito de multiplicación 807. En algunas realizaciones, los circuitos multiplicadores 805 y 807 realizan las multiplicaciones con precisión infinita sin saturación y utilizan un circuito sumador/de saturación 813 para saturar los resultados de la acumulación a más o menos infinito en caso de un desbordamiento y a cero en caso de cualquier subdesbordamiento. En otras realizaciones, los circuitos multiplicadores 805 y 807 realizan la saturación por sí mismos. En algunas realizaciones, estos circuitos multiplicadores 805 y 807 se reutilizan para otras posiciones de elementos de datos empaquetados. En otras realizaciones, se usan circuitos multiplicadores adicionales para que los elementos de datos empaquetados se procesen en paralelo. En algunos contextos, la ejecución paralela se realiza usando carriles que son del tamaño del tercer origen con signo (valor inicial o resultado de la iteración anterior) 809. Los resultados de cada una de las multiplicaciones se suman al tercer origen con signo 809 utilizando el circuito de adición/saturación 813.

El circuito de suma/saturación 813 (acumulador) conserva un signo de un operando cuando la suma da como resultado un valor que es demasiado grande. En particular, la evaluación de saturación ocurre en el resultado de precisión infinita entre la adición multidireccional y la escritura en el destino o la siguiente iteración. Cuando el acumulador 813 es de punto flotante y los términos de entrada son enteros, la suma de productos y el valor de entrada del acumulador de punto flotante se convierten en valores de precisión infinita (números de punto fijo de cientos de bits), se realiza la suma de los resultados de la multiplicación y la tercera entrada, y se realiza un solo redondeo al tipo de acumulador real.

La saturación sin signo significa que los valores de salida están limitados a un número máximo sin signo para ese ancho de elemento (todos 1). La saturación con signo significa que un valor está limitado a estar en el rango entre un número negativo mínimo y un número positivo máximo para ese ancho de elemento (para bytes, por ejemplo, el rango es de -128 (= - 2^7) a 127(= 2^7-1)).

Se almacena el resultado de la adición y la comprobación de saturación en el destino con signo 815 en una posición de elemento de datos empaquetados que corresponde a la posición de elemento de datos empaquetados usada desde el tercer origen con signo 809 o se pasa a la siguiente iteración, si la hay. En algunas realizaciones, se aplica una máscara de escritura a este almacenamiento de modo que si se establece una máscara de escritura (bit) correspondiente, se realiza el almacenamiento y, si no se establece, no se realiza el almacenamiento.

La **Figura 9** ilustra una realización de un subconjunto de la ejecución de una iteración de una instrucción de multiplicación acumulación fusionada en cadena. En particular, esto ilustra el circuito de ejecución de una iteración de la posición de un elemento de datos empaquetado del destino. En esta realización, la multiplicación acumulación fusionada en cadena opera en un origen con signo y en un origen sin signo en donde el acumulador tiene 4 veces el tamaño de los datos de entrada.

Un primer origen con signo (origen 1901) y un segundo origen sin signo (origen 2 903) tiene cada uno cuatro elementos de datos empaquetados. Cada uno de estos elementos de datos empaquetados contiene datos tales como datos de punto flotante o enteros. Un tercer origen con signo (valor inicial o resultado 915) tiene un elemento de datos empaquetados del cual almacena datos con signo. Los tamaños del primer y segundo orígenes 901 y 903 son una cuarta parte de los del tercer origen con signo 915. Por ejemplo, la primera y segunda del primer y segundo origen 901 y 903 podrían tener elementos de datos empaquetados de 16 bits (por ejemplo, word) y el tercer origen firmada 915 podría tener elementos de datos empaquetados de 64 bits (por ejemplo, punto flotante de doble precisión o entero de 64 bits).

En esta ilustración, se muestran las cuatro posiciones de elementos de datos empaquetados más significativas del primer y segundo orígenes 901 y 903 y la posición de elementos de datos empaquetados más significativa del tercer origen con signo 915. Por supuesto, también se procesarían otras posiciones de elementos de datos empaquetados, si las hubiera.

Como se ilustra, los elementos de datos empaquetados se procesan en cuartetos. Por ejemplo, los datos de las posiciones de elementos de datos empaquetados más significativas del primer y segundo orígenes 901 y 903 se multiplican usando un circuito de multiplicación 905, los datos de las segundas posiciones de elementos de datos empaquetados más significativas del primer y segundo orígenes 901 y 903 se multiplican usando un circuito de multiplicación 907, los datos de las terceras posiciones de elementos de datos empaquetados más significativas del primer y segundo orígenes 901 y 903 se multiplican usando un circuito de multiplicación 909, y los datos de las posiciones de elementos de datos empaquetados menos significativas del primer y segundo orígenes 901 y 903 se multiplican usando un circuito de multiplicación 911. En algunas realizaciones, los elementos de datos empaquetados con signo del primer origen 901 tienen signo extendido y los elementos de datos empaquetados sin signo del segundo origen 903 tienen extensión cero antes de las multiplicaciones.

En algunas realizaciones, estos circuitos multiplicadores 905-911 se reutilizan para otras posiciones de elementos de datos empaquetados. En otras realizaciones, se usan circuitos multiplicadores adicionales para que los elementos de datos empaquetados se procesen en paralelo. En algunos contextos, la ejecución paralela se realiza usando carriles que son del tamaño del tercer origen con signo 915. Los resultados de cada una de las multiplicaciones se suman usando circuitería de suma 913.

El resultado de la suma de los resultados de las multiplicaciones se suma a los datos de la posición de elemento de datos empaquetados más significativo del origen con signo 3 915 (usando un sumador diferente 917 o el mismo sumador 913).

Finalmente, el resultado 919 de la segunda adición se almacena en el destino con signo en una posición de elemento de datos empaquetados que corresponde a la posición de elemento de datos empaquetados usada desde el tercer origen con signo 915 o se pasa a la siguiente iteración. En algunas realizaciones, se aplica una máscara de escritura a este almacenamiento de modo que si se establece una máscara de escritura (bit) correspondiente, se realiza el almacenamiento y, si no se establece, no se realiza el almacenamiento.

La **Figura 10** ilustra una realización de un subconjunto de la ejecución de una iteración de instrucción de multiplicación acumulación fusionada en cadena. En particular, esto ilustra el circuito de ejecución de una iteración de la posición de un elemento de datos empaquetado del destino. En esta realización, la multiplicación acumulación fusionada en cadena opera en un origen con signo y en un origen sin signo en donde el acumulador tiene 4 veces el tamaño de los datos de entrada.

Un primer origen con signo 1001 y un segundo origen no con signo 1003 tienen cada una cuatro elementos de datos empaquetados. Cada uno de estos elementos de datos empaquetados almacena datos tales como datos de punto flotante o enteros. Un tercer origen con signo 1015 (resultado inicial o anterior) tiene un elemento de datos empaquetado que almacena datos con signo. Los tamaños del primer y segundo orígenes son una cuarta parte del tercer origen con signo 1015 (resultado inicial o anterior). Por ejemplo, el primer y segundo origen podrían tener elementos de datos empaquetados de 16 bits (por ejemplo, word) y el tercer origen con signo 1015 (resultado inicial o anterior) podría tener elementos de datos empaquetados de 64 bits (por ejemplo, punto flotante de doble precisión o entero de 64 bits).

En esta ilustración, se muestran las cuatro posiciones de elementos de datos empaquetados más significativos del primer origen con signo 1001 y el segundo origen sin signo 1003 y la posición de elemento de datos empaquetados más significativo del tercer origen con signo 1015. Por supuesto, también se procesarían otras posiciones de elementos de datos empaquetados, si las hubiera.

Como se ilustra, los elementos de datos empaquetados se procesan en cuartetos. Por ejemplo, los datos de las posiciones de elementos de datos empaquetados más significativos del primer origen con signo 1001 y el segundo origen sin signo 1003 se multiplican utilizando un circuito de multiplicación 1005, los datos de las segundas posiciones de elementos de datos empaquetados más significativos del primer origen con signo 1001 y el segundo origen sin signo 1003 se multiplican utilizando un circuito de multiplicación 1007, los datos de las terceras posiciones de elementos de datos empaquetados más significativos del primer origen con signo 1001 y el segundo origen sin signo 1003 se multiplican utilizando un circuito de multiplicación 1009, y los datos de las posiciones de elementos de datos empaquetados menos significativos del primer origen con signo 1001 y el segundo origen sin signo 1003 se multiplican utilizando un circuito de multiplicación 1011. En algunas realizaciones, los elementos de datos empaquetados con signo del primer origen con signo 1001 se extienden en signo y los elementos de datos empaquetados sin signo del segundo origen sin signo 1003 se extienden en cero antes de las multiplicaciones.

En algunas realizaciones, estos circuitos multiplicadores 1005-1011 se reutilizan para otras posiciones de elementos de datos empaquetados. En otras realizaciones, se usan circuitos multiplicadores adicionales para que los elementos de datos empaquetados se procesen en paralelo. En algunos contextos, la ejecución paralela se realiza utilizando líneas que tienen el tamaño del tercer origen con signo 1015 (resultado inicial o anterior). El resultado de la suma de los resultados de las multiplicaciones se agrega a los datos de la posición de elemento de datos empaquetado más significativo del tercer origen con signo 1015 (resultado inicial o anterior) utilizando un circuito sumador/de saturación 1013.

El circuito de adición/saturación (acumulador) 1013 conserva un signo de un operando cuando la adición da como resultado un valor demasiado grande o demasiado pequeño para la saturación con signo. En particular, la evaluación de saturación ocurre en el resultado de precisión infinita entre la adición multidireccional y la escritura en el destino. Cuando el acumulador 1013 es de punto flotante y los términos de entrada son enteros, la suma de productos y el valor de entrada del acumulador de punto flotante se convierten en valores de precisión infinita (números de punto fijo de cientos de bits), se realiza la suma de los resultados de la multiplicación y la tercera entrada, y se realiza un solo redondeo al tipo de acumulador real.

El resultado 1019 de la comprobación de adición y saturación se almacena en el destino firmado en una posición de elemento de datos empaquetado que corresponde a la posición de elemento de datos empaquetado utilizado desde el tercer origen con signo 1015 (resultado inicial o anterior) o se pasa a la siguiente iteración. En algunas realizaciones, se aplica una máscara de escritura a este almacenamiento de modo que si se establece una máscara de escritura (bit) correspondiente, se realiza el almacenamiento y, si no se establece, no se realiza el almacenamiento.

La **Figura 11** ilustra implementaciones de SIMD con tamaño de potencia de dos en las que los acumuladores usan tamaños de entrada que son mayores que las entradas a los multiplicadores de acuerdo con una realización. Obsérvese que los valores origen (a los multiplicadores) y acumulador pueden ser valores con signo o sin signo. Para un acumulador que tiene tamaños de entrada 2X (en otras palabras, el valor de entrada del acumulador es el doble del tamaño de los tamaños de elementos de datos empaquetados de los orígenes), la tabla 1101 ilustra diferentes configuraciones. Para orígenes con tamaño de bytes, el acumulador usa valores de palabra o de coma flotante de precisión media (HPFP) que tienen un tamaño de 16 bits. Para orígenes con tamaño de palabra, el acumulador usa valores de números enteros de 32 bits o valores de coma flotante de precisión sencilla (SPFP) que tienen un tamaño de 32 bits. Para orígenes de SPFP o con tamaño de números entero de 32 bits, el acumulador usa valores de coma flotante de números enteros de 64 o de precisión doble (DPFP) que tienen un tamaño de 64 bits.

Para un acumulador que tiene tamaños de entrada 4X (en otras palabras, el valor de entrada del acumulador es cuatro veces el tamaño de los tamaños de elementos de datos empaquetados de los orígenes), la tabla 1103 ilustra diferentes configuraciones. Para orígenes con tamaño de bytes, el acumulador usa valores de números enteros de 32 bits o valores de coma flotante de precisión sencilla (SPFP) que tienen un tamaño de 32 bits. Para orígenes con tamaño de palabra, el acumulador usa valores de números enteros de 64 bits o de coma flotante de precisión doble (DPFP) que tienen un tamaño de 64 bits en algunas realizaciones.

Para un acumulador que tiene tamaños de entrada 8X (en otras palabras, el valor de entrada del acumulador es ocho veces el tamaño de los tamaños de elementos de datos empaquetados de los orígenes), la tabla 1105 ilustra una configuración. Para orígenes con tamaño de bytes, el acumulador usa un número entero de 64 bits.

Como se indicó anteriormente, el circuito de operaciones de matrices puede incluirse en un núcleo o como un acelerador externo. La **Figura 12** ilustra una realización de un sistema que utiliza una circuitería de operaciones de matriz. En esta ilustración, varias entidades están acopladas con una interconexión de anillo 1245.

Una pluralidad de núcleos, núcleo 0 1201, núcleo 1 1203, núcleo 2 1205 y núcleo N 1207 proporcionan soporte de instrucciones no basadas en teselas. En algunas realizaciones, el circuito de operaciones de matrices 1251 se proporciona en un núcleo 1203, y en otras realizaciones, el circuito de operaciones de matrices 1211 y 1213 es accesible en la interconexión en anillo 1245.

Adicionalmente, se proporciona uno o más controladores de memoria 1223-1225 para comunicarse con la memoria 1233 y 1231 en nombre de los núcleos y/o el circuito de operaciones de matrices.

La **Figura 13** ilustra una realización de una canalización central de procesador que soporta operaciones de matrices usando teselas. El circuito de predicción de ramas y decodificación 1303 realiza predicción de ramas de instrucciones, decodificación de instrucciones y/o ambas a partir de instrucciones almacenadas en el almacenamiento de instrucciones 1301. Por ejemplo, las instrucciones detalladas en el presente documento pueden almacenarse en un almacenamiento de instrucciones. En algunas implementaciones, se usa circuitería separada para la predicción de ramificaciones y, en algunas realizaciones, al menos algunas instrucciones se decodifican en una o más microoperaciones, puntos de entrada de microcódigo, microinstrucciones, otras instrucciones u otras señales de control usando el microcódigo 1305. El circuito de predicción de ramas y decodificación 1303 puede implementarse usando diversos mecanismos diferentes. Ejemplos de mecanismos adecuados incluyen, pero sin limitación, tablas de consulta, implementaciones de hardware, matrices lógicas programables (PLA), memorias de sólo lectura de microcódigo (ROM), etc.

El circuito de predicción y decodificación de ramas 1303 está acoplado al circuito de asignación/renombrado 1307 que está acoplado, en algunas realizaciones, al circuito de planificador 1309. En algunas realizaciones, estos circuitos proporcionan funcionalidad de cambio de nombre de registro, asignación de registro y/o planificación realizando uno o más de: 1) cambiar el nombre de los valores de operandos lógicos a valores de operandos físicos (por ejemplo, una tabla de alias de registros en algunas realizaciones), 2) asignar bits de estado y banderas a la instrucción decodificada, y 3) planificar la instrucción decodificada para su ejecución en el circuito de ejecución fuera de una agrupación de instrucciones (por ejemplo, usando una estación de reserva en algunas realizaciones).

El circuito de planificador 1309 representa cualquier número de planificadores diferentes, incluyendo estaciones de reserva, ventana de instrucciones central, etc. El circuito de planificador 1309 está acoplado a, o incluye, archivo o archivos de registro físico 1315. Cada uno del o de los archivos de registro físico 1315 representa uno o más archivos de registro físico, de los cuales diferentes almacenan uno o más tipos de datos diferentes, tales como entero escalar, punto flotante escalar, entero empaquetado, punto flotante empaquetado, entero vectorial, punto flotante vectorial, estado (por ejemplo, un puntero de instrucción que es la dirección de la siguiente instrucción que se ejecutará), teselas, etc. En una realización, los archivos de registro físico 1315 comprenden circuitos de registros vectoriales, circuitos de registros de máscara de escritura y circuitos de registros escalares. Estos circuitos de registro pueden proporcionar registros vectoriales arquitectónicos, registros de máscara de vector y registros de propósito general. El archivo o archivos de registro físico 1315 se superponen con un circuito de retiro 1317 para ilustrar diversas formas en las que se puede implementar el cambio de nombre de registro y la ejecución en desorden (por ejemplo, usando una memoria o memorias intermedias de reordenación y un archivo o archivos de registro de retiro; usando un archivo o archivos futuros, una memoria o memorias intermedias de historial y un archivo o archivos de registro de retiro; usando mapas de registros y una agrupación de registros; etc.). El circuito de retiro 1317 y el o los archivos de registro físico 1315 están acoplados al circuito de ejecución 1311.

Si bien el cambio de nombre de registro se describe en el contexto de la ejecución fuera de orden, debe entenderse que el cambio de nombre de registro puede usarse en una arquitectura en orden. Si bien la realización ilustrada del procesador también puede incluir unidades de caché de datos e instrucciones separadas y una unidad de caché L2 compartida, unas realizaciones alternativas pueden tener una única caché interna tanto para instrucciones como para datos, tal como, por ejemplo, una caché interna de nivel 1 (L1), o múltiples niveles de caché interna. En algunas realizaciones, el sistema puede incluir una combinación de una memoria caché interna y una memoria caché externa que es externa al núcleo y/o al procesador. Alternativamente, toda la memoria caché puede ser externa al núcleo y/o al procesador.

El circuito de ejecución 1311 es un conjunto de uno o más circuitos de ejecución, incluyendo circuitos escalares 1321, circuitos vectoriales/SIMD 1323 y circuitos de posición de elemento de datos empaquetado 1327, así como circuitos de acceso a memoria 1325 para acceder a la memoria caché 1313. Los circuitos de ejecución realizan varias operaciones (por ejemplo, desplazamientos, suma, resta, multiplicación) y sobre varios tipos de datos (por ejemplo, punto flotante escalar, entero empaquetado, punto flotante empaquetado, entero vectorial, punto flotante vectorial). Si bien algunas realizaciones pueden incluir varias unidades de ejecución dedicadas a funciones específicas o conjuntos de funciones, otras realizaciones pueden incluir solo una unidad de ejecución o múltiples unidades de ejecución que realizan todas las funciones. El circuito escalar 1321 realiza operaciones escalares, el circuito vectorial/SIMD 1323 realiza operaciones vectoriales/SIMD y el circuito de operaciones de matrices 1327 realiza operaciones de matrices (teselas) detalladas en el presente documento.

A modo de ejemplo, la arquitectura central ilustrativa de cambio de nombre de registro, emisión/ejecución desordenada puede implementar una canalización de la siguiente manera: 1) un circuito de obtención de instrucciones realiza las etapas de obtención y decodificación de longitud; 2) el circuito de ramas y decodificación 1303 realiza una etapa de decodificación; 3) el circuito de asignación/renombrado 1307 realiza una etapa de asignación y una etapa de renombrado; 4) el circuito de planificador 1309 realiza una etapa de planificación; 5) el o los archivos de registro físico (acoplados a, o incluidos en, el circuito de planificador 1309 y el circuito de asignación/renombrado 1307 y una unidad de memoria realizan una etapa de lectura de registro/lectura de memoria; el circuito de ejecución 1311 realiza una

etapa de ejecución; 6) una unidad de memoria y la o las unidades de archivos de registro físico realizan una etapa de reescritura/escritura de memoria; 7) varias unidades pueden estar involucradas en la etapa de manejo de excepciones; y 8) una unidad de retiro y la o las unidades de archivos de registro físico realizan una etapa de confirmación.

El núcleo puede soportar uno o más conjuntos de instrucciones (por ejemplo, el conjunto de instrucciones x86 (con algunas extensiones que se han añadido con versiones más recientes); el conjunto de instrucciones MIPS de MIPS Technologies de Sunnyvale, CA; el conjunto de instrucciones ARM (con extensiones opcionales adicionales tal como NEON) de ARM Holdings de Sunnyvale, CA), incluyendo la instrucción o instrucciones descritas en el presente documento. En una realización, el núcleo 1390 incluye una lógica para soportar una extensión del conjunto de instrucciones de datos empaquetados (por ejemplo, AVX1, AVX2), permitiendo de este modo que las operaciones usadas por muchas aplicaciones multimedia se realicen usando datos empaquetados.

Debe entenderse que el núcleo puede soportar multiprocesamiento (ejecución de dos o más conjuntos paralelos de operaciones o subprocesos) y puede hacerlo de diversas maneras, incluyendo multiprocesamiento en porciones de tiempo, multiprocesamiento simultáneo (donde un solo núcleo físico proporciona un núcleo lógico para cada uno de los subprocesos que ese núcleo físico está multiprocesando simultáneamente) o una combinación de estos (por ejemplo, obtención y decodificación en porciones de tiempo y multiprocesamiento simultáneo a partir de entonces, tal como en la tecnología Intel® Hyperthreading).

La **Figura 14** ilustra una realización de una canalización central de procesador que soporta operaciones de matrices usando teselas. El circuito de predicción de ramas y decodificación 1403 realiza predicción de ramas de instrucciones, decodificación de instrucciones y/o ambas a partir de instrucciones almacenadas en el almacenamiento de instrucciones 1401. Por ejemplo, las instrucciones detalladas en el presente documento pueden almacenarse en un almacenamiento de instrucciones. En algunas implementaciones, se usa circuitería separada para la predicción de ramificaciones y, en algunas realizaciones, al menos algunas instrucciones se decodifican en una o más microoperaciones, puntos de entrada de microcódigo, microinstrucciones, otras instrucciones u otras señales de control usando el microcódigo 1405. El circuito de predicción de ramas y decodificación 1403 puede implementarse usando diversos mecanismos diferentes. Ejemplos de mecanismos adecuados incluyen, pero sin limitación, tablas de consulta, implementaciones de hardware, matrices lógicas programables (PLA), memorias de sólo lectura de microcódigo (ROM), etc.

El circuito de predicción y decodificación de ramas 1403 está acoplado al circuito de asignación/renombrado 1407 que está acoplado, en algunas realizaciones, al circuito de planificación 1409. En algunas realizaciones, estos circuitos proporcionan funcionalidad de cambio de nombre de registro, asignación de registro y/o planificación realizando uno o más de: 1) cambiar el nombre de los valores de operandos lógicos a valores de operandos físicos (por ejemplo, una tabla de alias de registros en algunas realizaciones), 2) asignar bits de estado y banderas a la instrucción decodificada, y 3) planificar la instrucción decodificada para su ejecución en el circuito de ejecución fuera de una agrupación de instrucciones (por ejemplo, usando una estación de reserva en algunas realizaciones).

El circuito de planificador 1409 representa cualquier número de planificadores diferentes, incluyendo estaciones de reserva, ventana de instrucciones central, etc. El circuito de planificador de la unidad o unidades de planificador 1409 está acoplada a, o incluye, el archivo o archivos de registro físico 1415. Cada uno del o de los archivos de registro físico 1415 representa uno o más archivos de registro físico, de los cuales diferentes almacenan uno o más tipos de datos diferentes, tales como entero escalar, punto flotante escalar, entero empaquetado, punto flotante empaquetado, entero vectorial, punto flotante vectorial, estado (por ejemplo, un puntero de instrucción que es la dirección de la siguiente instrucción que se ejecutará), teselas, etc. En una realización, los archivos de registro físico 1415 comprenden circuitos de registros vectoriales, circuitos de registros de máscara de escritura y circuitos de registros escalares. Estos circuitos de registro pueden proporcionar registros vectoriales arquitectónicos, registros de máscara vectorial y registros de propósito general. El archivo o archivos de registro físico 1415 se superponen con un circuito de retiro 1417 para ilustrar diversas formas en las que se puede implementar el cambio de nombre de registro y la ejecución en desorden (por ejemplo, usando una memoria o memorias intermedias de reordenación y un archivo o archivos de registro de retiro; usando un archivo o archivos futuros, una memoria o memorias intermedias de historial y un archivo o archivos de registro de retiro; usando mapas de registros y una agrupación de registros; etc.). El circuito de retiro 1417 y el o los archivos de registro físico 1415 están acoplados al circuito de ejecución 1411.

Si bien el cambio de nombre de registros se describe en el contexto de la ejecución fuera de orden, debe entenderse que el cambio de nombre de registros puede usarse en una arquitectura en orden. Si bien la realización ilustrada del procesador también puede incluir unidades de caché de datos e instrucciones separadas y una unidad de caché L2 compartida, unas realizaciones alternativas pueden tener una única caché interna tanto para instrucciones como para datos, tal como, por ejemplo, una caché interna de nivel 1 (L1), o múltiples niveles de caché interna. En algunas realizaciones, el sistema puede incluir una combinación de una memoria caché interna y una memoria caché externa que es externa al núcleo y/o al procesador. Alternativamente, toda la memoria caché puede ser externa al núcleo y/o al procesador.

El circuito de ejecución 1411 es un conjunto de uno o más circuitos de ejecución 1427 y un conjunto de uno o más circuitos de acceso a memoria 1425 para acceder a la memoria caché 1413. Los circuitos de ejecución 1427 realizan operaciones de matrices (teselas) detalladas en el presente documento.

A modo de ejemplo, la arquitectura central ilustrativa de cambio de nombre de registro, emisión/ejecución desordenada puede implementar una canalización de la siguiente manera: 1) un circuito de obtención de instrucciones realiza las etapas de obtención y decodificación de longitud; 2) el circuito de ramas y decodificación 1403 realiza una etapa de decodificación; 3) el circuito de asignación/renombrado 1407 realiza una etapa de asignación y una etapa de renombrado; 4) el circuito de planificador 1409 realiza una etapa de planificación; 5) el o los archivos de registro físico (acoplados a, o incluidos en, el circuito de planificador 1409 y el circuito de asignación/renombrado 1407 y una unidad de memoria realizan una etapa de lectura de registro/lectura de memoria; el circuito de ejecución 1411 realiza una etapa de ejecución; 6) una unidad de memoria y la o las unidades de archivos de registro físico realizan una etapa de reescritura/escritura de memoria; 7) varias unidades pueden estar involucradas en la etapa de manejo de excepciones; y 8) una unidad de retiro y la o las unidades de archivos de registro físico realizan una etapa de confirmación.

El núcleo puede soportar uno o más conjuntos de instrucciones (por ejemplo, el conjunto de instrucciones x86 (con algunas extensiones que se han añadido con versiones más recientes); el conjunto de instrucciones MIPS de MIPS Technologies de Sunnyvale, CA; el conjunto de instrucciones ARM (con extensiones opcionales adicionales tal como NEON) de ARM Holdings de Sunnyvale, CA), incluyendo la instrucción o instrucciones descritas en el presente documento. En una realización, el núcleo 1490 incluye una lógica para soportar una extensión del conjunto de instrucciones de datos empaquetados (por ejemplo, AVX1, AVX2), permitiendo de este modo que las operaciones usadas por muchas aplicaciones multimedia se realicen usando datos empaquetados.

Debe entenderse que el núcleo puede soportar multiprocesamiento (ejecución de dos o más conjuntos paralelos de operaciones o subprocesos) y puede hacerlo de diversas maneras, incluyendo multiprocesamiento en porciones de tiempo, multiprocesamiento simultáneo (donde un solo núcleo físico proporciona un núcleo lógico para cada uno de los subprocesos que ese núcleo físico está multiprocesando simultáneamente) o una combinación de estos (por ejemplo, obtención y decodificación en porciones de tiempo y multiprocesamiento simultáneo a partir de entonces, tal como en la tecnología Intel® Hyperthreading).

DISPOSICIÓN

A través de toda esta descripción, los datos se expresan usando la distribución de datos de filas principales. Los usuarios de columna principal deben trasladar los términos de acuerdo con su orientación. La **Figura 15** ilustra un ejemplo de una matriz expresada en formato de fila principal y formato de columna principal. Como se muestra, la matriz A es una matriz de 2x3. Cuando esta matriz se almacena en formato de fila principal, los elementos de datos de una fila son consecutivos. Cuando esta matriz se almacena en formato de columna principal, los elementos de datos de una columna son consecutivos. Es una propiedad bien conocida de las matrices que $A^T * B^T = (BA)^T$, donde el superíndice T significa transformación. Al leer los datos principales de columna como datos principales de fila, la matriz parece una matriz de transformación.

En algunas realizaciones, se utiliza la semántica de fila principal en el hardware, y los datos de columna principal se utilizan para intercambiar el orden de los operandos y el resultado son transformaciones de matriz, pero para las lecturas de columna principal posteriores desde la memoria, es la matriz correcta, no transformada.

Por ejemplo, si hay dos matrices de columna principal para multiplicar:

a b	g i k	ag+bh ai+bj ak+bl
c d *	h j l =	cg+dh ci+dj ck+dl
e f		eg+fh ei+fj ek+fl
(3x2)	(2x3)	(3x3)

Las matrices de entrada se almacenarían en la memoria lineal (columna principal) como:

acebdf

y

g h i j k l.

Cuando se leen estas matrices como fila principal con dimensiones 2x3 y 3x2, aparecerían como:

a c e	y	g h
b d f		i j
k l		

Intercambiando el orden y multiplicando la matriz:

g h		a c e	ag+bh cg+dh eg+fh
i j	*	b d f =	ai+bj ci+dj ei+fj
k l			ak+bl ck+dl ek+fl

La matriz de transformación está lista y luego se puede almacenar en orden de fila principal:

ag+bh cg+dh eg+fh ai+bj ci+dj ei+fj ak+bl ck+dl ek+fl

y utilizada en los cálculos de las columnas principales posteriores, es la matriz correcta sin transformar:

ag+bh	ai+bj	ak+bl
cg+dh	ci+dj	ck+dl
eg+fh	ei+fj	ek+fl

USO ILUSTRATIVO

La **Figura 16** ilustra un ejemplo de uso de matrices (teselas). En este ejemplo, la matriz C 1601 incluye dos teselas, la matriz A 1603 incluye una tesela y la matriz B 1605 incluye dos teselas. Esta figura muestra un ejemplo del bucle interno de un algoritmo para calcular una multiplicación de matrices. En este ejemplo, se usan dos teselas de resultados, tmm0 y tmm1, de la matriz C 1601 para acumular los resultados intermedios. Una tesela de la matriz A 1603 (tmm2) se reutiliza dos veces, ya que se multiplica por dos teselas de la matriz B 1605. Punteros para cargar una nueva matriz A (tesela) y dos nuevas matrices B (teselas) desde las direcciones indicadas por las flechas. Un bucle exterior, no mostrado, ajusta los punteros de las teselas C.

El código ilustrativo como se muestra incluye el uso de una instrucción de configuración de teselas y se ejecuta para configurar el uso de teselas, cargar teselas, un bucle para procesar las teselas, almacenar teselas en la memoria y liberar el uso de teselas.

La **Figura 17** ilustra una realización de uso de matrices (teselas). En 1701, se configura el uso de teselas. Por ejemplo, se ejecuta una instrucción TILECONFIG para configurar el uso de teselas, que incluye la configuración de un número de filas y columnas por tesela. Típicamente, al menos una matriz (tesela) se carga desde la memoria en 1703. Se realiza al menos una operación de matriz (tesela) en 1705 usando las matrices (teselas). En 1707, se almacena al menos una matriz (tesela) en la memoria y puede ocurrir un cambio de contexto en 1709.

CONFIGURACIÓN ILUSTRATIVA

SOPORTE DE HARDWARE DE CONFIGURACIÓN DE TESELA

Como se mencionó anteriormente, el uso de teselas típicamente necesita configurarse antes de su uso. Por ejemplo, es posible que no sea necesario el uso completo de todas las filas y columnas. La configuración de estas filas y columnas no sólo ahorra energía en algunas realizaciones, sino que la configuración puede usarse para determinar si una operación generará un error. Por ejemplo, una multiplicación de matrices de la forma (N x M) * (L x N) normalmente no funcionará si M y L no son iguales.

Antes de usar matrices que usan teselas, en algunas realizaciones, ha de configurarse el soporte de teselas. Por ejemplo, se configura cuántas filas y columnas por tesela, teselas que se van a usar, etc. Una instrucción TILECONFIG es una mejora para un ordenador en sí, ya que proporciona soporte para configurar el ordenador para usar un acelerador matricial (ya sea como parte de un núcleo de procesador o como un dispositivo externo). En particular, una ejecución de la instrucción TILECONFIG hace que se recupere una configuración de la memoria y se aplique a los ajustes de matriz (tesela) dentro de un acelerador de matriz.

CONFIGURACIÓN DEL USO DE TESELAS

La **Figura 18** ilustra el soporte para la configuración del uso de teselas de acuerdo con una realización. Una memoria 1801 contiene la descripción de tesela 1803 de las matrices (teselas) que se van a soportar.

Los recursos de ejecución de instrucciones 1811 de un procesador/núcleo 1805 almacenan aspectos de una descripción de tesela 1803 en configuraciones de tesela 1817. Las configuraciones de tesela 1817 incluyen la tabla de paletas 1813 para detallar qué teselas para una paleta están configuradas (el número de filas y columnas en cada tesela) y una marca que indica que el soporte de matriz está en uso. En particular, los recursos de ejecución de instrucciones 1811 están configurados para utilizar teselas de acuerdo con lo especificado por las configuraciones de tesela 1817. Los recursos de ejecución de instrucciones 1811 también pueden incluir un registro específico de la máquina o un registro de configuración para indicar el uso de teselas. También se establecen valores adicionales tales como los valores en uso y de inicio. Las configuraciones de tesela 1817 utilizan registro o registros 1819 para almacenar información de uso y configuración de tesela.

La **Figura 19** ilustra una realización de una descripción de las matrices (teselas) que se van a soportar. Esta es la descripción que se va a almacenar tras la ejecución de una instrucción STTILECFG. En este ejemplo, cada campo es un byte. En el byte [0], se almacena un ID de paleta 1901. El ID de paleta se usa para indexar una tabla de paleta 1813 que almacena, por ID de paleta, un número de bytes en una tesela y bytes por fila de las teselas que están asociados con este ID según se define por la configuración.

El byte 1 almacena un valor que se almacenará en un registro "startRow" 1903 y el byte 2 almacena un valor que se almacenará en un registro, startP 1905. Para soportar el reinicio de instrucciones después de estos eventos, las instrucciones almacenan información de estos registros. Para soportar instrucciones de reinicio después de eventos de interrupción como los detallados anteriormente, las instrucciones almacenan información en estos registros. El valor de startRow indica la fila que se debe usar para reiniciar. El valor de startP indica la posición dentro de la fila para operaciones de almacenamiento cuando se usan pares y, en algunas realizaciones, indica la mitad inferior de la fila (en la tesela inferior de un par) o la mitad superior de la fila (en la tesela superior de un par). Generalmente, la posición en la fila (la columna) no es necesaria.

Con la excepción de TILECONFIG y STTILECFG, la ejecución con éxito de instrucciones de matriz (tesela) establecerá startRow y startP a cero.

Cada vez que no se reinicia una instrucción de matriz (tesela) interrumpida, es responsabilidad del software poner a cero los valores startRow y startP. Por ejemplo, los manejadores de excepciones de punto flotante sin máscara podrían decidir finalizar la operación en el software y cambiar el valor del contador del programa a otra instrucción, normalmente la siguiente instrucción. En este caso, el manejador de excepciones de software debe poner a cero los valores de startRow y startP en la excepción que le presenta el sistema operativo antes de reanudar el programa. Posteriormente, el sistema operativo recargará esos valores mediante una instrucción de restauración.

El byte 3 almacena una indicación de pares (1b por tesela) de las teselas 1907.

Los bytes 16 y 17 almacenan el número de filas 1913 y columnas 1915 para la tesela 0, los bytes 18 y 19 almacenan el número de filas y columnas para la tesela 1, etc. En otras palabras, cada grupo de 2 bytes especifica un número de filas y columnas para una tesela. Si no se usa un grupo de 2 bytes para especificar los parámetros de la tesela, estos deben tener el valor cero. La especificación de parámetros de tesela para más teselas que el límite de implementación o el límite de paleta da como resultado un fallo. Las teselas no configuradas se configuran en un estado inicial con 0 filas y 0 columnas.

Finalmente, la configuración en la memoria típicamente termina con una delimitación final, tal como todos ceros durante varios bytes consecutivos.

TESELAS Y ALMACENAMIENTO DE CONFIGURACIÓN TESELAS ILUSTRATIVOS

Las **Figuras 20(A)-(D)** ilustran ejemplos de registro o registros 1819. La **Figura 20(A)** ilustra una pluralidad de registros 1819. Como se muestra en cada tesela (TMM0 2001... TMMN 2003) tiene un registro separado y cada registro almacena un tamaño de fila y columna para esa tesela particular. StartP 2011 y StartRow 2013 se almacenan en registros separados. Se configura uno o más registros de estado 2015 (por ejemplo, TILES_CONFIGURED = 1) para indicar que las teselas están configuradas para su uso.

La **Figura 20(B)** ilustra una pluralidad de registros 1819. Como se muestra, cada tesela tiene registros separados para sus filas y columnas. Por ejemplo, la configuración de filas TMM0 2021, la configuración de columnas TMM0 2023, StartP 2011 y StartRow 2013 se almacenan en registros separados. Se configura uno o más registros de estado 2015 (por ejemplo, TILES_CONFIGURED = 1) para indicar que las teselas están configuradas para su uso.

La **Figura 20(C)** Ilustra un registro único 1819. Como se muestra, este registro almacena las configuraciones de teselas (filas y columnas por tesela) 2031, StartP 2011 y StartRow 2013 se almacenan en un registro único como

registros de datos empaquetados. Se configura uno o más registros de estado 2015 (por ejemplo, TILES_CONFIGURED = 1) para indicar que las teselas están configuradas para su uso.

La **Figura 20(D)** ilustra una pluralidad de registros 1819. Como se muestra, un solo registro almacena la configuración de tesela (filas y columnas por tesela) 2031. StartP y StartRow se almacenan en registros separados 2011 y 2013. Se configura uno o más registros de estado 2015 (por ejemplo, TILES_CONFIGURED = 1) para indicar que las teselas están configuradas para su uso.

Se contemplan otras combinaciones tales como combinar los registros de inicio en un registro único donde se muestran por separado, etc.

TILE16BDP

Como se mencionó anteriormente, el hardware especial para la multiplicación general de matrices (también conocido como GEMM) es una buena opción para mejorar el pico de rendimiento de cálculo (y la eficiencia energética) de ciertas aplicaciones, tal como el aprendizaje profundo. Algunas de estas aplicaciones, incluido el aprendizaje profundo, pueden operar en elementos de datos de entrada con relativamente pocos bits sin perder precisión, siempre que los elementos de salida tengan suficientes bits (es decir, más que las entradas).

En consecuencia, los métodos y sistemas dados a conocer realizan una operación de producto escalar de matrices de punto flotante de 16 bits (TILE16BDP), que toma matrices de origen (teselas) que tienen elementos de punto flotante de 16 bits, realiza multiplicaciones de producto escalar y acumula los productos resultantes con un destino de precisión simple de 32 bits.

La instrucción TILE16BDP dada a conocer debe ser ejecutada por un procesador que incluye circuito de obtención para obtener una instrucción que tiene campos para especificar un opcode y ubicaciones de una matriz de destino M por N (tesela) que tiene elementos de precisión simple, una primera matriz de origen M por K (tesela) y una segunda matriz de origen K por N (tesela), los elementos de las matrices de origen primera y segunda especificadas que incluyen un par de valores de punto flotante de 16 bits pares e impares, en donde el opcode es para indicar que el circuito de ejecución debe, para cada elemento (M, N) de la matriz de destino especificada (tesela), convertir K pares de elementos de la fila M de la primera matriz de origen especificada (tesela) y K pares correspondientes de elementos de la columna N de la segunda matriz de origen especificada (tesela) a valores de precisión simple, multiplicar los elementos pares convertidos de las dos matrices de origen especificadas (teselas) y multiplicar por separado los elementos impares convertidos de las matrices de origen especificadas (teselas), y luego acumular esos productos por separado, una suma de los productos pares y una suma de los productos impares con contenidos previos del elemento (M, N). El procesador también incluirá otro hardware de soporte, tal como circuitería de decodificación para decodificar la instrucción obtenida y circuitería de ejecución para responder a la instrucción decodificada de acuerdo con lo especificado por el opcode.

La **Figura 21** es un diagrama de bloques que ilustra el uso de una instrucción TILE16BDP para acelerar la multiplicación de matrices, de acuerdo con algunas realizaciones. Como se muestra, la instrucción 2101 incluye campos para especificar un opcode 2102 (por ejemplo, TILE16BDP) y ubicaciones de una matriz de destino (tesela) M por N 2104 que tiene elementos de precisión simple, una primera matriz de origen (tesela) M por K 2106 y una segunda matriz de origen (tesela) K por N 2108, teniendo las matrices de origen especificadas elementos que comprenden un par de valores de punto flotante de 16 bits. Un formato de la instrucción TILE16BDP, de acuerdo con algunas realizaciones, se ilustra y describe además al menos con respecto a

Figuras 24, 25A-B y 26A-D.

Aquí, la primera matriz de origen especificada (tesela) 2112A tiene dimensiones de M=4 por K=3. La segunda matriz de origen especificada (tesela) 2112B tiene dimensiones de K=3 por N=5. Se muestra que K, M y N tienen valores diferentes con fines ilustrativos, pero en otras realizaciones pueden ser iguales.

En funcionamiento, el procesador 2100 debe responder al opcode 2102 (TILE16BDP) convirtiendo, para cada elemento (M, N) de la matriz de destino especificada (tesela) 2122, mediante el circuito de conversión 2116A, K pares de elementos de la fila M de la primera matriz de origen especificada (tesela) 2112A, y, utilizando el circuito de conversión 2116B, K pares de elementos de la columna N de la segunda matriz de origen especificada (tesela) 2112B a precisión simple, por ejemplo, punto flotante de precisión simple binary32 como se especifica en IEEE 794. El procesador 2100 debe entonces multiplicar, utilizando el circuito de multiplicación 2118, los K valores pares convertidos juntos y los K valores impares convertidos juntos, y acumular, utilizando el circuito de acumulación 2120, los K productos con contenidos previos del elemento (M,N).

Aquí se ilustra el rendimiento de la instrucción TILE16BDP para establecer el elemento de destino en la ubicación de matriz (tesela), (1, 0). En consecuencia, el procesador 2100 debe convertir, utilizando los circuitos de conversión 2116A y 2116B, K (=3) pares de elementos de la fila M (=1) de la primera matriz de origen especificada (tesela) 2112A y K (=3) pares de elementos de la columna N (=0) de la segunda matriz de origen especificada (tesela) 2112B a precisión

simple. El procesador 2100 debe entonces utilizar el circuito de multiplicación 2118 para multiplicar los elementos pares convertidos de las dos matrices de origen especificadas (teselas) y multiplicar por separado los elementos impares convertidos de las matrices de origen especificadas (teselas), y luego utilizar el circuito de acumulación 2120 para acumular esos productos por separado, una suma de los productos pares y una suma de los productos impares, con contenidos previos del elemento (M,N), que aquí es el elemento C(1,0).

Como se muestra, tres flechas viajan desde cada una de las matrices de origen primera y segunda especificadas (teselas), para indicar que las conversiones y multiplicaciones ocurren en paralelo. En algunas realizaciones, el procesador responde a la instrucción decodificada generando y almacenando resultados en cada elemento de la matriz de destino especificada (tesela) en paralelo. En algunas realizaciones, se generan nuevos valores y se almacenan en el destino fila por fila o columna por columna.

Las realizaciones dadas a conocer mejoran los enfoques alternativos al permitir que el software ejecute la instrucción TILE16BDP con tamaños de elementos de origen reducidos, lo que permite utilizar menos espacio de memoria y menos ancho de banda de memoria y mejora el rendimiento pico de cálculo (y la eficiencia energética) de ciertas aplicaciones. En algunas aplicaciones, tal como el aprendizaje profundo, se puede operar con elementos de datos de entrada con relativamente pocos bits sin perder precisión, siempre que los elementos de salida tengan suficientes bits (es decir, más que las entradas).

Se ilustran y describen con más detalle los sistemas y métodos para ejecutar una instrucción TILE16BDP, al menos con respecto a las Figuras 22A-C, 23 y 28A-B.

EJECUCIÓN ILUSTRATIVA

La **Figura 22A** es un pseudocódigo que ilustra la ejecución ejemplar de una instrucción TILE16BDP de acuerdo con algunas realizaciones. Como se muestra, la instrucción 2201 incluye un opcode 2202 (por ejemplo, TILE16BDP) y ubicaciones de una matriz de destino M por N 2204 que tiene elementos de precisión simple, una primera matriz de origen M por K 2206 y una segunda matriz de origen K por N 2208, teniendo las matrices de origen especificadas elementos que comprenden un par de valores de punto flotante de 16 bits. El opcode 2202 (TILE16BDP) indica que el procesador debe, como se muestra en el pseudocódigo 2200, para cada elemento (M,N) de la matriz de destino especificada (tesela), convertir K pares de elementos de la fila M de la primera matriz de origen especificada (tesela) y K pares de elementos de la columna N de la segunda matriz de origen especificada (tesela) a precisión simple, multiplicar los elementos pares convertidos de las dos matrices de origen especificadas (teselas) y multiplicar por separado los elementos impares convertidos de las dos matrices de origen especificadas (teselas), y luego acumular esos productos por separado, una suma de los productos pares y una suma de los productos impares, con el contenido anterior del elemento (M,N). En otras realizaciones, no mostradas, las multiplicaciones tienen lugar antes de las conversiones.

En funcionamiento, M, K y N se deben especificar de una o más de varias maneras: como operandos de la instrucción TILE16BDP (como aquí), como sufijos o prefijos del opcode especificado (en el presente documento se utiliza un asterisco como abreviatura para referirse a esos sufijos y prefijos opcionales), como parte de un inmediato proporcionado con la instrucción (por ejemplo, K, M y N se deben especificar cada uno como 8 bits diferentes de un inmediato de 32 bits), como parte de registros de control programados por software (por ejemplo, XTILECONFIG es un registro cargado por una instrucción de configuración de matriz, tal como TILECFG o una instrucción XRSTORE*, y se almacena por una instrucción de guardado de matriz, tal como XSAVE*), o incluso como valores predeterminados arquitectónicos.

La instrucción 2201 especifica además la ubicación de la matriz de destino (tesela) 2204, la ubicación de la primera matriz de origen (tesela) 2206 y la ubicación de la segunda matriz de origen (tesela) 2208. Cada ubicación de matriz (tesela) especificada puede apuntar a cualquiera de las siguientes: una ubicación de memoria, una colección de registros vectoriales y una colección de registros de tesela.

La **Figura 22B** es un pseudocódigo que ilustra la ejecución ejemplar de una instrucción TILE16BDP de acuerdo con algunas realizaciones. Como se muestra, la instrucción 2211 incluye un opcode 2212 (por ejemplo, TILE16BDP) y ubicaciones de una matriz de destino M por N 2214 que tiene elementos de precisión simple, una primera matriz de origen M por K 2216 y una segunda matriz de origen K por N 2218, teniendo las matrices de origen especificadas elementos que comprenden un par de valores de punto flotante de 16 bits. El pseudocódigo 2210 es similar al pseudocódigo 2200 (**Figura 22A**), excepto que los productos de los elementos de origen impares se acumulan con el elemento de destino antes que los de los elementos de origen pares.

La ejecución de la instrucción TILE16BDP se ilustra y describe con más detalle en relación con las **Figuras 21, 22B, 23**, 28A-B y **29A-B**. El formato de las instrucciones TILE16BDP se ilustra y describe con más detalle en relación con las **Figuras 24-26**.

La **Figura 22C** es un pseudocódigo para funciones auxiliares ejemplares para su uso con una instrucción TILE16BDP, de acuerdo con algunas realizaciones. Como se muestra, el pseudocódigo 2220 define una función make_fp32(), una

función write_row_and_zero(), una función zero_upper_rows() y una función zero_tileconfig_start(), todas las cuales son utilizadas por el pseudocódigo TILE16BDP de la **Figura 22A**.

La ejecución de la instrucción TILE16BDP se ilustra y describe con más detalle en relación con las **Figuras 21, 22B, 23**, 28A-B y **29A-B**. El formato de las instrucciones TILE16BDP se ilustra y describe con más detalle en relación con las **Figuras 24-26**.

MÉTODO O MÉTODOS EJEMPLARES DE EJECUCIÓN

La **Figura 23** es un diagrama de flujo de bloques que ilustra un procesador que responde a una instrucción TILE16BDP. Como se muestra en el diagrama de flujo 2300, en 2301, el procesador debe obtener, utilizando un circuito de obtención, una instrucción que tiene campos para especificar un opcode y ubicaciones de una matriz de destino M por N que tiene elementos de precisión simple, una primera matriz de origen M por K y una segunda matriz de origen K por N, teniendo las matrices de origen especificadas elementos que comprenden un par de valores de punto flotante de 16 bits.

En realizaciones que utilizan un archivo de registro físico del procesador para almacenar matrices (teselas), dado que los elementos de destino son dos veces más anchos que los elementos de origen, tener un par de formatos de punto flotante de 16 bits en la de origen permite un uso eficiente, cuando las matrices (teselas) son una colección de registros vectoriales, del mismo tipo de registro vectorial, ya sea un registro xmm de 128 bits, un registro ymm de 256 bits o un registro zmm de 512 bits. Este uso eficiente también se puede conseguir cuando las matrices se almacenan en registros de teselas. En otras realizaciones, no mostradas, un único vector de origen que tiene elementos de punto flotante de 16 bits se convierte en elementos de 32 bits almacenados en un vector de destino que tiene la mitad del ancho del vector de origen.

El opcode especificado es para indicar que el circuito de ejecución debe, para cada elemento (M, N) de la matriz de destino especificada, convertir k pares de elementos de la fila M de la primera matriz de origen especificada y k pares de elementos de la columna n de la segunda matriz de origen especificada a precisión simple, multiplicar los elementos pares convertidos de las dos matrices de origen especificadas (teselas) y multiplicar por separado los elementos impares convertidos de las dos matrices de origen especificadas (teselas), y luego acumular esos productos por separado, una suma de los productos pares y una suma de los productos impares, con los contenidos anteriores del elemento (m, n).

En 2303, el procesador debe decodificar, utilizando circuitos de decodificación, la instrucción obtenida. Por ejemplo, la instrucción TILE16BDP obtenida se decodifica mediante un circuito de decodificación tal como el que se detalla en el presente documento. En el contexto del sistema ilustrado, el circuito de decodificación es similar al ilustrado y descrito al menos con respecto a las **Figuras 13, 14** y 28A-B.

A las 2305 se planifica la ejecución de la instrucción decodificada (según sea necesario), lo cual es opcional (como lo indica su borde punteado) en la medida en que puede ocurrir en un momento diferente o no ocurrir en absoluto. En 2307, el procesador ha de responder, usando circuitería de ejecución, a la instrucción decodificada como se especifica por el opcode.

En algunas realizaciones, la instrucción se confirma o se retira en 2309, lo cual es opcional (como lo indica su borde punteado) en la medida en que puede ocurrir en un momento diferente o no ocurrir en absoluto.

Los circuitos de ejecución se ilustran y describen con más detalle con respecto a las **Figuras 3-14**. En algunas realizaciones, el circuito de ejecución es un acelerador de operaciones de matrices, tal como el ilustrado y descrito como acelerador 307 (**Figura 3**). En algunas realizaciones, el circuito de ejecución es un circuito de operaciones de matrices, tal como el circuito de operaciones de matrices 405 (**Figura 4**), 505 (**Figura 5**) o 1213 (**Figura 12**) y 1327 (**Figura 13**).

FORMATO O FORMATOS DE INSTRUCCIÓN EJEMPLARES

La **Figura 24** es un diagrama de bloques que ilustra un formato de una instrucción TILE16BDP, de acuerdo con algunas realizaciones. Como se muestra, la instrucción TILE16BDP 2400 incluye campos para especificar un opcode 2402 (TILE16BDP*), que indica que el procesador debe, para cada elemento (M,N) de la matriz de destino especificada, convertir K pares de elementos de la fila M de la primera matriz de origen especificada y K pares de elementos de la columna N de la segunda matriz de origen especificada a precisión simple, multiplicar los elementos pares convertidos de las dos matrices de origen especificadas (teselas) y multiplicar por separado los elementos impares convertidos de las dos matrices de origen especificadas (teselas), y luego acumular esos productos por separado, una suma de los productos pares y una suma de los productos impares, con los contenidos anteriores del elemento (m,n).

La instrucción 2400 incluye además la ubicación de matriz de destino (tesela) 2404, la ubicación de primera matriz de origen (tesela) 2406 y la ubicación de segunda matriz de origen (tesela) 2408. Cada una de las ubicaciones de matriz

de origen y destino especificadas puede estar en cualquiera de las siguientes: una ubicación de memoria, una colección de registros vectoriales y una colección de registros de tesela.

La instrucción TILE16BDP 2400 incluye además varios parámetros opcionales para controlar el comportamiento del procesador, incluido el formato del elemento de origen 2410, K 2412, M 2414 y N 2416. En algunas realizaciones, N y M son cada uno de 4, 8, 16 y 32 (pero la invención no establece un límite superior ni para M ni para N, que podría ser 32, 64 o mayor). En algunas realizaciones, N y M son cada uno un número entero mayor o igual a 4.

El opcode 2402 se muestra incluyendo un asterisco, que sirve para indicar que se pueden agregar prefijos y/o sufijos adicionales para especificar el comportamiento de la instrucción. Se pueden especificar uno o más de los modificadores de instrucciones 2410, 2412, 2414 y 2416 utilizando prefijos o sufijos para el opcode 2402.

En algunas realizaciones, uno o más de los modificadores de instrucciones 2410, 2412, 2414 y 2416 opcionales están codificados en un campo inmediato (no mostrado) incluido opcionalmente con la instrucción 2400. En algunas realizaciones, uno o más de los modificadores de instrucciones 2410, 2412, 2414 y 2416 opcionales se especifican a través de un registro de configuración/estado (por ejemplo, XTILECONFIG).

Cuando uno o más de los modificadores 2410, 2412, 2414 y 2416 opcionales no están especificados por la instrucción, a veces se utilizan valores predeterminados o parámetros implícitos que se heredan de otras partes de la arquitectura de tesela.

SISTEMAS, PROCESADORES Y EMULACIÓN EJEMPLARES DETALLADOS

En el presente documento se detallan ejemplos de hardware, software, etc. para ejecutar las instrucciones descritas anteriormente. Por ejemplo, lo que se describe a continuación detalla aspectos de la ejecución de instrucciones, incluyendo diversas etapas de canalización, tal como extraer, decodificar, planificar, ejecutar, retirar, etc.

CONJUNTOS DE INSTRUCCIONES

Un conjunto de instrucciones puede incluir uno o más formatos de instrucción. Un formato de instrucción dado puede definir varios campos (por ejemplo, número de bits, ubicación de los bits) para especificar, entre otras cosas, la operación que se realizará (por ejemplo, opcode) y el o los operandos en los que se realizará esa operación y/u otros campos de datos (por ejemplo, máscara). Algunos formatos de instrucción se desglosan aún más a través de la definición de plantillas de instrucciones (o subformatos). Por ejemplo, las plantillas de instrucciones de un formato de instrucción dado pueden definirse para tener diferentes subconjuntos de los campos del formato de instrucción (los campos incluidos normalmente están en el mismo orden, pero al menos algunos tienen diferentes posiciones de bits porque hay menos campos incluidos) y/o definirse para que un campo dado se interprete de manera diferente. De este modo, cada instrucción de una ISA se expresa utilizando un formato de instrucción dado (y, si está definido, en una de las plantillas de instrucciones de ese formato de instrucción) e incluye campos para especificar la operación y los operandos. Por ejemplo, una instrucción ADD ejemplar tiene un opcode específico y un formato de instrucción que incluye un campo opcode para especificar ese opcode y campos de operandos para seleccionar operandos (source1/destination y source2); y una ocurrencia de esta instrucción ADD en un flujo de instrucciones tendrá contenidos específicos en los campos de operandos que seleccionan operandos específicos. Se ha publicado y/o lanzado un conjunto de extensiones SIMD denominadas Extensiones Vectoriales Avanzadas (AVX) (AVX1 y AVX2) y que utilizan el esquema de codificación Extensiones Vectoriales (VEX) (por ejemplo, consulte el manual del desarrollador de software de las arquitecturas Intel® 64 e IA-32, septiembre de 2014; y consulte la referencia de programación de extensiones vectoriales avanzadas Intel®, octubre de 2014).

FORMATO O FORMATOS DE INSTRUCCIÓN EJEMPLARES

Las realizaciones de la o las instrucciones descritas en el presente documento pueden realizarse en diferentes formatos. Además, a continuación se detallan sistemas, arquitecturas y canales ejemplares. Se pueden ejecutar realizaciones de la instrucción o instrucciones en tales sistemas, arquitecturas y canalizaciones, pero no se limitan a las detalladas.

Formato de instrucciones genérico compatible con vectores

Un formato de instrucción compatible con vectores es un formato de instrucción adecuado para instrucciones vectoriales (por ejemplo, hay ciertos campos específicos para operaciones vectoriales). Si bien se describen realizaciones en las que se soportan operaciones vectoriales y escalares a través del formato de instrucción compatible con vectores, realizaciones alternativas utilizan únicamente operaciones vectoriales en el formato de instrucción compatible con vectores.

Las **Figuras 25A-25B** son diagramas de bloques que ilustran un formato de instrucción compatible con vectores genérico y plantillas de instrucciones del mismo de acuerdo con realizaciones. La **Figura 25A** es un diagrama de bloques que ilustra un formato de instrucción compatible con vectores genérico y plantillas de instrucciones de clase

A del mismo de acuerdo con realizaciones; mientras que la **Figura 25B** es un diagrama de bloques que ilustra el formato de instrucción compatible con vectores genérico y plantillas de instrucciones de clase B del mismo de acuerdo con realizaciones. Específicamente, un formato de instrucción compatible con vectores genérico 2500 para el cual se definen plantillas de instrucciones de clase A y clase B, las cuales incluyen plantillas de instrucciones sin acceso a memoria 2505 y plantillas de instrucciones con acceso a memoria 2520. El término genérico en el contexto del formato de instrucción compatible con vectores se refiere a que el formato de instrucción no está vinculado a un conjunto de instrucciones específico.

Si bien se describirán realizaciones en las que el formato de instrucción compatible con vectores soporta lo siguiente: una longitud (o tamaño) de operando de vector de 64 bytes con anchos (o tamaños) de elementos de datos de 32 bits (4 bytes) o 64 bits (8 bytes) (y por lo tanto, un vector de 64 bytes consta de 16 elementos de tamaño de doubleword o, alternativamente, 8 elementos de tamaño de quadword); una longitud (o tamaño) de operando de vector de 64 bytes con anchos (o tamaños) de elementos de datos de 16 bits (2 bytes) u 8 bits (1 byte); una longitud (o tamaño) de operando de vector de 32 bytes con anchos (o tamaños) de elementos de datos de 32 bits (4 bytes), 64 bits (8 bytes), 16 bits (2 bytes) u 8 bits (1 byte); y una longitud (o tamaño) de operando vectorial de 16 bytes con anchos (o tamaños) de elementos de datos de 32 bits (4 bytes), 64 bits (8 bytes), 16 bits (2 bytes) u 8 bits (1 byte); realizaciones alternativas pueden soportar más, menos y/o diferentes tamaños de operandos vectoriales (por ejemplo, operandos vectoriales de 256 bytes) con más, menos o diferentes anchos de elementos de datos (por ejemplo, anchos de elementos de datos de 128 bits (16 bytes)).

Las plantillas de instrucciones de clase A en la **Figura 25A** incluyen: 1) dentro de las plantillas de instrucciones sin acceso a memoria 2505 se muestra una plantilla de instrucción de operación de tipo de control de redondeo completo, sin acceso a memoria 2510 y una plantilla de instrucción de operación de tipo de transformación de datos, sin acceso a memoria 2515; y 2) dentro de las plantillas de instrucciones con acceso a memoria 2520 se muestra una plantilla de instrucción de acceso a memoria, temporal 2525 y una plantilla de instrucción de acceso a memoria, no temporal 2530. Las plantillas de instrucciones de clase B en la **Figura 25B** incluyen: 1) dentro de las plantillas de instrucciones sin acceso a memoria 2505 se muestra una plantilla de instrucción de operación de tipo de control de redondeo parcial, control de máscara de escritura, sin acceso a memoria 2512 y una plantilla de instrucción de operación de tipo de control de máscara de escritura, sin acceso a memoria, vsize 2517; y 2) dentro de las plantillas de instrucciones con acceso a memoria 2520 se muestra una plantilla de instrucción de acceso a memoria, control de máscara de escritura 2527.

El formato de instrucción compatible con vectores genérico 2500 incluye los siguientes campos enumerados a continuación en el orden ilustrado en las **Figuras 25A-25B**.

Campo de formato 2540 - un valor específico (un valor de identificador de formato de instrucción) en este campo identifica de forma única el formato de instrucción compatible con vectores y, por lo tanto, las apariciones de instrucciones en el formato de instrucción compatible con vectores en los flujos de instrucciones. Como tal, este campo es opcional en el sentido de que no es necesario para un conjunto de instrucciones que solo tiene el formato de instrucción compatible con vectores genérico.

Campo de operación base 2542 - su contenido distingue diferentes operaciones base.

Campo de índice de registro 2544 - su contenido, directamente o mediante la generación de direcciones, especifica las ubicaciones de los operandos de origen y destino, ya sea en registros o en memoria. Estos incluyen un número suficiente de bits para seleccionar N registros de un archivo de registros PxQ (por ejemplo, 32x512, 16x128, 32x1024, 64x1024). Si bien una realización soporta hasta tres registro de orígenes y uno de destino, realizaciones alternativas pueden soportar más o menos registros de orígenes y de destino (por ejemplo, pueden soportar hasta dos orígenes donde uno de estos orígenes también actúa como destino, pueden soportar hasta tres orígenes donde uno de estos orígenes también actúa como destino, pueden soportar hasta dos orígenes y un destino).

Campo modificador 2546 - su contenido distingue las ocurrencias de instrucciones en el formato de instrucción vectorial genérico que especifican acceso a memoria de aquellas que no lo hacen; es decir, entre plantillas de instrucciones sin acceso a memoria 2505 y plantillas de instrucciones con acceso a memoria 2520. Las operaciones de acceso a memoria leen y/o escriben en la jerarquía de memoria (en algunos casos, especificando las direcciones de origen y/o destino utilizando valores en registros), mientras que las operaciones sin acceso a memoria no lo hacen (por ejemplo, el origen y los destinos son registros). Si bien en una realización este campo también selecciona entre tres formas diferentes de realizar cálculos de direcciones de memoria, realizaciones alternativas pueden soportar más, menos o diferentes formas de realizar cálculos de direcciones de memoria.

Campo de operación de aumento 2550 - su contenido distingue cuál de una diversidad de operaciones diferentes se realizará además de la operación de base. Este campo es específico del contexto. En una realización, este campo se divide en un campo de clase 2568, un campo alfa 2552 y un campo beta 2554. El campo de operación de aumento 2550 permite que se realicen grupos comunes de operaciones en una sola instrucción en lugar de 2, 3 o 4 instrucciones.

Campo de escala 2560 - su contenido permite escalar el contenido del campo de índice para la generación de direcciones de memoria (por ejemplo, para la generación de direcciones que utiliza 2^{escala} * índice + base).

Campo de desplazamiento 2562A - su contenido se utiliza como parte de la generación de direcciones de memoria (por ejemplo, para la generación de direcciones que utiliza 2^{escala} * índice + base + desplazamiento).

Campo de factor de desplazamiento 2562B (obsérvese que la yuxtaposición del campo de desplazamiento 2562A directamente sobre el campo de factor de desplazamiento 2562B indica que se usa uno u otro) - su contenido se usa como parte de la generación de direcciones; especifica un factor de desplazamiento a escalar de acuerdo con el tamaño de un acceso a memoria (N) - donde N es el número de bytes en el acceso a memoria (por ejemplo, para la generación de direcciones que usa 2^{escala} * índice + base + desplazamiento escalado). Los bits redundantes de bajo orden se ignoran y, por lo tanto, el contenido del campo del factor de desplazamiento se multiplica por el tamaño total de los operandos de memoria (N) para generar el desplazamiento final que se usará para calcular una dirección efectiva. El valor de N está determinado por el hardware del procesador en tiempo de ejecución basándose en el campo opcode completo 2574 (descrito más adelante en el presente documento) y el campo de manipulación de datos 2554C. El campo de desplazamiento 2562A y el campo de factor de desplazamiento 2562B son opcionales en el sentido de que no se usan para las plantillas de instrucciones sin acceso a memoria 2505 y/o diferentes realizaciones pueden implementar únicamente uno o ninguno de los dos.

Campo de ancho de elemento de datos 2564 - su contenido distingue cuál de varios anchos de elementos de datos se utilizará (en algunas realizaciones para todas las instrucciones; en otras realizaciones solo para algunas de las instrucciones). Este campo es opcional en el sentido de que no es necesario si solo se soporta un ancho de elemento de datos y/o se soportan anchos de elementos de datos utilizando algún aspecto de los códigos de operación.

Campo de máscara de escritura 2570 - su contenido controla, en función de la posición de cada elemento de datos, si esa posición del elemento de datos en el operando del vector de destino refleja el resultado de la operación base y la operación de aumento. Las plantillas de instrucciones de clase A soportan el enmascaramiento de escritura de fusión, mientras que las plantillas de instrucciones de clase B soportan el enmascaramiento de escritura tanto de fusión como de puesta a cero. Cuando se fusionan, las máscaras vectoriales permiten proteger de actualizaciones cualquier conjunto de elementos en el destino durante la ejecución de cualquier operación (especificada por la operación de base y la operación de aumento); en otra realización, conservando el valor antiguo de cada elemento del destino donde el bit de máscara correspondiente tiene un 0. Por el contrario, cuando se ponen a cero las máscaras vectoriales se permite poner a cero cualquier conjunto de elementos en el destino durante la ejecución de cualquier operación (especificada por la operación base y la operación de aumento); en una realización, un elemento del destino se establece en 0 cuando el bit de máscara correspondiente tiene un valor 0. Un subconjunto de esta funcionalidad es la capacidad de controlar la longitud de vector de la operación que se está realizando (es decir, el lapso de elementos que se están modificando, desde el primero hasta el último); sin embargo, no es necesario que los elementos que se modifican sean consecutivos. Por lo tanto, el campo de máscara de escritura 2570 permite operaciones de vector parcial, incluyendo cargas, almacenamientos, aritméticas, lógicas, etc. Si bien se describen realizaciones en las que el contenido del campo de máscara de escritura 2570 selecciona uno de varios registros de máscara de escritura que contienen la máscara de escritura que se utilizará (y por lo tanto, el contenido del campo de máscara de escritura 2570 identifica indirectamente ese enmascaramiento que se realizará), realizaciones alternativas en su lugar o adicionales permiten que el contenido del campo de escritura de máscara 2570 especifique directamente el enmascaramiento que se realizará.

Campo inmediato 2572 - su contenido permite la especificación de un inmediato. Este campo es opcional en el sentido de que no está presente en una implementación del formato genérico compatible con vectores que no soporta inmediato y no está presente en instrucciones que no utilizan un inmediato.

Campo de clase 2568 - su contenido distingue entre diferentes clases de instrucciones. Con referencia a las Figuras 25A-B, el contenido de este campo selecciona entre instrucciones de clase A y clase B. En las Figuras 25A-B, se usan cuadrados de esquinas redondeadas para indicar que un valor específico está presente en un campo (por ejemplo, clase A 2568A y clase B 2568B para el campo de clase 2568 respectivamente en las Figuras 25A-B).

PLANTILLAS DE INSTRUCCIONES DE CLASE A

En el caso de las plantillas de instrucciones sin acceso a memoria 2505 de clase A, el campo alfa 2552 se interpreta como un campo RS 2552A, cuyo contenido distingue cuál de los diferentes tipos de operación de aumento se va a realizar (por ejemplo, el redondeo 2552A.1 y la transformada de datos 2552A.2 se especifican, respectivamente, para las plantillas de instrucciones de operación de tipo redondeo sin acceso a memoria 2510 y de operación de tipo transformada de datos sin acceso a memoria 2515), mientras que el campo beta 2554 distingue cuál de las operaciones del tipo especificado se va a realizar. En las plantillas de instrucciones sin acceso a memoria 2505, no están presentes el campo de escala 2560, el campo de desplazamiento 2562A ni el campo de escala de desplazamiento 2562B.

PLANTILLAS DE INSTRUCCIONES SIN ACCESO A MEMORIA - OPERACIÓN DE TIPO DE CONTROL DE REDONDEO COMPLETO

En la plantilla de instrucción de operación de tipo de control de redondeo completo sin acceso a memoria 2510, el campo beta 2554 se interpreta como un campo de control de redondeo 2554A, cuyo contenido o contenidos proporcionan redondeo estático. Si bien en las realizaciones descritas el campo de control de redondeo 2554A incluye un campo de supresión de todas las excepciones de punto flotante (SAE) 2556 y un campo de control de operación de redondeo 2558, realizaciones alternativas pueden soportar la codificación de ambos conceptos en el mismo campo o solo tener uno u otro de estos conceptos/campos (por ejemplo, pueden tener solo el campo de control de operación de redondeo 2558).

Campo SAE 2556 - su contenido distingue si se debe deshabilitar o no el informe de eventos de excepción; cuando el contenido del campo SAE 2556 indica que la supresión está habilitada, una instrucción dada no informa un tipo de indicador de excepción de punto flotante y no genera un manejador de excepciones de punto flotante.

Campo de control de operación de redondeo 2558 - su contenido distingue cuál de un grupo de operaciones de redondeo realizar (por ejemplo, redondeo hacia arriba, redondeo hacia abajo, redondeo hacia cero y redondeo al más cercano). Por tanto, el campo de control de operación de redondeo 2558 permite el cambio del modo de redondeo por instrucción. En una realización donde un procesador incluye un registro de control para especificar modos de redondeo, el contenido del campo de control de operación de redondeo 2550 sobrescribe ese valor de registro.

PLANTILLAS DE INSTRUCCIONES SIN ACCESO A MEMORIA - OPERACIÓN DE TIPO DE TRANSFORMACIÓN DE DATOS

En la plantilla de instrucciones de operación de tipo de transformación de datos sin acceso a memoria 2515, el campo beta 2554 se interpreta como un campo de transformación de datos 2554B, cuyo contenido distingue cuál de un número de transformaciones de datos se va a realizar (por ejemplo, sin transformación de datos, mezcla, difusión).

En el caso de una plantilla de instrucción de acceso a memoria 2520 de clase A, el campo alfa 2552 se interpreta como un campo de sugerencia de desalojo 2552B, cuyo contenido distingue cuál de las sugerencias de desalojo se va a usar (en la **Figura 25A**, temporal 2552B.1 y no temporal 2552B.2 se especifican respectivamente para la plantilla de instrucción de acceso a memoria, temporal 2525 y la plantilla de instrucción de acceso a memoria, no temporal 2530), mientras que el campo beta 2554 se interpreta como un campo de manipulación de datos 2554C, cuyo contenido distingue cuál de un número de operaciones de manipulación de datos (también conocidas como primitivas) se va a realizar (por ejemplo, sin manipulación; difusión; conversión ascendente de un origen y conversión descendente de un destino). Las plantillas de instrucciones de acceso a memoria 2520 incluyen el campo de escala 2560 y, opcionalmente, el campo de desplazamiento 2562A o el campo de escala de desplazamiento 2562B.

Las instrucciones de memoria vectorial realizan cargas vectoriales desde y almacenamientos vectoriales en la memoria, con soporte de conversión. Al igual que con las instrucciones vectoriales regulares, las instrucciones de memoria vectorial transfieren datos desde/hacia la memoria elemento por elemento, y los elementos que realmente se transfieren están determinados por el contenido de la máscara de vector que se selecciona como máscara de escritura.

PLANTILLAS DE INSTRUCCIONES DE ACCESO A MEMORIA - TEMPORALES

Los datos temporales son datos que probablemente se reutilizarán lo suficientemente pronto como para beneficiarse del almacenamiento en caché. Sin embargo, esto es una sugerencia, y diferentes procesadores pueden implementarla de diferentes maneras, incluso ignorando la sugerencia por completo.

PLANTILLAS DE INSTRUCCIONES DE ACCESO A MEMORIA - NO TEMPORALES

Los datos no temporalefres son datos que es poco probable que se reutilicen lo suficientemente pronto como para beneficiarse del almacenamiento en caché en la memoria caché de primer nivel y se les debe dar prioridad para su expulsión. Sin embargo, esto es una sugerencia, y diferentes procesadores pueden implementarla de diferentes maneras, incluso ignorando la sugerencia por completo.

PLANTILLAS DE INSTRUCCIONES DE LA CLASE B

En el caso de las plantillas de instrucciones de clase B, el campo alfa 2552 se interpreta como un campo de control de máscara de escritura (Z) 2552C, cuyo contenido distingue si el enmascaramiento de escritura controlado por el campo de máscara de escritura 2570 debe ser una fusión o una puesta a cero.

En el caso de las plantillas de instrucciones sin acceso a memoria 2505 de clase B, parte del campo beta 2554 se interpreta como un campo RL 2557A, cuyo contenido distingue cuál de los diferentes tipos de operación de aumento se va a realizar (por ejemplo, redondeo 2557A.1) y la longitud del vector (VSIZE) 2557A.2 se especifica

respectivamente para la plantilla de instrucción de la operación de tipo de control de redondeo parcial de control de máscara de escritura sin acceso a memoria 2512 y la plantilla de instrucción de la operación de tipo VSIZE de control de máscara de escritura sin acceso a memoria 2517), mientras que el resto del campo beta 2554 distingue cuál de las operaciones del tipo especificado se va a realizar. En las plantillas de instrucciones sin acceso a memoria 2505, el campo de escala 2560, el campo de desplazamiento 2562A y el campo de escala de desplazamiento 2562B no están presentes.

En la plantilla de instrucción de operación de tipo control de redondeo parcial, de control de máscara de escritura, sin acceso a memoria 2510, el resto del campo beta 2554 se interpreta como un campo de operación de redondeo 2559A y se inhabilita la notificación de eventos de excepción (una instrucción dada no notifica tipo alguno de indicador de excepción de coma flotante y no genera manejador de excepciones de coma flotante alguno).

Campo de control de operación de redondeo 2559A - al igual que el campo de control de operación de redondeo 2558, su contenido distingue cuál de un grupo de operaciones de redondeo realizar (por ejemplo, redondeo hacia arriba, redondeo hacia abajo, redondeo hacia cero y redondeo hacia el valor más cercano). De este modo, el campo de control de operación de redondeo 2559A permite cambiar el modo de redondeo según cada instrucción. En una realización donde un procesador incluye un registro de control para especificar modos de redondeo, el contenido del campo de control de operación de redondeo 2550 sobrescribe ese valor de registro.

En la plantilla de instrucciones de operación de tipo VSIZE de control de máscara de escritura sin acceso a memoria 2517, el resto del campo beta 2554 se interpreta como un campo de longitud de vector 2559B, cuyo contenido distingue cuál de un número de longitudes de vector de datos se va a realizar en (por ejemplo, 128, 256 o 512 bytes).

En el caso de una plantilla de instrucción con acceso a memoria 2520 de clase B, parte del campo beta 2554 se interpreta como un campo de difusión 2557B, cuyo contenido distingue si se va a realizar o no la operación de manipulación de datos de tipo difusión, mientras que el resto del campo beta 2554 se interpreta como el campo de longitud de vector 2559B. Las plantillas de instrucciones de acceso a memoria 2520 incluyen el campo de escala 2560 y, opcionalmente, el campo de desplazamiento 2562A o el campo de escala de desplazamiento 2562B.

Con respecto al formato de instrucción compatible con vectores genérico 2500, se muestra un campo opcode completo 2574 que incluye el campo de formato 2540, el campo de operación base 2542 y el campo de ancho de elemento de datos 2564. Si bien se muestra una realización donde el campo opcode completo 2574 incluye todos estos campos, el campo opcode completo 2574 incluye menos de todos estos campos en realizaciones que no los soportan todos. El campo opcode completo 2574 proporciona el código de operación (opcode).

El campo de operación de aumento 2550, el campo de ancho de elemento de datos 2564 y el campo de máscara de escritura 2570 permiten que estas características se especifiquen por instrucción en el formato de instrucción compatible con vectores genérico.

La combinación del campo de máscara de escritura y el campo de ancho de elemento de datos crea instrucciones de un tipo concreto que permiten que se aplique la máscara basándose en diferentes anchuras de elementos de datos.

Las diversas plantillas de instrucciones que se encuentran dentro de la clase A y la clase B son beneficiosas en diferentes situaciones. En algunas realizaciones, diferentes procesadores o diferentes núcleos dentro de un procesador pueden soportar solo la clase A, solo la clase B o ambas clases. Por ejemplo, un núcleo fuera de orden de propósito general de alto rendimiento destinado a computación de propósito general puede soportar solo la clase B, un núcleo destinado principalmente a computación gráfica y/o científica (rendimiento) puede soportar solo la clase A, y un núcleo destinado a ambos puede soportar ambas (por supuesto, un núcleo que tiene alguna combinación de plantillas e instrucciones de ambas clases pero no todas las plantillas e instrucciones de ambas clases están dentro del alcance de la invención). Además, un solo procesador puede incluir varios núcleos, todos ellos soportan la misma clase o en los que diferentes núcleos soportan clases diferentes. Por ejemplo, en un procesador con núcleos gráficos y de propósito general separados, uno de los núcleos gráficos destinados principalmente a gráficos y/o computación científica puede soportar solo la clase A, mientras que uno o más de los núcleos de propósito general pueden ser núcleos de propósito general de alto rendimiento con ejecución fuera de orden y cambio de nombre de registro destinados a computación de propósito general que soporten solo la clase B. Otro procesador que no tenga un núcleo gráfico separado puede incluir uno o más núcleos de propósito general en orden o fuera de orden que soporten tanto la clase A como la clase B. Por supuesto, las características de una clase también pueden implementarse en la otra clase en diferentes realizaciones. Los programas escritos en un lenguaje de alto nivel se colocarían (por ejemplo, compilados justo a tiempo o compilados estáticamente) en una variedad de formas ejecutables diferentes, incluyendo: 1) una forma que tiene solo instrucciones de la o las clases soportadas por el procesador de destino para su ejecución; o 2) una forma que tiene rutinas alternativas escritas utilizando diferentes combinaciones de las instrucciones de todas las clases y que tiene un código de flujo de control que selecciona las rutinas a ejecutar en función de las instrucciones soportadas por el procesador que actualmente está ejecutando el código.

FORMATO DE INSTRUCCIÓN COMPATIBLE CON VECTORES ESPECÍFICO EJEMPLAR

La **Figura 26A** es un diagrama de bloques que ilustra un formato de instrucción compatible con vectores específico ejemplar de acuerdo con realizaciones. La **Figura 26A** muestra un formato de instrucción compatible con vectores específico 2600 que es específico en el sentido de que especifica la ubicación, el tamaño, la interpretación y el orden de los campos, así como los valores para algunos de esos campos. El formato de instrucción compatible con vectores específico 2600 se puede utilizar para ampliar el conjunto de instrucciones x86 y, por lo tanto, algunos de los campos son similares o iguales a los utilizados en el conjunto de instrucciones x86 existente y su extensión (por ejemplo, AVX). Este formato sigue siendo coherente con el campo de codificación de prefijo, el campo de bytes de opcode real, el campo MOD R/M, el campo SIB, el campo de desplazamiento y los campos inmediatos del conjunto de instrucciones x86 existente con extensiones. Si ilustran los campos de la **Figura 25** en los que se mapean los campos de la **Figura 26A**.

Debe entenderse que, aunque las realizaciones se describen con referencia al formato de instrucción compatible con vectores específico 2600 en el contexto del formato de instrucción compatible con vectores genérico 2500 para fines ilustrativos, la invención no está limitada al formato de instrucción compatible con vectores específico 2600 excepto donde se reivindique. Por ejemplo, el formato de instrucción compatible con vectores genérico 2500 contempla una variedad de tamaños posibles para los diversos campos, mientras que el formato de instrucción compatible con vectores específico 2600 se muestra como si tuviera campos de tamaños específicos. A modo de ejemplo específico, mientras que el campo de ancho de elemento de datos 2564 se ilustra como un campo de un bit en el formato de instrucción compatible con vectores específico 2600, la invención no está así limitada (es decir, el formato de instrucción compatible con vectores genérico 2500 contempla otros tamaños del campo de ancho de elemento de datos 2564).

El formato de instrucción compatible con vectores genérico 2500 incluye los siguientes campos enumerados a continuación en el orden ilustrado en la **Figura 26A.**

Prefijo EVEX 2602 (bytes 0-3) - está codificado en formato de cuatro bytes.

Campo de formato 2540 (byte EVEX 0, bits [7:0]) - el primer byte (byte EVEX 0) es el campo de formato 2540 y contiene 0x62 (el valor único utilizado para distinguir el formato de instrucción compatible con vectores en una realización).

Los bytes segundo a cuarto (bytes EVEX 1 a 3) incluyen una serie de campos de bits que proporcionan una capacidad específica.

El campo REX 2605 (EVEX Byte 1, bits [7-5]) - consiste en un campo de bits EVEX.R (EVEX Byte 1, bit [7] - R), un campo de bits EVEX.X (EVEX byte 1, bit [6] - X), y 2557BEX byte 1, bit[5] - B). Los campos de bits EVEX.R, EVEX.X y EVEX.B proporcionan la misma funcionalidad que los campos de bits VEX correspondientes y se codifican en forma de complemento a 1, es decir, ZMM0 se codifica como 1111B, ZMM15 se codifica como 0000B. Otros campos de las instrucciones codifican los tres bits inferiores de los índices de registro como es conocido en la técnica (rrr, xxx y bbb), de modo que Rrrr, Xxxx y Bbbb pueden formarse sumando EVEX.R, EVEX.X, y EVEX.B.

Campo REX' 2510 - esta es la primera parte del campo REX' 2510 y es el campo de bit EVEX.R' (byte EVEX 1, bit [4] - R') que se utiliza para codificar los 16 superiores o los 16 inferiores del conjunto extendido de 32 registros. En una realización, este bit, junto con otros como se indica a continuación, se almacena en formato de bits invertido para distinguirlo (en el conocido modo x86 de 32 bits) de la instrucción BOUND, cuyo byte de opcode real es 62, pero no acepta en el campo MOD R/M (descrito a continuación) el valor de 11 en el campo MOD; realizaciones alternativas no almacenan esto y los otros bits indicados a continuación en el formato invertido. Se usa un valor de 1 para codificar los 16 registros inferiores. En otras palabras, R'Rrrr se forma combinando EVEX.R', EVEX.R y el otro RRR de otros campos.

Campo de mapa de opcode 2615 (byte EVEX 1, bits [3:0] - mmmm) - su contenido codifica un byte de opcode principal implícito (0F, 0F 38 o 0F 3).

El campo de ancho de elemento de datos 2564 (byte EVEX 2, bit [7] - W) está representado por la notación EVEX.W. EVEX.W se usa para definir la granularidad (tamaño) del tipo de datos (ya sean elementos de datos de 32 bits o elementos de datos de 64 bits).

EVEX.vvvv 2620 (EVEX Byte 2, bits [6:3]-vvvvv)- la función de EVEX.vvvv puede incluir lo siguiente: 1) EVEX.vvvv codifica el primer operando de registro de origen, especificado en forma invertida (complemento a 1) y es válido para instrucciones con 2 o más operandos de origen; 2) EVEX.vvvv codifica el operando de registro de destino, especificado en forma de complemento a 1 para ciertos desplazamientos de vector; o 3) EVEX.vvvv no codifica ningún operando, el campo está reservado y debe contener 1111b. Por lo tanto, el campo EVEX.vvvv 2620 codifica los 4 bits de orden inferior del primer especificador de registro de origen almacenado en forma invertida (complemento a 1). Dependiendo de la instrucción, se utiliza un campo de bits EVEX diferente adicional para ampliar el tamaño del especificador a 32 registros.

EVEX.U 2568 Campo de clase (byte EVEX 2, bit [2]-U) - Si EVEX.U = 0, indica clase A o EVEX.U0; si EVEX.U = 1, indica clase B o EVEX.U1.

Campo de codificación de prefijo 2625 (byte EVEX 2, bits [1:0]-pp) - proporciona bits adicionales para el campo de operación base. Además de proporcionar soporte para las instrucciones SSE heredadas en el formato de prefijo EVEX, esto también tiene el beneficio de compactar el prefijo SIMD (en lugar de requerir un byte para expresar el prefijo SIMD, el prefijo EVEX requiere solo 2 bits). En una realización, para soportar instrucciones SSE heredadas que usan un prefijo de SIMD (66H, F2H, F3H) tanto en el formato heredado como en el formato de prefijo EVEX, estos prefijos de SIMD heredados se codifican en el campo de codificación de prefijo de SIMD; y en el tiempo de ejecución se expanden en el prefijo de SIMD heredado antes de proporcionarse a la PLA del decodificador (para que la PLA pueda ejecutar tanto el formato heredado como el EVEX de estas instrucciones heredadas sin modificaciones). Aunque las instrucciones más nuevas podrían usar el contenido del campo de codificación del prefijo EVEX directamente como una extensión de opcode, ciertas realizaciones se expanden de manera similar para lograr coherencia, pero permiten que estos prefijos SIMD heredados especifiquen significados diferentes. Una realización alternativa puede rediseñar el PLA para soportar las codificaciones de prefijo SIMD de 2 bits y, por lo tanto, no requerir la expansión.

Campo alfa 2552 (byte EVEX 3, bit [7] - EH; también conocido como EVEX.EH, EVEX.rs, EVEX.RL, EVEX.write mask control y EVEX.N; también ilustrado con α): como se describió anteriormente, este campo es específico del contexto.

Campo beta 2554 (byte EVEX 3, bits [6:4]-SSS, también conocido como EVEX.s_{2-0}, EVEX.r_{2-0}, EVEX.rr1, EVEX.LL0, EVEX.LLB; también ilustrado con βββ): como se describió anteriormente, este campo es específico del contexto.

Campo REX' 2510 - este es el resto del campo REX' y es el campo de bit EVEX.V' (byte EVEX 3, bit [3] - V') que puede usarse para codificar los 16 superiores o los 16 inferiores del conjunto extendido de 32 registros. Este bit se almacena en formato de bits invertido. Se usa un valor de 1 para codificar los 16 registros inferiores. En otras palabras, V'VVVV se forma combinando EVEX.V', EVEX.vvvv.

Campo de máscara de escritura 2570 (byte EVEX 3, bits [2:0]-kkk) - su contenido especifica el índice de un registro en los registros de máscara de escritura como se describió anteriormente. En una realización, el valor específico EVEX.kkk=000 tiene un comportamiento especial que implica que no se utiliza una máscara de escritura para la instrucción particular (esto se puede implementar de diversas maneras, incluido el uso de una máscara de escritura cableada a todos aquello o al hardware que omite el hardware de enmascaramiento).

El campo opcode real 2630 (byte 4) también se conoce como byte de opcode. Parte del opcode se especifica en este campo.

El campo MOD R/M 2640 (byte 5) incluye el campo MOD 2642, el campo Reg 2644 y el campo R/M 2646. Como se describió anteriormente, el contenido del campo MOD 2642 distingue entre operaciones de acceso a memoria y operaciones sin acceso a memoria. La función del campo Reg 2644 se puede resumir en dos situaciones: codificar el operando del registro de destino o un operando del registro de origen o tratarlo como una extensión de opcode y no usarse para codificar un operando de instrucción. La función del campo R/M 2646 puede incluir lo siguiente: codificar el operando de instrucción que hace referencia a una dirección de memoria o codificar el operando del registro de destino o un operando del registro de origen.

Escala, índice, base (SIB) Byte (Byte 6) - como se describió anteriormente, el contenido de SIB 2650 se utiliza para la generación de direcciones de memoria. SIB.xxx 2654 y SIB.bbb 2656 - el contenido de estos campos se ha mencionado anteriormente con respecto a los índices de registro Xxxx y Bbbb.

Campo de desplazamiento 2562A (Bytes 7-10) - cuando el campo MOD 2642 contiene 10, los bytes 7-10 son el campo de desplazamiento 2562A, y funciona igual que el desplazamiento de 32 bits heredado (disp32) y funciona con granularidad de byte.

Campo de factor de desplazamiento 2562B (Byte 7) - cuando el campo MOD 2642 contiene 01, el byte 7 es el campo de factor de desplazamiento 2562B. La ubicación de este campo es la misma que la del desplazamiento de 8 bits (disp8) del conjunto de instrucciones x86 heredado, que funciona con granularidad de bytes. Como disp8 es de signo extendido, solo puede abordar desplazamientos entre -128 y +127 bytes; en términos de líneas de caché de 64 bytes, disp8 usa 8 bits que pueden configurarse solo en cuatro valores realmente útiles: -128, - 64, 0 y 64; como a menudo se necesita un rango mayor, se usa disp32; sin embargo, disp32 requiere 4 bytes. A diferencia de disp8 y disp32, el campo de factor de desplazamiento 2562B es una reinterpretación de disp8; cuando se usa el campo de factor de desplazamiento 2562B, el desplazamiento real se determina por el contenido del campo de factor de desplazamiento multiplicado por el tamaño del acceso de operando de memoria (N). Este tipo de desplazamiento se denomina disp8*N. Esto reduce la longitud promedio de instrucción (se utiliza un solo byte para el desplazamiento pero con un rango mucho mayor). Este desplazamiento comprimido supone que el desplazamiento efectivo es múltiplo de la granularidad del acceso a memoria y, por lo tanto, no es necesario codificar los bits redundantes de orden inferior del desplazamiento de dirección. En otras palabras, el campo de factor de desplazamiento 2562B sustituye el desplazamiento de 8 bits del conjunto de instrucciones x86 heredado. De esta forma, el campo de factor de

desplazamiento 2562B se codifica de la misma manera que un desplazamiento de 8 bits del conjunto de instrucciones x86 (por lo que no hay cambios en las reglas de codificación ModRM/SIB) con la única excepción de que disp8 se sobrecarga a disp8*N. En otras palabras, no hay cambios en las reglas de codificación ni en las longitudes de codificación, sino solo en la interpretación del valor de desplazamiento por parte del hardware (que necesita escalar el desplazamiento por el tamaño del operando de memoria para obtener un desplazamiento de dirección byte por byte). El campo inmediato 2572 funciona como se ha descrito anteriormente.

CAMPO OPCODE COMPLETO

La **Figura 26B** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico 2600 que conforman el campo opcode completo 2574 de acuerdo con una realización. Específicamente, el campo opcode completo 2574 incluye el campo de formato 2540, el campo de operación base 2542 y el campo de ancho de elemento de datos (W) 2564. El campo de operación base 2542 incluye el campo de codificación de prefijo 2625, el campo de mapa de opcode 2615 y el campo opcode real 2630.

Campo de índice de registro

La **Figura 26C** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico 2600 que conforman el campo de índice de registro 2544 de acuerdo con una realización. Específicamente, el campo de índice de registro 2544 incluye el campo REX 2605, el campo REX' 2610, el campo MODR/M.reg 2644, el campo MODR/Mr/m 2646, el campo VVVV 2620, el campo xxx 2654 y el campo bbb 2656.

CAMPO DE OPERACIÓN DE AUMENTO

La **Figura 26D** es un diagrama de bloques que ilustra los campos del formato de instrucción compatible con vectores específico 2600 que conforman el campo de operación de aumento 2550 de acuerdo con una realización. Cuando el campo de clase (U) 2568 contiene 0, significa EVEX.U0 (clase A 2568A); cuando contiene 1, significa EVEX.U1 (clase B 2568B). Cuando U=0 y el campo MOD 2642 contiene 11 (lo que significa una operación sin acceso a memoria), el campo alfa 2552 (byte de EVEX 3, bit [7] - EH) se interpreta como el campo de rs 2552A. Cuando el campo rs 2552A contiene un 1 (redondeo 2552A.1), el campo beta 2554 (byte EVEX 3, bits [6:4]- SSS) se interpreta como el campo de control de redondeo 2554A. El campo de control de redondeo 2554A incluye un campo SAE de un bit 2556 y un campo de operación de redondeo de dos bits 2558. Cuando el campo rs 2552A contiene un 0 (transformación de datos 2552A.2), el campo beta 2554 (byte EVEX 3, bits [6:4]- SSS) se interpreta como un campo de transformación de datos de tres bits 2554B. Cuando U=0 y el campo MOD 2642 contiene 00, 01 o 10 (lo que significa una operación de acceso a memoria), el campo alfa 2552 (EVEX byte 3, bit [7] - EH) se interpreta como el campo de sugerencia de desalojo (EH) 2552B y el campo beta 2554 (EVEX byte 3, bits [6:4]-SSS) se interpreta como un campo de manipulación de datos de tres bits 2554C.

Cuando U=1, el campo alfa 2552 (byte de EVEX 3, bit [7] - EH) se interpreta como el campo de control de máscara de escritura (Z) 2552C. Cuando U=1 y el campo MOD 2642 contiene 11 (lo que significa una operación sin acceso a memoria), parte del campo beta 2554 (byte EVEX 3, bit [4] - So) se interpreta como el campo RL 2557A; cuando contiene un 1 (redondeo 2557A.1) el resto del campo beta 2554 (byte EVEX 3, bit [6-5] - S_{2-1}) se interpreta como el campo de operación de redondeo 2559A, mientras que cuando el campo RL 2557A contiene un 0 (VSIZE 2557A.2) el resto del campo beta 2554 (byte EVEX 3, bit [6-5] - S_{2-1}) se interpreta como el campo de longitud de vector 2559B (byte EVEX 3, bit [6-5] - L_{1-10}). Cuando U=1 y el campo MOD 2642 contiene 00, 01 o 10 (lo que significa una operación de acceso a memoria), el campo beta 2554 (byte de EVEX 3, bits [6:4] - SSS) se interpreta como el campo de longitud de vector 2559B (byte de EVEX 3, bit [6-5] - L_{1-0}) y el campo de radiodifusión 2557B (byte de EVEX 3, bit [4] - B).

ARQUITECTURA DE REGISTRO ILUSTRATIVA

La **Figura 27** es un diagrama de bloques de una arquitectura de registro 2700 de acuerdo con una realización. En la realización ilustrada, hay 32 registros vectoriales 2710 que tienen 512 bits de ancho; estos registros se referencian como zmm0 a zmm31. Los 256 bits de orden inferior de los 16 registros zmm inferiores se superponen a los registros ymm0-16. Los 128 bits de orden inferior de los 16 registros zmm inferiores (los 128 bits de orden inferior de los registros ymm) se superponen a los registros xmm0-15. El formato de instrucción compatible con vectores específico 2600 opera en estos archivos de registros superpuestos como se ilustra en las siguientes tablas.

Longitud del vector ajustable	Clase	Operaciones	Registros
	A (**Figura 25A**; U=0)	2510, 2515, 2525, 2530	registros zmm (la longitud de vector es de 64 bytes)
Plantillas de instrucciones que no incluyen el campo de longitud del vector 2559B	B (**Figura 25B**; U=1)	2512	registros zmm (la longitud de vector es de 64 bytes)

Longitud del vector ajustable	Clase	Operaciones	Registros
Plantillas de instrucciones que incluyen el campo de longitud de vector 2559B	B (**Figura 25B**; U=1)	2517, 2527	registros zmm, ymm o xmm (la longitud del vector es de 64 bytes, 32 bytes o 16 bytes) de acuerdo con el campo de longitud del vector 2559B

En otras palabras, el campo de longitud de vector 2559B selecciona entre una longitud máxima y una o más longitudes más cortas, donde cada una de dichas longitudes más cortas es la mitad de la longitud anterior; y las plantillas de instrucciones sin el campo de longitud de vector 2559B operan sobre la longitud de vector máxima. Además, en una realización, las plantillas de instrucciones de clase B del formato de instrucción compatible con vectores específico 2600 operan en datos de punto flotante de precisión simple o doble empaquetados o escalares y en datos enteros empaquetados o escalares. Las operaciones escalares son operaciones realizadas en la posición del elemento de datos de orden más bajo en un registro zmm/ymm/xmm; las posiciones de los elementos de datos de orden superior se dejan igual que antes de la instrucción o se ponen a cero dependiendo de la realización.

Registros de máscara de escritura 2715 - en la realización ilustrada, hay 8 registros de máscara de escritura (k0 a k7), cada uno de 64 bits de tamaño. En una realización alternativa, los registros de máscara de escritura 2715 tienen un tamaño de 16 bits. Como se describió anteriormente, en una realización, el registro de máscara de vector k0 no se puede utilizar como una máscara de escritura; cuando se utiliza la codificación que normalmente indicaría k0 para una máscara de escritura, selecciona una máscara de escritura cableada de 0xFFFF, deshabilitando efectivamente el enmascaramiento de escritura para esa instrucción.

Registros de propósito general 2725 - en la realización ilustrada, hay dieciséis registros de propósito general de 64 bits que se utilizan junto con los modos de direccionamiento x86 existentes para direccionar operandos de memoria. A estos registros se hace referencia con los nombres RAX, RBX, RCX, RDX, RBP, RSI, RDI, RSP y R8 a R15.

Archivo de registro de pila de punto flotante escalar (pila x87) 2745, en el que se encuentra asociado el archivo de registro plano de números enteros empaquetados MMX 2750 - en la realización ilustrada, la pila x87 es una pila de ocho elementos que se utiliza para realizar operaciones de punto flotante escalar en datos de punto flotante de 32/64/80 bits utilizando la extensión del conjunto de instrucciones x87; mientras que los registros MMX se utilizan para realizar operaciones en datos de números enteros empaquetados de 64 bits, así como para contener operandos para algunas operaciones realizadas entre los registros MMX y XMM.

Realizaciones alternativas pueden usar registros más anchos o más estrechos. Además, las realizaciones alternativas pueden utilizar más, menos o diferentes archivos de registro y registros.

Arquitecturas de núcleo, procesadores y arquitecturas informáticas ejemplares

Los núcleos de procesador se pueden implementar de diferentes maneras, para diferentes propósitos y en diferentes procesadores. Por ejemplo, las implementaciones de tales núcleos pueden incluir: 1) un núcleo ordenado de propósito general destinado a la computación de propósito general; 2) un núcleo desordenado de propósito general de alto rendimiento destinado a computación de propósito general; 3) un núcleo de propósito especial destinado principalmente a gráficos y/o computación científica (rendimiento). Las implementaciones de diferentes procesadores pueden incluir: 1) una CPU que incluye uno o más núcleos en orden de propósito general destinados a computación de propósito general y/o uno o más núcleos fuera de orden de propósito general destinados a computación de propósito general; y 2) un coprocesador que incluye uno o más núcleos de propósito especial destinados principalmente a gráficos y/o científica (de rendimiento). Estos diferentes procesadores dan lugar a diferentes arquitecturas de sistemas informáticos, que pueden incluir: 1) el coprocesador en un chip separado de la CPU; 2) el coprocesador en una matriz separada en el mismo paquete que una CPU; 3) el coprocesador en la misma matriz que una CPU (en cuyo caso, a dicho coprocesador a veces se lo denomina lógica de propósito especial, como gráficos integrados y/o lógica científica (de rendimiento), o como núcleos de propósito especial); y 4) un sistema en un chip que puede incluir en la misma matriz la CPU descrita (a veces denominada núcleo o núcleos de aplicación o procesador o procesadores de aplicación), el coprocesador descrito anteriormente y funcionalidad adicional. A continuación, se describen arquitecturas de núcleo ilustrativas, seguidas de descripciones de procesadores y arquitecturas informáticas ilustrativas.

ARQUITECTURAS DE NÚCLEO ILUSTRATIVAS

DIAGRAMA DE BLOQUES DEL NÚCLEO EN ORDEN Y FUERA DE ORDEN

La **Figura 28A** es un diagrama de bloques que ilustra tanto un canal en orden ejemplar al igual que un canal de ejecución/emisión fuera de orden, de cambio de nombre de registro ejemplar de acuerdo con realizaciones. La **Figura 28B** es un diagrama de bloques que ilustra tanto una realización ejemplar de un núcleo de arquitectura en orden al igual que un núcleo de arquitectura de ejecución/emisión fuera de orden y cambio de nombre de registro ejemplar que se incluirá en un procesador de acuerdo con realizaciones. Los cuadros con líneas continuas en las Figuras 28A-B

ilustran el canal en orden y el núcleo en orden, mientras que la adición opcional de los cuadros con líneas discontinuas ilustra el núcleo y el canal de emisión/ejecución fuera de orden de cambio de nombre de registro. Dado que el aspecto en orden es un subconjunto del aspecto fuera de orden, se describirá el aspecto fuera de orden.

En la **Figura 28A**, un canal de procesador 2800 incluye una etapa de obtención 2802, una etapa de decodificación de longitud 2804, una etapa de decodificación 2806, una etapa de asignación 2808, una etapa de cambio de nombre 2810, una etapa de planificación (también conocida como despacho o emisión) 2812, una etapa de lectura de registro/lectura de memoria 2814, una etapa de ejecución 2816, una etapa de escritura diferida/escritura de memoria 2818, una etapa de manejo de excepciones 2822 y una etapa de confirmación 2824.

La **Figura 28B** muestra el núcleo de procesador 2890 que incluye una unidad de extremo frontal 2830 acoplada a una unidad de motor de ejecución 2850, y ambas están acopladas a una unidad de memoria 2870. El núcleo 2890 puede ser un núcleo de cálculo de conjunto de instrucciones reducido (RISC), un núcleo de cálculo de conjunto de instrucciones complejo (CISC), un núcleo de palabras de instrucción muy largas (VLIW) o un tipo de núcleo híbrido o alternativo. Como otra opción más, el núcleo 2890 puede ser un núcleo de propósito especial, tal como, por ejemplo, un núcleo de red o comunicación, un motor de compresión, un núcleo de coprocesador, un núcleo de unidad de procesamiento de gráficos informáticos de propósito general (GPGPU), un núcleo de gráficos o similar.

La unidad de extremo frontal 2830 incluye una unidad de predicción de ramas 2832 acoplada a una unidad de caché de instrucciones 2834, que está acoplada a una memoria intermedia de búsqueda de traducción de instrucciones (TLB) 2836, que está acoplada a una unidad de obtención de instrucciones 2838, que está acoplada a una unidad de decodificación 2840. La unidad de decodificación 2840 (o decodificador) puede decodificar instrucciones y generar como salida una o más microoperaciones, puntos de entrada de microcódigo, microinstrucciones, otras instrucciones u otras señales de control, que se decodifican a partir de las instrucciones originales o que, de otro modo, reflejan o se derivan de ellas. La unidad de decodificación 2840 puede implementarse utilizando varios mecanismos diferentes. Los ejemplos de mecanismos adecuados incluyen, pero no se limitan a, tablas de búsqueda, implementaciones de hardware, matrices lógicas programables (PLA), memorias de solo lectura (ROM) de microcódigo, etc. En una realización, el núcleo 2890 incluye una ROM de microcódigo u otro medio que almacena microcódigo para ciertas macroinstrucciones (por ejemplo, en la unidad de decodificación 2840 o de otro modo dentro de la unidad de extremo frontal 2830). La unidad de decodificación 2840 está acoplada a una unidad de cambio de nombre/asignación 2852 en la unidad de motor de ejecución 2850.

La unidad de motor de ejecución 2850 incluye la unidad de cambio de nombre/asignación 2852 acoplada a una unidad de retiro 2854 y un conjunto de una o más unidades de planificación 2856. La o las unidades de planificador 2856 representan cualquier número de planificadores diferentes, incluidas estaciones de reserva, ventana de instrucción central, etc. La o las unidades de planificador 2856 están acopladas a la o las unidades de archivos de registro físico 2858. Cada una de las unidades de archivos de registro físico 2858 representa uno o más archivos de registro físico, de los cuales cada uno almacena uno o más tipos de datos diferentes, tales como entero escalar, punto flotante escalar, entero empaquetado, punto flotante empaquetado, entero vectorial, punto flotante vectorial, estado (por ejemplo, un puntero de instrucción que es la dirección de la siguiente instrucción que se ejecutará), etc. En una realización, la unidad de archivos de registro físico 2858 comprende una unidad de registros vectoriales, una unidad de registros de máscara de escritura y una unidad de registros escalares. Estas unidades de registro pueden proporcionar registros vectoriales arquitectónicos, registros de máscara de vector y registros de propósito general. La o las unidades de archivos de registro físico 2858 se superpone con la unidad de retiro 2854 para ilustrar varias formas en las que se puede implementar el cambio de nombre de registro y la ejecución fuera de orden (por ejemplo, utilizando una o más memorias intermedias de reordenamiento y un o unos archivos de registro de retiro; utilizando un o unos archivos futuros, una o más memorias intermedias de historial y un o unos archivos de registro de retiro; utilizando mapas de registro y un grupo de registros; etc.). La unidad de retiro 2854 y la o las unidades de archivos de registro físico 2858 están acopladas al o a los grupos de ejecución 2860. El o los grupos de ejecución 2860 incluyen un conjunto de una o más unidades de ejecución 2862 y un conjunto de una o más unidades de acceso a memoria 2864. Las unidades de ejecución 2862 pueden realizar diversas operaciones (por ejemplo, desplazamientos, suma, resta, multiplicación) y en diversos tipos de datos (por ejemplo, coma flotante escalar, entero empaquetado, coma flotante empaquetada, entero vectorial, coma flotante vectorial). Si bien algunas realizaciones pueden incluir varias unidades de ejecución dedicadas a funciones específicas o conjuntos de funciones, otras realizaciones pueden incluir solo una unidad de ejecución o múltiples unidades de ejecución que realizan todas las funciones. La o las unidades de planificador 2856, la o las unidades de archivos de registro físico 2858 y el o los grupos de ejecución 2860 se muestran como posiblemente plurales porque ciertas realizaciones crean canales separados para ciertos tipos de datos/operaciones (por ejemplo, un canal de enteros escalares, un canal de enteros empaquetados/de punto flotante empaquetados/de enteros vectoriales/de punto flotante vectoriales y/o un canal de acceso a memoria que tienen cada una su propia unidad de planificador, unidad de archivos de registro físico y/o grupo de ejecución - y en el caso de un canal de acceso a memoria separado, se implementan ciertas realizaciones en las que solo el grupo de ejecución de este canal tiene la o las unidades de acceso a memoria 2864). También debe entenderse que cuando se utilizan canales separados, uno o más de estos canales pueden estar fuera de orden en su emisión/ejecución y el resto en orden.

El conjunto de unidades de acceso a memoria 2864 está acoplado a la unidad de memoria 2870, que incluye una unidad TLB de datos 2872 acoplada a una unidad de caché de datos 2874 acoplada a una unidad de caché de nivel 2 (L2) 2876. En una realización ejemplar, las unidades de acceso a memoria 2864 pueden incluir una unidad de carga, una unidad de dirección de almacenamiento y una unidad de datos de almacenamiento, cada una de las cuales está acoplada a la unidad TLB de datos 2872 en la unidad de memoria 2870. La unidad de caché de instrucciones 2834 está acoplada además a una unidad de caché de nivel 2 (L2) 2876 en la unidad de memoria 2870. La unidad de caché L2 2876 está acoplada a uno o más niveles de caché y eventualmente a una memoria principal.

A modo de ejemplo, la arquitectura central de ejecución/emisión fuera de orden y cambio de nombre de registro ejemplar puede implementar el canal 2800 de la siguiente manera: 1) la unidad de obtención de instrucciones 2838 realiza las etapas de obtención y decodificación de longitud 2802 y 2804; 2) la unidad de decodificación 2840 realiza la etapa de decodificación 2806; 3) la unidad de renombramiento/asignación 2852 realiza la etapa de asignación 2808 y la etapa de renombramiento 2810; 4) la o las unidades de planificadores 2856 realizan la etapa de planificadores 2812; 5) la o las unidades de archivos de registro físico 2858 y la unidad de memoria 2870 realizan la etapa de lectura de registro/lectura de memoria 2814; el grupo de ejecución 2860 realiza la etapa de ejecución 2816; 6) la unidad de memoria 2870 y la o las unidades de archivos de registro físico 2858 realizan la etapa de escritura diferida/escritura de memoria 2818; 7) varias unidades pueden estar involucradas en la etapa de manejo de excepciones 2822; y 8) la unidad de retiro 2854 y la o las unidades de archivos de registro físico 2858 realizan la etapa de confirmación 2824.

El núcleo 2890 puede soportar uno o más conjuntos de instrucciones (por ejemplo, el conjunto de instrucciones x86 (con algunas extensiones que se han agregado con versiones más nuevas); el conjunto de instrucciones MIPS de MIPS Technologies de Sunnyvale, CA; el conjunto de instrucciones ARM (con extensiones adicionales opcionales como NEON) de ARM Holdings de Sunnyvale, CA), incluidas las instrucciones descritas en el presente documento. En una realización, el núcleo 2890 incluye lógica para soportar una extensión de conjunto de instrucciones de datos empaquetados (por ejemplo, AVX1, AVX2), permitiendo así que las operaciones utilizadas por muchas aplicaciones multimedia se realicen utilizando datos empaquetados.

Debe entenderse que el núcleo puede soportar multiprocesamiento (ejecución de dos o más conjuntos paralelos de operaciones o subprocesos) y puede hacerlo de diversas maneras, incluyendo multiprocesamiento en porciones de tiempo, multiprocesamiento simultáneo (donde un solo núcleo físico proporciona un núcleo lógico para cada uno de los subprocesos que ese núcleo físico está multiprocesando simultáneamente) o una combinación de estos (por ejemplo, obtención y decodificación en porciones de tiempo y multiprocesamiento simultáneo a partir de entonces, tal como en la tecnología Intel® Hyperthreading).

Si bien el cambio de nombre de registro se describe en el contexto de la ejecución fuera de orden, debe entenderse que el cambio de nombre de registro puede usarse en una arquitectura en orden. Si bien la realización ilustrada del procesador también incluye unidades de caché de instrucciones y datos 2834/2874 separadas y una unidad de caché L2 2876 compartida, realizaciones alternativas pueden tener una única caché interna tanto para instrucciones como para datos, como, por ejemplo, una caché interna de Nivel 1 (L1), o múltiples niveles de caché interna. En algunas realizaciones, el sistema puede incluir una combinación de una memoria caché interna y una memoria caché externa que es externa al núcleo y/o al procesador. Alternativamente, toda la memoria caché puede ser externa al núcleo y/o al procesador.

ARQUITECTURA DE NÚCLEO EN ORDEN ILUSTRATIVA ESPECÍFICA

Las **Figuras 29A-B** ilustran un diagrama de bloques de una arquitectura de núcleo en orden ejemplar más específica, cuyo núcleo sería uno de varios bloques lógicos (incluidos otros núcleos del mismo tipo y/o tipos diferentes) en un chip. Los bloques lógicos se comunican a través de una red de interconexión de gran ancho de banda (por ejemplo, una red de anillo) con cierta lógica de función fija, interfaces de E/S de memoria y otra lógica de E/S necesaria, de acuerdo con la aplicación.

La **Figura 29A** es un diagrama de bloques de un solo núcleo de procesador, junto con su conexión a la red de interconexión en chip 2902 y con su subconjunto local de la memoria caché de Nivel 2 (L2) 2904, de acuerdo con las realizaciones. En una realización, un decodificador de instrucciones 2900 soporta el conjunto de instrucciones x86 con una extensión de conjunto de instrucciones de datos empaquetados. Un caché L1 2906 permite accesos de baja latencia a la memoria caché en las unidades escalares y vectoriales. Si bien en una realización (para simplificar el diseño), una unidad escalar 2908 y una unidad vectorial 2910 usan conjuntos de registros separados (respectivamente, registros escalares 2912 y registros vectoriales 2914) y los datos transferidos entre ellos se escriben en la memoria y a continuación se vuelven a leer desde una caché de nivel 1 (L1) 2906, realizaciones alternativas pueden usar un enfoque diferente (por ejemplo, usar un único conjunto de registros o incluir una ruta de comunicaciones que permita que los datos se transfieran entre los dos archivos de registro sin tener que escribirse ni volverse a leer).

El subconjunto local de la caché de L2 2904 es parte de una caché de L2 global que se divide en subconjuntos locales separados, uno por núcleo de procesador. Cada núcleo de procesador tiene una ruta de acceso directo a su propio subconjunto local de la caché L2 2904. Los datos leídos por un núcleo de procesador se almacenan en su subconjunto de caché L2 2904 y se puede acceder a ellos rápidamente, en paralelo con otros núcleos de procesador que acceden

a sus propios subconjuntos de caché L2 locales. Los datos escritos por un núcleo de procesador se almacenan en su propio subconjunto de caché L2 2904 y se restablece de otros subconjuntos, si es necesario. La red en anillo garantiza la coherencia de los datos compartidos. La red de anillo es bidireccional para permitir que agentes, tales como núcleos de procesador, cachés L2 y otros bloques lógicos, se comuniquen entre sí dentro del chip. Cada ruta de datos del anillo tiene 1012 bits de ancho por dirección.

La **Figura 29B** es una vista ampliada de parte del núcleo de procesador en la **Figura 29A** de acuerdo con las realizaciones. La **Figura 29B** incluye una caché de datos L1 2906A parte de la caché L1 2904, así como más detalles con respecto a la unidad vectorial 2910 y los registros vectoriales 2914. Específicamente, la unidad vectorial 2910 es una unidad de procesamiento vectorial (VPU) de ancho 16 (véase la UAL 2928 16 ancho), que ejecuta una o más de instrucciones de números enteros, flotantes de precisión simple y flotantes de precisión doble. La VPU soporta la conversión de entradas de registro con la unidad de conversión 2920, la conversión numérica con las unidades de conversión numérica 2922A-B y la replicación con la unidad de replicación 2924 en la entrada de memoria. Los registros de máscara de escritura 2926 permiten predecir escrituras vectoriales resultantes.

La **Figura 30** es un diagrama de bloques de un procesador 3000 que puede tener más de un núcleo, puede tener un controlador de memoria integrado y puede tener gráficos integrados de acuerdo con realizaciones. Los cuadros de líneas continuas en la **Figura 30** ilustran un procesador 3000 con un solo núcleo 3002A, un agente de sistema 3010, un conjunto de una o más unidades de controlador de bus 3016, mientras que la adición opcional de los cuadros de líneas discontinuas ilustra un procesador alternativo 3000 con múltiples núcleos 3002A-N, un conjunto de una o más unidades de controlador de memoria integrado 3014 en la unidad de agente de sistema 3010 y una lógica de propósito especial 3008.

Por tanto, diferentes implementaciones del procesador 3000 pueden incluir: 1) una CPU con la lógica de propósito especial 3008 que es lógica gráfica y/o científica (rendimiento) integrada (que puede incluir uno o más núcleos), y los núcleos 3002A-N son uno o más núcleos de propósito general (por ejemplo, núcleos en orden de propósito general, núcleos fuera de orden de propósito general, una combinación de los dos); 2) un coprocesador con los núcleos 3002A-N con una gran cantidad de núcleos de propósito especial destinados principalmente a gráficos y/o científica (rendimiento); y 3) un coprocesador con los núcleos 3002A-N con una gran cantidad de núcleos en orden de propósito general. De este modo, el procesador 3000 puede ser un procesador de propósito general, un coprocesador o un procesador de propósito especial, tal como, por ejemplo, un procesador de red o de comunicaciones, un motor de compresión, un procesador de gráficos, una GPGPU (unidad de procesamiento de gráficos de propósito general), un coprocesador de muchos núcleos integrados (MIC) de alto rendimiento (incluyendo 30 o más núcleos), un procesador integrado o similar. El procesador puede implementarse en uno o más chips. El procesador 3000 puede ser parte y/o puede implementarse en uno o más sustratos utilizando cualquiera de varias tecnologías de proceso, tales como, por ejemplo, BiCMOS, CMOS o NMOS.

La jerarquía de memoria incluye uno o más niveles de caché dentro de los núcleos, un conjunto de una o más unidades de caché compartida 3006 y una memoria externa (no mostrada) acoplada al conjunto de unidades de controlador de memoria integrado 3014. El conjunto de unidades de caché compartida 3006 puede incluir una o más cachés de nivel medio, como el nivel 2 (L2), el nivel 3 (L3), el nivel 4 (L4) u otros niveles de caché, una caché de último nivel (LLC) y/o combinaciones de las mismas. Si bien en una realización una unidad de interconexión basada en anillo 3012 interconecta la lógica de propósito especial 3008 (la lógica de gráficos integrados es un ejemplo de y también se la denomina en el presente documento como lógica de propósito especial), el conjunto de unidades de caché compartida 3006 y la unidad de agente de sistema 3010/unidad o unidades de controlador de memoria integrado 3014, realizaciones alternativas pueden utilizar cualquier número de técnicas bien conocidas para interconectar dichas unidades. En una realización, se mantiene la coherencia entre una o más unidades de caché 3006 y núcleos 3002A-N.

En algunas realizaciones, uno o más de los núcleos 3002A-N son capaces de realizar multiprocesamiento. El agente de sistema 3010 incluye aquellos componentes que coordinan y operan los núcleos 3002A-N. La unidad de agente de sistema 3010 puede incluir, por ejemplo, una unidad de control de energía (PCU) y una unidad de visualización. La PCU puede ser o incluir lógica y componentes necesarios para regular el estado de energía de los núcleos 3002A-N y la lógica de propósito especial 3008. La unidad de visualización es para controlar una o más pantallas conectadas externamente.

Los núcleos 3002A-N pueden ser homogéneos o heterogéneos en términos de conjunto de instrucciones de arquitectura; es decir, dos o más de los núcleos 3002A-N pueden ser capaces de ejecutar el mismo conjunto de instrucciones, mientras que otros pueden ser capaces de ejecutar solo un subconjunto de ese conjunto de instrucciones o un conjunto de instrucciones diferente.

ARQUITECTURAS INFORMÁTICAS ILUSTRATIVAS

Las **Figuras 31-34** son diagramas de bloques de arquitecturas informáticas ilustrativas. Otros diseños y configuraciones de sistemas conocidos en las técnicas para computadoras portátiles, de escritorio, PC de mano, asistentes digitales personales, estaciones de trabajo de ingeniería, servidores, dispositivos de red, concentradores

de red, conmutadores, procesadores integrados, procesadores de señales digitales (DSP), dispositivos gráficos, dispositivos de videojuegos, decodificadores, microcontroladores, teléfonos móviles, reproductores multimedia portátiles, dispositivos de mano y varios otros dispositivos electrónicos también son adecuados. En general, son generalmente adecuados una gran variedad de sistemas o dispositivos electrónicos capaces de incorporar un procesador y/u otra lógica de ejecución como la descrita en el presente documento.

Haciendo referencia ahora a la **Figura 31**, se muestra un diagrama de bloques de un sistema 3100 de acuerdo con una realización de la presente invención. El sistema 3100 puede incluir uno o más procesadores 3110, 3115, que están acoplados a un concentrador controlador 3120. En una realización, el concentrador controlador 3120 incluye un concentrador controlador de memoria gráfica (GMCH) 3190 y un concentrador de entrada/salida (IOH) 3150 (que pueden estar en chips separados); el GMCH 3190 incluye controladores de memoria y gráficos a los que están acoplados la memoria 3140 y un coprocesador 3145; el IOH 3150 acopla dispositivos de entrada/salida (E/S) 3160 al GMCH 3190. Alternativamente, uno o ambos controladores de memoria y gráficos están integrados dentro del procesador (como se describe en el presente documento), la memoria 3140 y el coprocesador 3145 están acoplados directamente al procesador 3110 y el concentrador controlador 3120 en un solo chip con el IOH 3150.

La naturaleza opcional de los procesadores adicionales 3115 se indica en la **Figura 31** con líneas discontinuas. Cada procesador 3110, 3115 puede incluir uno o más de los núcleos de procesamiento descritos en el presente documento y puede ser alguna versión del procesador 3000.

La memoria 3140 puede ser, por ejemplo, una memoria dinámica de acceso aleatorio (DRAM), una memoria de cambio de fase (PCM) o una combinación de ambas. Para al menos una realización, el concentrador de controlador 3120 se comunica con el o los procesadores 3110, 3115 a través de un bus multipunto, tal como un bus frontal (FSB), una interfaz punto a punto tal como QuickPath Interconnect (QPI), o conexión similar 3195.

En una realización, el coprocesador 3145 es un procesador de propósito especial, tal como, por ejemplo, un procesador MIC de alto rendimiento, un procesador de red o comunicación, un motor de compresión, un procesador de gráficos, una GPGPU, un procesador integrado o similar. En una realización, el concentrador controlador 3120 puede incluir un acelerador de gráficos integrado.

Puede haber varias diferencias entre los recursos físicos 3110, 3115 en lo que respecta a un espectro de métricas de mérito, que incluyen características arquitectónicas, microarquitectónicas, térmicas, de consumo de energía y similares.

En una realización, el procesador 3110 ejecuta instrucciones que controlan operaciones de procesamiento de datos de un tipo general. Dentro de las instrucciones pueden estar incorporadas instrucciones de coprocesador. El procesador 3110 reconoce estas instrucciones de coprocesador como de un tipo que debería ser ejecutado por el coprocesador adjunto 3145. En consecuencia, el procesador 3110 emite estas instrucciones de coprocesador (o señales de control que representan instrucciones de coprocesador) en un bus de coprocesador u otra interconexión, al coprocesador 3145. El o los coprocesadores 3145 aceptan y ejecutan las instrucciones de coprocesador recibidas.

Con referencia ahora a la **Figura 32**, se muestra un diagrama de bloques de un primer sistema 3200 ilustrativo más específico de acuerdo con una realización de la presente invención. Como se muestra en la **Figura 32**, el sistema multiprocesador 3200 es un sistema de interconexión punto a punto e incluye un primer procesador 3270 y un segundo procesador 3280 acoplados a través de una interconexión punto a punto 3250. Cada uno de los procesadores 3270 y 3280 puede ser alguna versión del procesador 3000. En una realización, los procesadores 3270 y 3280 son respectivamente los procesadores 3110 y 3115, mientras que el coprocesador 3238 es el coprocesador 3145. En otra realización, los procesadores 3270 y 3280 son respectivamente el procesador 3110 y el coprocesador 3145.

Se muestran los procesadores 3270 y 3280 incluyendo las unidades de controlador de memoria integrado (IMC) 3272 y 3282, respectivamente. El procesador 3270 también incluye como parte de sus unidades controladoras de bus las interfaces punto a punto (P-P) 3276 y 3278; de manera similar, el segundo procesador 3280 incluye las interfaces P-P 3286 y 3288. Los procesadores 3270, 3280 pueden intercambiar información a través de una interfaz punto a punto (P-P) 3250 utilizando circuitos de interfaz P-P 3278, 3288. Como se muestra en la **Figura 32**, los IMC 3272 y 3282 acoplan los procesadores a las respectivas memorias, en concreto, una memoria 3232 y una memoria 3234, que pueden ser porciones de la memoria principal conectadas localmente a los respectivos procesadores.

Los procesadores 3270, 3280 pueden intercambiar cada uno información con un conjunto de chips 3290 a través de interfaces P-P 3252, 3254 individuales utilizando circuitos de interfaz punto a punto 3276, 3294, 3286, 3298. El conjunto de chips 3290 puede intercambiar opcionalmente información con el coprocesador 3238 a través de una interfaz de alto rendimiento 3292. En una realización, el coprocesador 3238 es un procesador de propósito especial, tal como, por ejemplo, un procesador MIC de alto rendimiento, un procesador de red o comunicación, un motor de compresión, un procesador de gráficos, GPGPU, un procesador integrado o similar.

Se puede incluir una memoria caché compartida (no mostrada) en uno cualquiera de los procesadores o fuera de ambos procesadores, pero conectada con los procesadores por medio de una interconexión P-P, de tal forma que la

información de memoria caché local de uno cualquiera de los procesadores, o de ambos, se pueda almacenar en la memoria caché compartida si un procesador pasa a un modo de bajo consumo.

El conjunto de chips 3290 se puede acoplar a un primer bus 3216 a través de una interfaz 3296. En una realización, el primer bus 3216 puede ser un bus de interconexión de componentes periféricos (PCI), o un bus tal como un bus PCI Express u otro bus de interconexión de E/S de tercera generación, aunque el alcance de la presente invención no está limitado de esa manera.

Como se muestra en la **Figura 32**, varios dispositivos de E/S 3214 pueden estar acoplados al primer bus 3216, junto con un puente de bus 3218 que acopla el primer bus 3216 a un segundo bus 3220. En una realización, uno o más procesadores 3215 adicionales, tales como coprocesadores, procesadores MIC de alto rendimiento, GPGPU, aceleradores (tales como, por ejemplo, aceleradores de gráficos o unidades de procesamiento de señal digital (DSP)), matrices de puertas programables en campo o cualquier otro procesador, están acoplados al primer bus 3216. En una realización, el segundo bus 3220 puede ser un bus de bajo número de pines (LPC). Se pueden acoplar diversos dispositivos a un segundo bus 3220, incluidos, por ejemplo, un teclado y/o un ratón 3222, dispositivos de comunicaciones 3227 y una unidad de almacenamiento 3228, tal como una unidad de disco u otro dispositivo de almacenamiento masivo que puede incluir instrucciones/códigos y datos 3230, en una realización. Además, se puede acoplar una E/S de audio 3224 al segundo bus 3220. Obsérvese que son posibles otras arquitecturas. Por ejemplo, en lugar de la arquitectura punto a punto de la **Figura** 32, un sistema puede implementar un bus multipunto u otra arquitectura similar.

Con referencia ahora a la **Figura 33**, se muestra un diagrama de bloques de un segundo sistema 3300 ilustrativo más específico de acuerdo con una realización de la presente invención. Los elementos similares en las **Figuras 32 y 33** llevan números de referencia similares, y ciertos aspectos de la **Figura 32** se han omitido de la **Figura 33** para evitar enmascarar otros aspectos de la **Figura 33**.

La Figura 33 ilustra que los procesadores 3270, 3280 pueden incluir memoria integrada y lógica de control de E/S ("CL") 3372 y 3382, respectivamente. Así, los CL 3372, 3382 incluyen unidades controladoras de memoria integradas e incluyen lógica de control de E/S. La **Figura 33** ilustra que no solo las memorias 3232, 3234 están acopladas a la CL 3372, 3382, sino que también los dispositivos de E/S 3314 también están acoplados a la lógica de control 3372, 3382. Los dispositivos de E/S heredados 3315 están acoplados al conjunto de chips 3290.

Con referencia ahora a la **Figura 34**, se muestra un diagrama de bloques de un SoC 3400 de acuerdo con una realización de la presente invención. Los elementos similares en la **Figura 30** llevan números de referencia iguales. Además, los recuadros con línea discontinua son características opcionales en los SoC más avanzados. En la **Figura 34**, una o más unidades de interconexión 3402 están acopladas a: un procesador de aplicaciones 3410 que incluye un conjunto de uno o más núcleos 3002A-N, que incluyen unidades de caché 3004A-N y unidades de caché compartidas 3006; una unidad de agente de sistema 3010; una o más unidades de controlador de bus 3016; una o más unidades de controlador de memoria integrado 3014; un conjunto de uno o más coprocesadores 3420 que pueden incluir lógica gráfica integrada, un procesador de imágenes, un procesador de audio y un procesador de vídeo; una unidad de memoria de acceso aleatorio estático (SRAM) 3430; una unidad de acceso directo a memoria (DMA) 3432; y una unidad de visualización 3440 para acoplarse a una o más pantallas externas. En una realización, el o los coprocesadores 3420 incluyen un procesador de propósito especial, tal como, por ejemplo, un procesador de red o de comunicaciones, un motor de compresión, GPGPU, un procesador MIC de alto rendimiento, un procesador integrado o similar.

Las realizaciones de los mecanismos dados a conocer en el presente documento pueden implementarse en hardware, software, firmware o una combinación de dichos enfoques de implementación. Las realizaciones pueden implementarse como programas informáticos o código de programa que se ejecuta en sistemas programables que comprenden al menos un procesador, un sistema de almacenamiento (incluyendo memoria volátil y no volátil y/o elementos de almacenamiento), al menos un dispositivo de entrada y al menos un dispositivo de salida.

El código de programa, tal como el código 3230 ilustrado en la **Figura 32**, se puede aplicar a las instrucciones de entrada para realizar las funciones descritas en el presente documento y generar información de salida. La información de salida se puede aplicar a uno o más dispositivos de salida, de manera conocida. Para los fines de esta solicitud, un sistema de procesamiento incluye cualquier sistema que tenga un procesador, tal como, por ejemplo; un procesador de señal digital (DSP), un microcontrolador, un circuito integrado de aplicación específica (ASIC) o un microprocesador.

El código del programa puede implementarse en un lenguaje de programación orientado a objetos o procedimental de alto nivel para comunicarse con un sistema de procesamiento. El código del programa también puede implementarse en lenguaje ensamblador o lenguaje máquina, si se desea. De hecho, los mecanismos descritos en el presente documento no están limitados en su alcance a ningún lenguaje de programación particular. En cualquier caso, el lenguaje puede ser un lenguaje compilado o interpretado.

Uno o más aspectos de al menos una realización pueden implementarse mediante instrucciones representativas almacenadas en un medio legible por máquina que representa diversas lógicas dentro del procesador, que cuando

son leídas por una máquina hacen que la máquina fabrique lógica para realizar las técnicas descritas en el presente documento. Estas representaciones, conocidas como "núcleos IP", pueden almacenarse en un medio tangible, legible por máquina y suministrarse a diversos clientes o instalaciones de fabricación para cargarlas en las máquinas de fabricación que realmente producen la lógica o el procesador.

Dichos medios de almacenamiento legibles por máquina pueden incluir, sin limitación, disposiciones tangibles no transitorias de artículos fabricados o formados por una máquina o dispositivo, incluidos medios de almacenamiento tales como discos duros, cualquier otro tipo de disco, incluidos disquetes, discos ópticos, memorias de solo lectura de discos compactos (CD-ROM), discos compactos regrabables (CD-RW) y discos magnetoópticos, dispositivos semiconductores tales como memorias de solo lectura (ROM), memorias de acceso aleatorio (RAM) tales como memorias de acceso aleatorio dinámico (DRAM), memorias de acceso aleatorio estático (SRAM), memorias de solo lectura programables y borrables (EPROM), memorias flash, memorias de solo lectura programables y borrables eléctricamente (EEPROM), memoria de cambio de fase (PCM), tarjetas magnéticas u ópticas o cualquier otro tipo de medio adecuado para almacenar instrucciones electrónicas.

En consecuencia, las realizaciones también incluyen medios tangibles legibles por máquina, no transitorios, que contienen instrucciones o que contienen datos de diseño, tales como el lenguaje de descripción de hardware (HDL), que define estructuras, circuitos, aparatos, procesadores y/o características del sistema descritos en el presente documento. Estas realizaciones también pueden denominarse productos de programa.

EMULACIÓN (INCLUYENDO TRADUCCIÓN BINARIA, TRANSFORMACIÓN DE CÓDIGO, ETC.)

En algunos casos, se puede usar un convertidor de instrucciones para convertir una instrucción de un conjunto de instrucciones de origen a un conjunto de instrucciones objetivo. Por ejemplo, el convertidor de instrucciones puede traducir (por ejemplo, utilizando traducción binaria estática, traducción binaria dinámica incluyendo compilación dinámica), transformar, emular o convertir de otro modo una instrucción en una o más instrucciones diferentes para que sean procesadas por el núcleo. El convertidor de instrucciones puede implementarse en software, hardware, firmware o una combinación de ellos. El convertidor de instrucciones puede estar en el procesador, fuera del procesador o parcialmente dentro y parcialmente fuera del procesador.

La **Figura 35** es un diagrama de bloques que contrasta el uso de un convertidor de instrucciones de software para convertir instrucciones binarias en un conjunto de instrucciones de origen a instrucciones binarias en un conjunto de instrucciones de destino de acuerdo con realizaciones. En la realización ilustrada, el convertidor de instrucciones es un convertidor de instrucciones de software, aunque alternativamente el convertidor de instrucciones puede implementarse en software, firmware, hardware o varias combinaciones de los mismos. La **Figura 35** muestra un programa en un lenguaje de alto nivel 3502 que puede compilarse usando un compilador x86 3504 para generar código binario x86 3506 que puede ejecutarse de forma nativa mediante un procesador con al menos un núcleo de conjunto de instrucciones x86 3516. El procesador con al menos un núcleo de conjunto de instrucciones x86 3516 representa cualquier procesador que puede realizar sustancialmente las mismas funciones que un procesador Intel con al menos un núcleo de conjunto de instrucciones x86 ejecutando de manera compatible o procesando de otro modo (1) una parte sustancial del conjunto de instrucciones del núcleo de conjunto de instrucciones x86 de Intel o (2) versiones de código objeto de aplicaciones u otro software destinado a ejecutarse en un procesador Intel con al menos un núcleo de conjunto de instrucciones x86, con el fin de lograr sustancialmente el mismo resultado que un procesador Intel con al menos un núcleo de conjunto de instrucciones x86. El compilador x86 3504 representa un compilador que puede funcionar para generar código binario x86 3506 (por ejemplo, código objeto) que puede, con o sin procesamiento de vinculación adicional, ejecutarse en el procesador con al menos un núcleo de conjunto de instrucciones x86 3516. De manera similar, la **Figura 35** muestra que el programa en el lenguaje de alto nivel 3502 puede compilarse utilizando un compilador de conjunto de instrucciones alternativo 3508 para generar un código binario de conjunto de instrucciones alternativo 3510 que puede ejecutarse de forma nativa por un procesador sin al menos un núcleo de conjunto de instrucciones x86 3514 (por ejemplo, un procesador con núcleos que ejecutan el conjunto de instrucciones MIPS de MIPS Technologies de Sunnyvale, CA y/o que ejecutan el conjunto de instrucciones ARM de ARM Holdings de Sunnyvale, CA). El convertidor de instrucciones 3512 se usa para convertir el código binario x86 3506 en un código que el procesador puede ejecutar de forma nativa sin un núcleo de conjunto de instrucciones x86 3514. No es probable que este código convertido sea el mismo que el código binario de conjunto de instrucciones alternativo 3510 porque es difícil crear un convertidor de instrucciones capaz de hacer esto; sin embargo, el código convertido realizará la operación general y estará compuesto por instrucciones de conjunto de instrucciones alternativo. Así, el convertidor de instrucciones 3512 representa un software, firmware, hardware o una combinación de los mismos que, mediante emulación, simulación o cualquier otro proceso, permite a un procesador u otro dispositivo electrónico que no disponga de un procesador o núcleo de conjunto de instrucciones x86 ejecutar el código binario x86 3506.

REIVINDICACIONES

1. Una unidad de procesamiento que comprende:
circuito de obtención para obtener una instrucción;
circuito de decodificación para decodificar la instrucción, teniendo la instrucción un opcode, un primer campo para especificar una primera ubicación de almacenamiento de una pluralidad de elementos de datos correspondientes a una primera matriz que tiene M filas por N columnas de elementos de datos de punto flotante de precisión simple de 32 bits, un segundo campo para especificar una segunda ubicación de almacenamiento de una pluralidad de elementos de datos correspondientes a una segunda matriz que tiene M filas por K columnas de elementos de datos de punto flotante de 16 bits que tienen un formato bfloat16, y un tercer campo para especificar una tercera ubicación de almacenamiento de una pluralidad de elementos de datos correspondientes a una tercera matriz que tiene K filas por N columnas de elementos de datos de punto flotante de 16 bits que tienen el formato bfloat16; y
circuito de ejecución acoplado al circuito de decodificación, el circuito de ejecución para realizar operaciones correspondientes a la instrucción, para cada fila m de las M filas de la segunda matriz y para cada columna n de las N columnas de la tercera matriz, para:
generar un producto escalar a partir de K elementos de datos de punto flotante de 16 bits correspondientes a la fila m de la segunda matriz y K elementos de datos de punto flotante de 16 bits correspondientes a la columna n de la tercera matriz;
acumular el producto escalar con un elemento de datos de punto flotante de precisión simple de 32 bits correspondiente a una fila m de las M filas y correspondiente a una columna n de las N columnas, de la primera matriz para generar un elemento de datos de punto flotante de precisión simple de 32 bits como resultado; y
almacenar el elemento de datos de punto flotante de precisión simple de 32 bits resultante en una posición de la primera ubicación de almacenamiento correspondiente a la fila m y la columna n de la primera matriz,
en donde el circuito de ejecución debe realizar una operación correspondiente a la instrucción de procesar valores desnormalizados de las matrices segunda y tercera como cero.

2. La unidad de procesamiento de la reivindicación 1, que comprende además un registro de control para especificar un modo de redondeo, en donde la generación del producto escalar y la acumulación de los productos escalares incluyen cada una la aplicación de un único modo de redondeo independientemente del modo de redondeo especificado por el registro de control.

3. La unidad de procesamiento de la reivindicación 2, en donde la unidad de procesamiento no debe consultar el registro de control debido a la instrucción.

4. La unidad de procesamiento de cualquiera de las reivindicaciones 1 a 3, en donde la unidad de procesamiento no debe actualizar un registro de control debido a la instrucción.

5. La unidad de procesamiento de cualquiera de las reivindicaciones 1 a 4, en donde la unidad de procesamiento no debe denotar excepciones debido a la instrucción.

6. La unidad de procesamiento de cualquiera de las reivindicaciones 1 a 5, en donde el circuito de ejecución sirve para realizar una operación correspondiente a la instrucción de restablecer a cero los valores desnormalizados.

7. La unidad de procesamiento de cualquiera de las reivindicaciones 1 a 6, en donde las M filas de la segunda matriz y las N columnas de la tercera matriz son iguales en número.

8. La unidad de procesamiento de cualquiera de las reivindicaciones 1 a 7, en donde las ubicaciones de almacenamiento primera, segunda y tercera están en registros vectoriales de 128 bits.

9. La unidad de procesamiento de cualquiera de las reivindicaciones 1 a 8, en donde cada producto escalar es un producto escalar de punto flotante de precisión simple de 32 bits.

10. La unidad de procesamiento de cualquiera de las reivindicaciones 1 a 9, que comprende además:
un circuito de predicción de ramas;
un circuito de cambio de nombre de registro; y
circuito planificador para planificar la instrucción decodificada para su ejecución.

11. La unidad de procesamiento de acuerdo con cualquiera de las reivindicaciones 1 a 10, en donde la unidad de procesamiento es un núcleo de unidad central de procesamiento de propósito general, CPU.

12. La unidad de procesamiento de acuerdo con cualquiera de las reivindicaciones 1 a 11, en donde la unidad de procesamiento es un núcleo de computación de conjunto de instrucciones reducido, RISC.

13. La unidad de procesamiento de la reivindicación 2, en donde el modo de redondeo único es fijo para la instrucción.

14. Un sistema en un chip, SoC, que comprende:

un controlador de memoria; y
Un núcleo de unidad central de procesamiento de propósito general, CPU, acoplado con el controlador de memoria, siendo el núcleo de CPU de propósito general la unidad de procesamiento de cualquiera de las reivindicaciones 1 a 13.

15. El SoC de la reivindicación 14, que comprende además uno o más de un procesador de red acoplado con el núcleo de CPU de propósito general, un coprocesador acoplado con el núcleo de CPU de propósito general y un procesador de imágenes acoplado con el núcleo de CPU de propósito general.

16. Un sistema, que comprende:
una memoria; y
un núcleo de unidad central de procesamiento de propósito general, CPU, acoplado con la memoria, siendo el núcleo de CPU de propósito general la unidad de procesamiento de cualquiera de las reivindicaciones 1 a 13.

17. El sistema de la reivindicación 16, que comprende además uno o más de un dispositivo de almacenamiento masivo acoplado al núcleo de CPU de propósito general, un bus de interconexión de componentes periféricos, PCI Express, acoplado al núcleo de CPU de propósito general, un dispositivo de comunicaciones acoplado al núcleo de CPU de propósito general.

18. Un método para procesar datos en una unidad de procesamiento, que comprende:
obtener una instrucción;
decodificar la instrucción, teniendo la instrucción un opcode, un primer campo que especifica una primera ubicación de almacenamiento de una pluralidad de elementos de datos correspondientes a una primera matriz que tiene M filas por N columnas de elementos de datos de punto flotante de precisión simple de 32 bits, un segundo campo que especifica una segunda ubicación de almacenamiento de una pluralidad de elementos de datos correspondientes a una segunda matriz que tiene M filas por K columnas de elementos de datos de punto flotante de 16 bits que tienen un formato bfloat16, y un tercer campo que especifica una tercera ubicación de almacenamiento de una pluralidad de elementos de datos correspondientes a una tercera matriz que tiene K filas por N columnas de elementos de datos de punto flotante de 16 bits que tienen el formato bfloat16;y
realizar operaciones correspondientes a la instrucción que incluyen, para cada fila m de las M filas de la segunda matriz, y para cada columna n de las N columnas de la tercera matriz:
generar un producto escalar a partir de K elementos de datos de punto flotante de 16 bits correspondientes a la fila m de la segunda matriz y K elementos de datos de punto flotante de 16 bits correspondientes a la columna n de la tercera matriz;
acumular el producto escalar con un elemento de datos de punto flotante de precisión simple de 32 bits correspondiente a una fila m de las M filas, y correspondiente a una columna n de las N columnas, de la primera matriz; y
almacenar el elemento de datos de punto flotante de precisión simple de 32 bits resultante en una posición de la primera ubicación de almacenamiento correspondiente a la fila m y la columna n de la primera matriz,
en donde realizar operaciones comprende realizar una operación correspondiente a la instrucción de procesar valores desnormalizados de las matrices segunda y tercera como cero.

19. Un programa informático que comprende instrucciones que, cuando son ejecutadas por la unidad de procesamiento de la reivindicación 1, hacen que la unidad de procesamiento realice el método de la reivindicación 18.

20. Un medio legible por computadora que tiene almacenado en él el programa informático de la reivindicación 19.

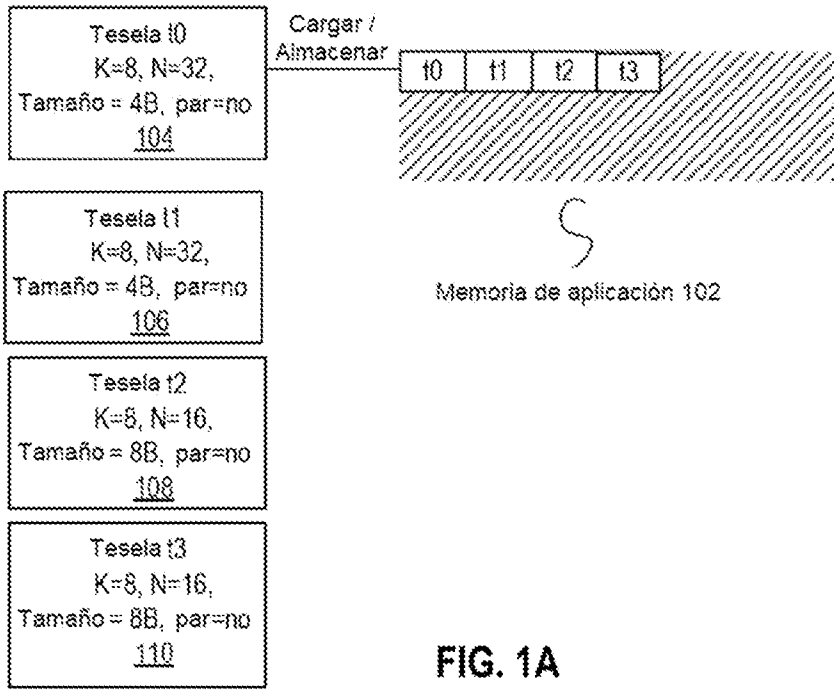

Teselа t0
K=8, N=32,
Tamaño = 4B, par=no
104
Cargar / Almacenar
t0
t1
t2
t3
Teselа t1
K=8, N=32,
Tamaño = 4B, par=no
106
Teselа t2
K=8, N=16,
Tamaño = 8B, par=no
108
Teselа t3
K=8, N=16,
Tamaño = 8B, par=no
110
Memoria de aplicación 102

FIG. 1A

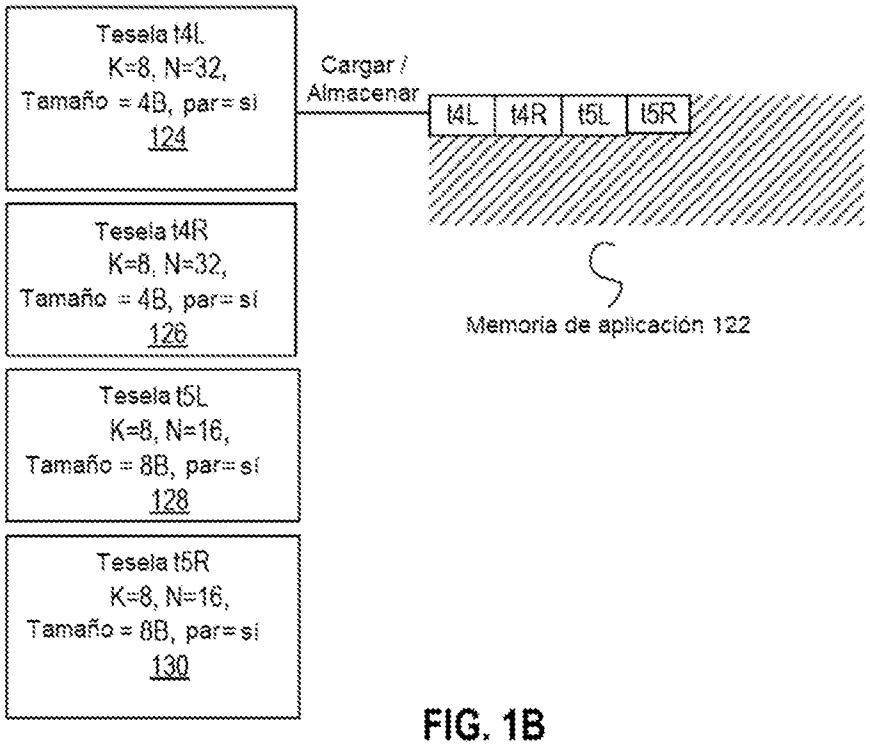

Teselа t4L
K=8, N=32,
Tamaño = 4B, par= sí
124
Cargar / Almacenar
t4L
t4R
t5L
t5R
Teselа t4R
K=8, N=32,
Tamaño = 4B, par= sí
126
Teselа t5L
K=8, N=16,
Tamaño = 8B, par= sí
128
Teselа t5R
K=8, N=16,
Tamaño = 8B, par= sí
130
Memoria de aplicación 122

FIG. 1B

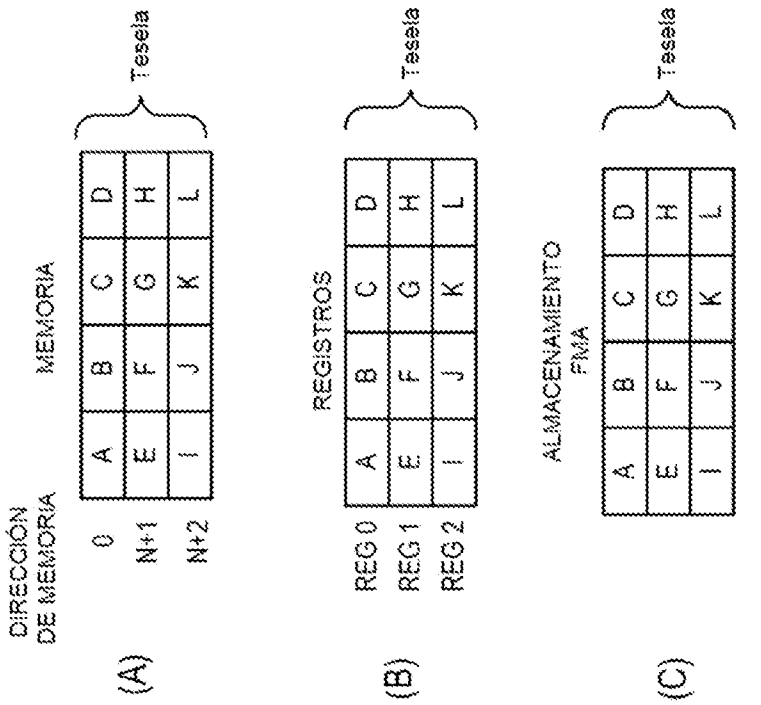

DIRECCIÓN DE MEMORIA
MEMORIA
0
N+1
N+2
A
B
C
D
E
F
G
H
I
J
K
L
Tesela
(A)
REGISTROS
REG 0
REG 1
REG 2
Tesela
(B)
ALMACENAMIENTO PMA
Tesela
(C)

FIG. 2

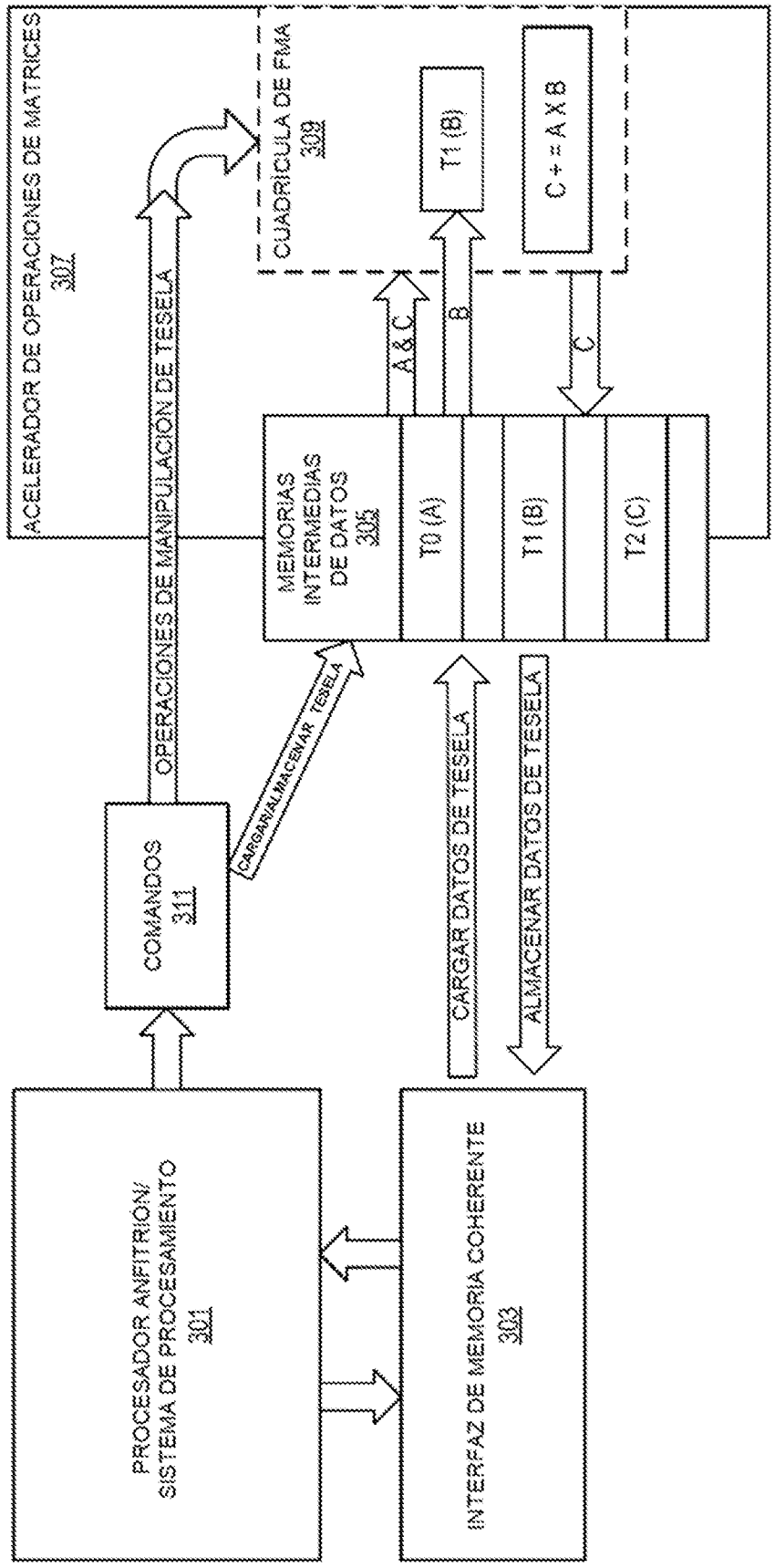

ACELERADOR DE OPERACIONES DE MATRICES
307
CUADRÍCULA DE FMA
309
T1 (B)
C + = A X B
OPERACIONES DE MANIPULACIÓN DE TESELA
A & C
B
C
MEMORIAS INTERMEDIAS DE DATOS
305
T0 (A)
T1 (B)
T2 (C)
CARGAR/ALMACENAR TESELA
COMANDOS
311
CARGAR DATOS DE TESELA
ALMACENAR DATOS DE TESELA
PROCESADOR ANFITRIÓN/ SISTEMA DE PROCESAMIENTO
301
INTERFAZ DE MEMORIA COHERENTE
303

FIG. 3

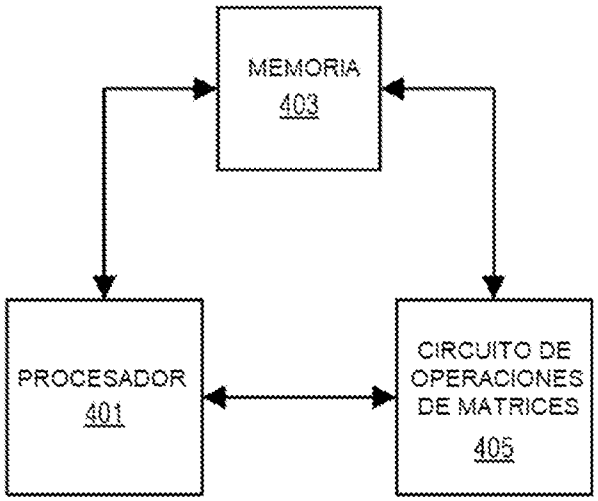

MEMORIA
403
PROCESADOR
401
CIRCUITO DE OPERACIONES DE MATRICES
405

FIG. 4

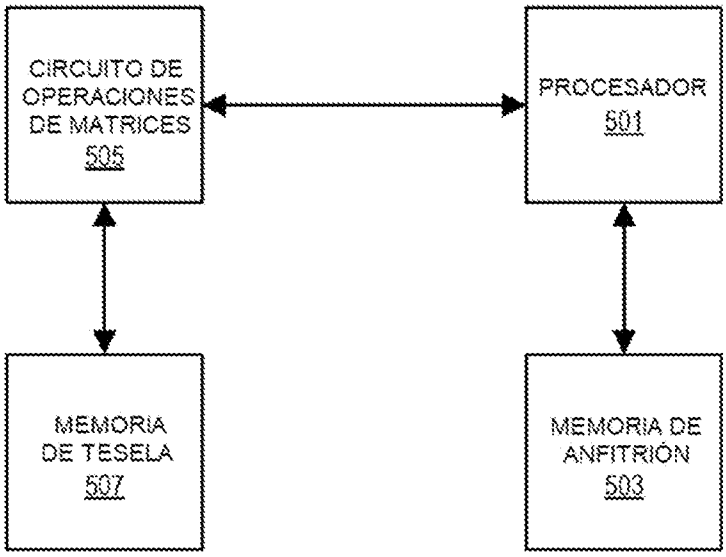

CIRCUITO DE OPERACIONES DE MATRICES
505
PROCESADOR
501
MEMORIA DE TESELA
507
MEMORIA DE ANFITRIÓN
503

FIG. 5

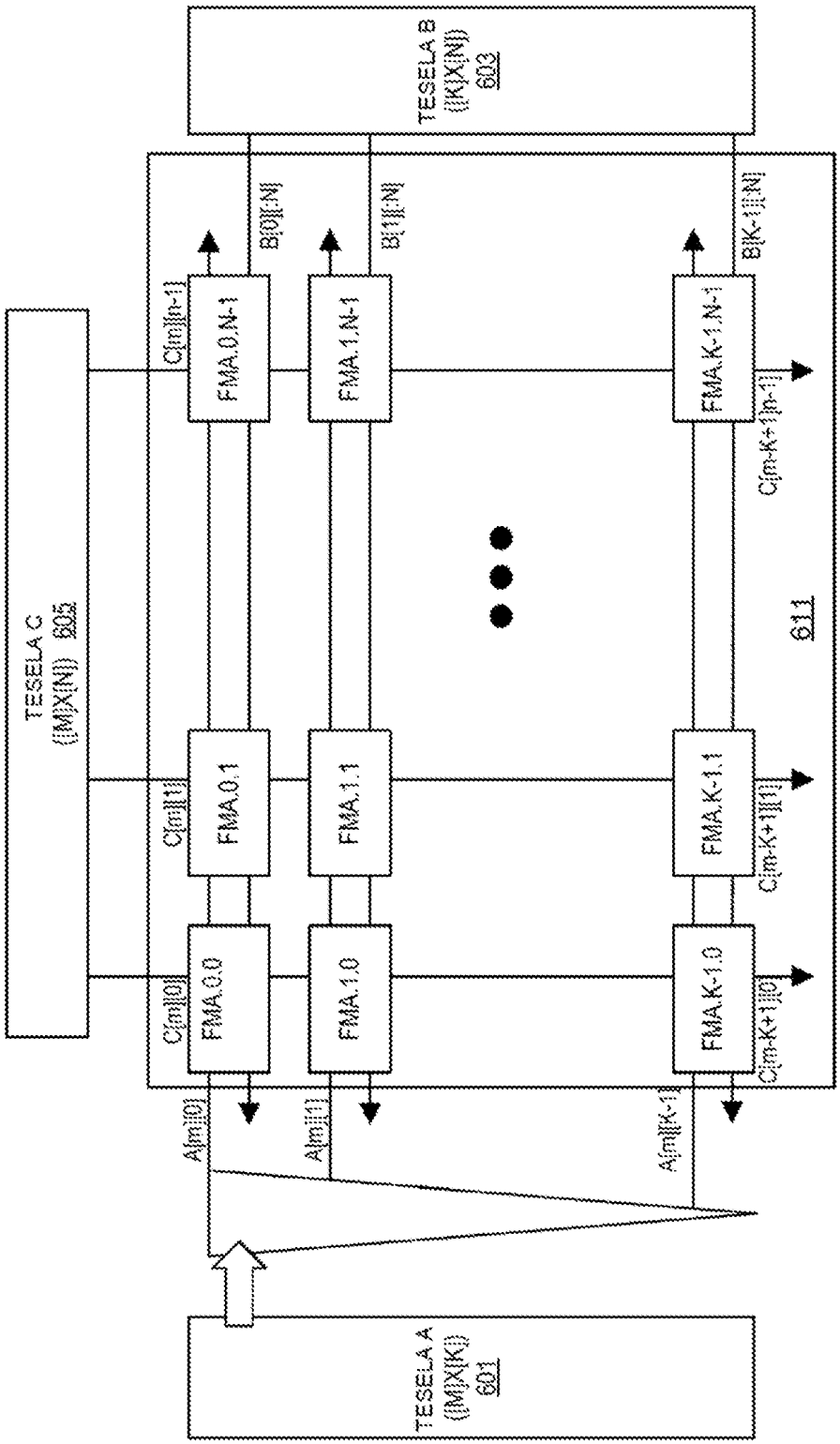

TESELA B
((K)X(N))
602
TESELA C
((M)X(N)) 605
TESELA A
((M)X(K))
601
611
B[0][N]
B[1][N]
B[K-1][N]
C[m][n-1]
C[m][1]
C[m][0]
FMA.0.N-1
FMA.1.N-1
FMA.K-1.N-1
FMA.0.1
FMA.1.1
FMA.K-1.1
FMA.0.0
FMA.1.0
FMA.K-1.0
C[m-K+1][n-1]
C[m-K+1][1]
C[m-K+1][0]
A[m][0]
A[m][1]
A[m][K-1]

FIG. 6

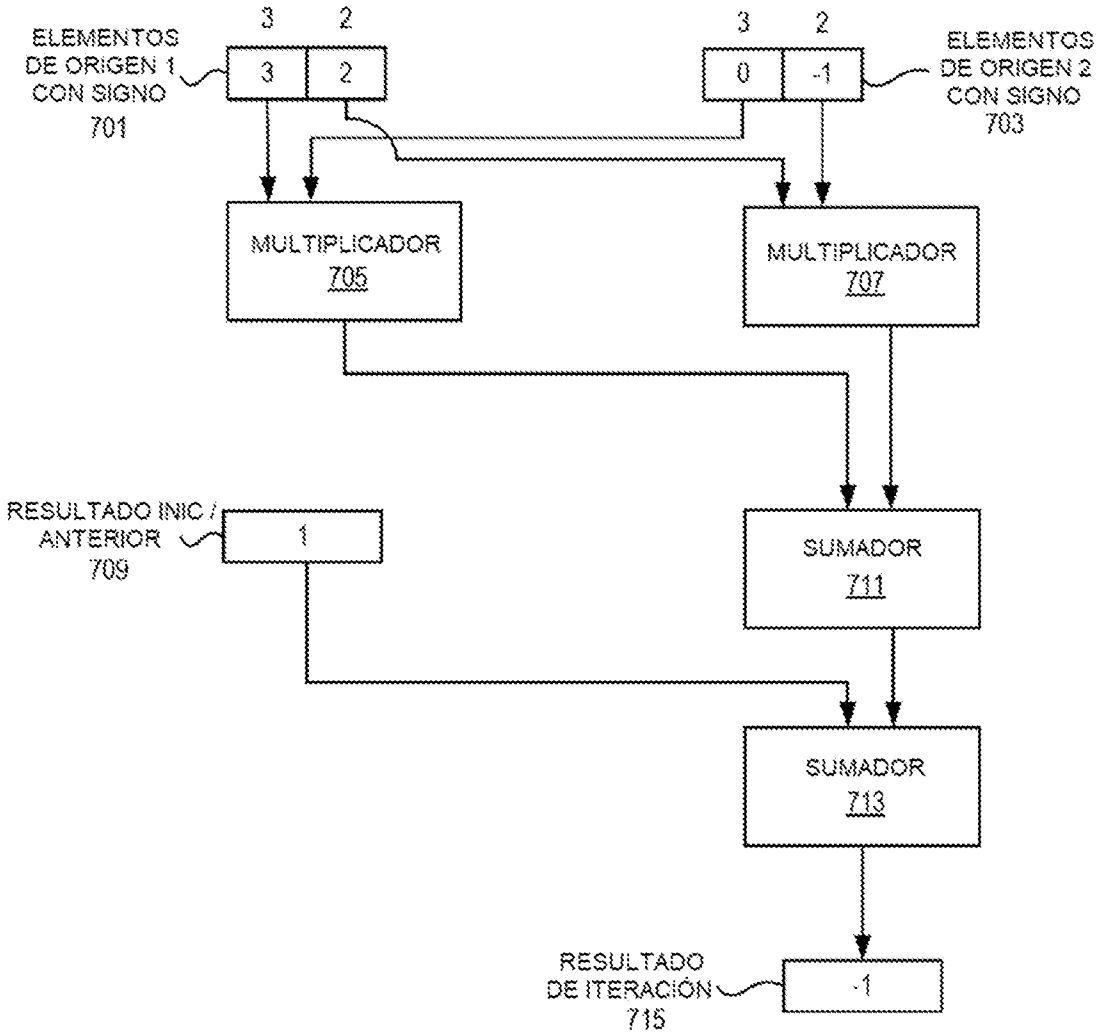

ELEMENTOS DE ORIGEN 1 CON SIGNO 701
3
2
3
2
ELEMENTOS DE ORIGEN 2 CON SIGNO 703
3
2
0
-1
MULTIPLICADOR 705
MULTIPLICADOR 707
RESULTADO INIC / ANTERIOR 709
1
SUMADOR 711
SUMADOR 713
RESULTADO DE ITERACIÓN 715
-1

FIG. 7

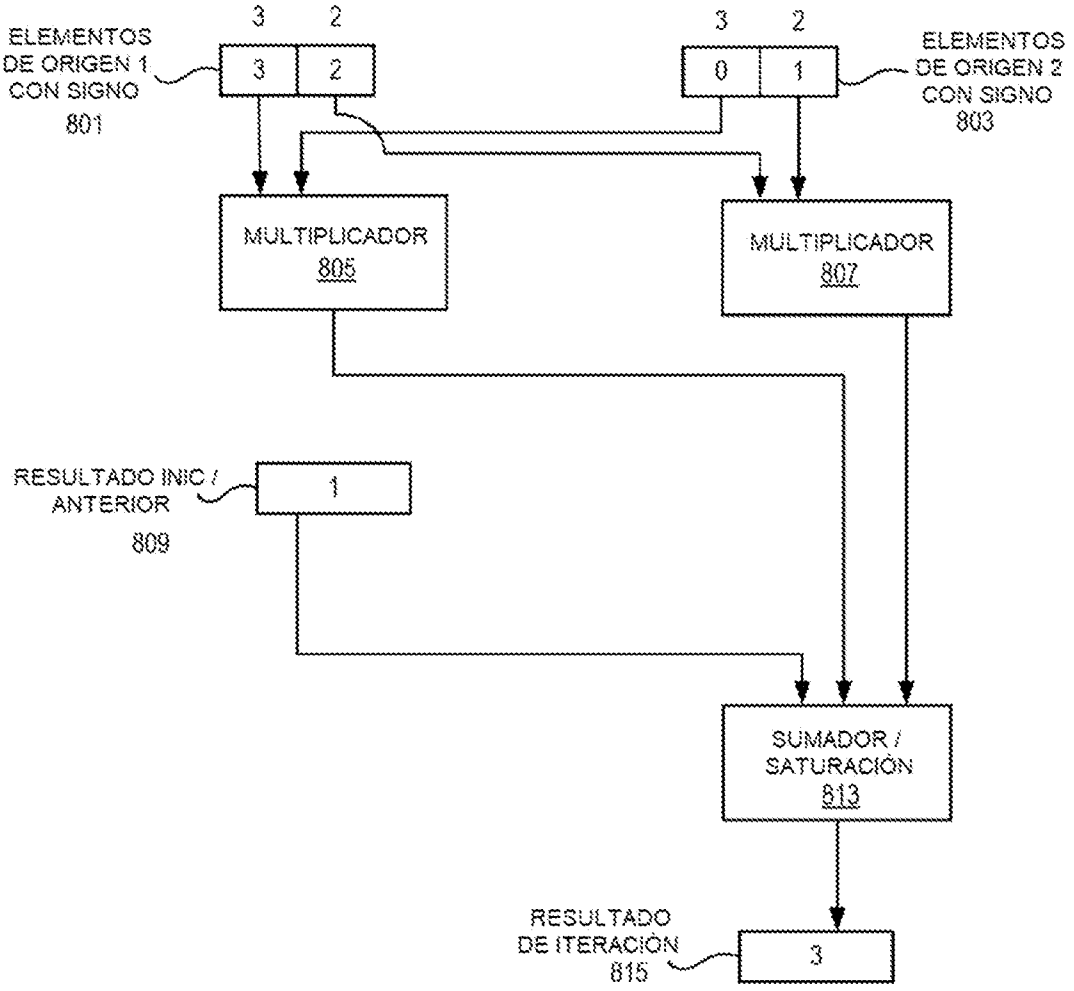

ELEMENTOS DE ORIGEN 1 CON SIGNO
801
3
2
3
2
3
2
0
1
ELEMENTOS DE ORIGEN 2 CON SIGNO
803
MULTIPLICADOR
805
MULTIPLICADOR
807
RESULTADO INIC / ANTERIOR
809
1
SUMADOR / SATURACIÓN
813
RESULTADO DE ITERACIÓN
815
3

FIG. 8

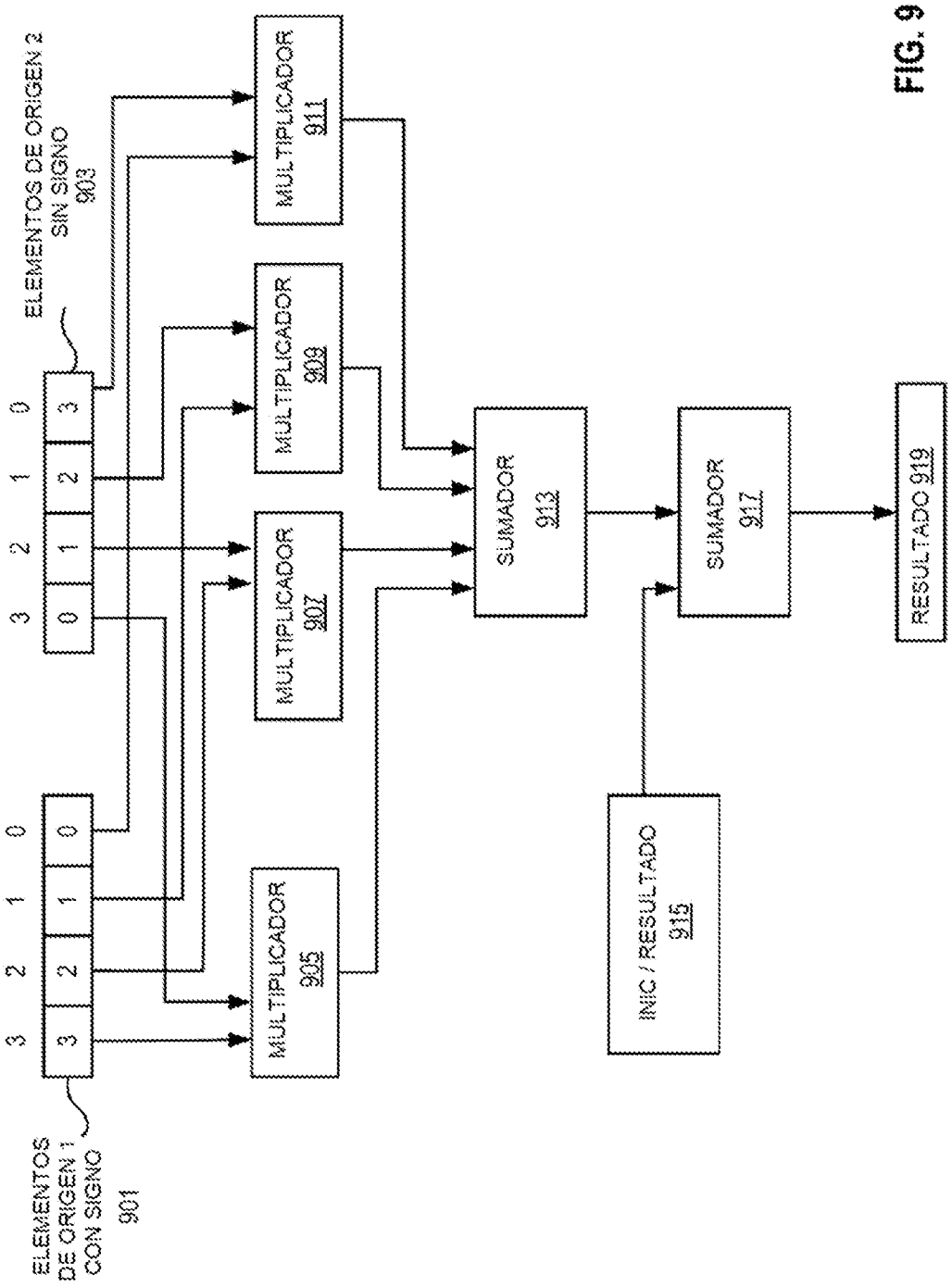

ELEMENTOS DE ORIGEN 2 SIN SIGNO 903
3 2 1 0
0 1 2 3
MULTIPLICADOR 911
MULTIPLICADOR 909
MULTIPLICADOR 907
MULTIPLICADOR 905
ELEMENTOS DE ORIGEN 1 CON SIGNO 901
3 2 1 0
3 2 1 0
SUMADOR 913
SUMADOR 917
RESULTADO 919
INIC / RESULTADO 915

FIG. 9

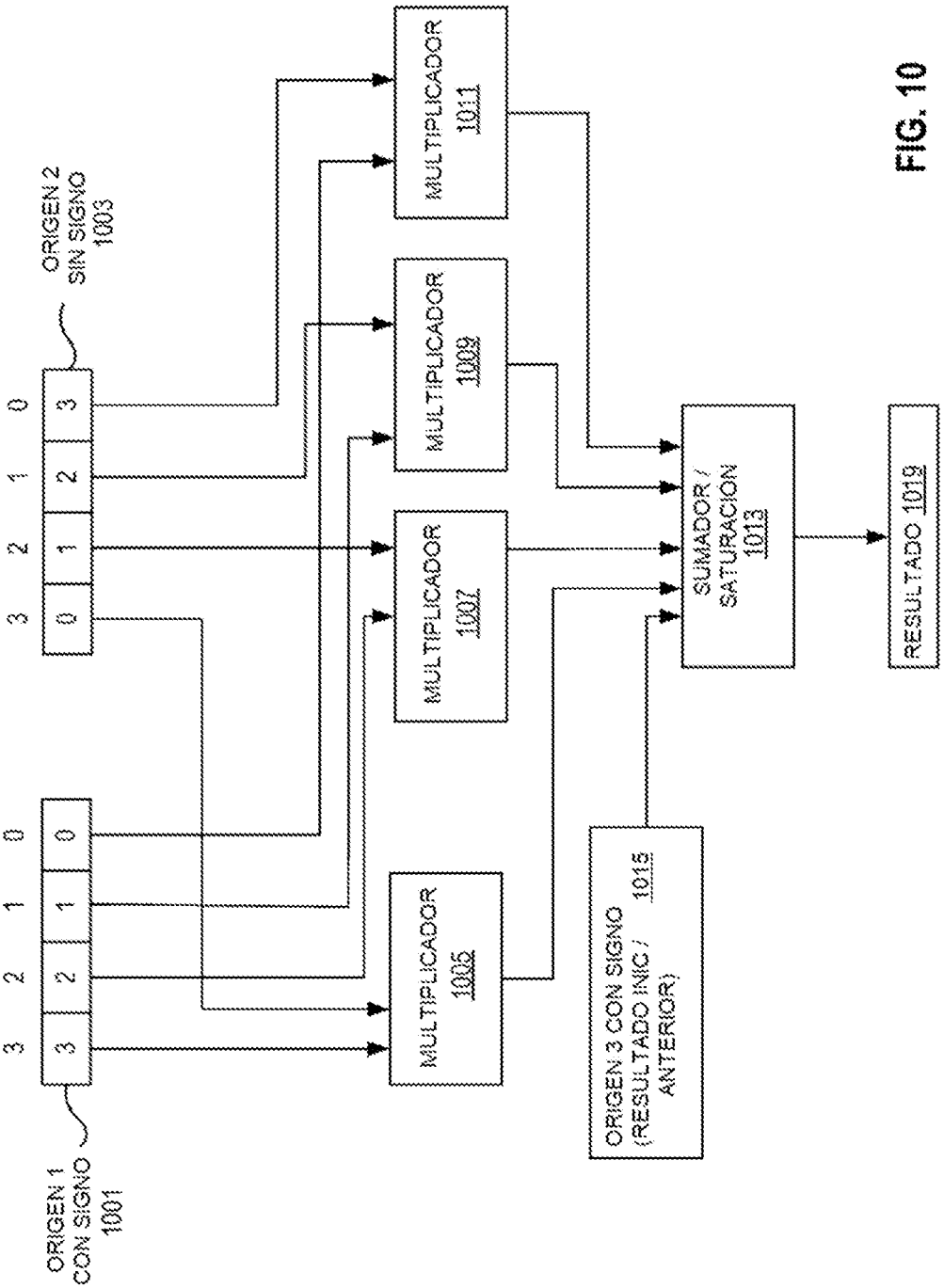

ORIGEN 1
CON SIGNO
1001
3 2 1 0
ORIGEN 2
SIN SIGNO
1003
3 2 1 0
MULTIPLICADOR
1005
MULTIPLICADOR
1007
MULTIPLICADOR
1009
MULTIPLICADOR
1011
ORIGEN 3 CON SIGNO
(RESULTADO INIC /
ANTERIOR)
1015
SUMADOR /
SATURACION
1013
RESULTADO 1019

FIG. 10

ACUMULADOR DE TAMAÑOS DE ENTRADA 2X 1101

ORÍGENES	BITS	ACUMULADOR	BITS
BYTE	8	WORD/HPFP	16
WORD	16	INT32/SPFP	32
SPFP/INT32	32	INT64/DPFP	64

ACUMULADOR DE TAMAÑOS DE ENTRADA 4X 1103

ORÍGENES	BITS	ACUMULADOR	BITS
BYTE	8	INT32/SPFP	32
WORD	16	INT64/DPFP	64

ACUMULADOR DE TAMAÑOS DE ENTRADA 8X 1105

ORÍGENES	BITS	ACUMULADOR	BITS
BYTE	8	INT64/DPFP	64

FIG. 11

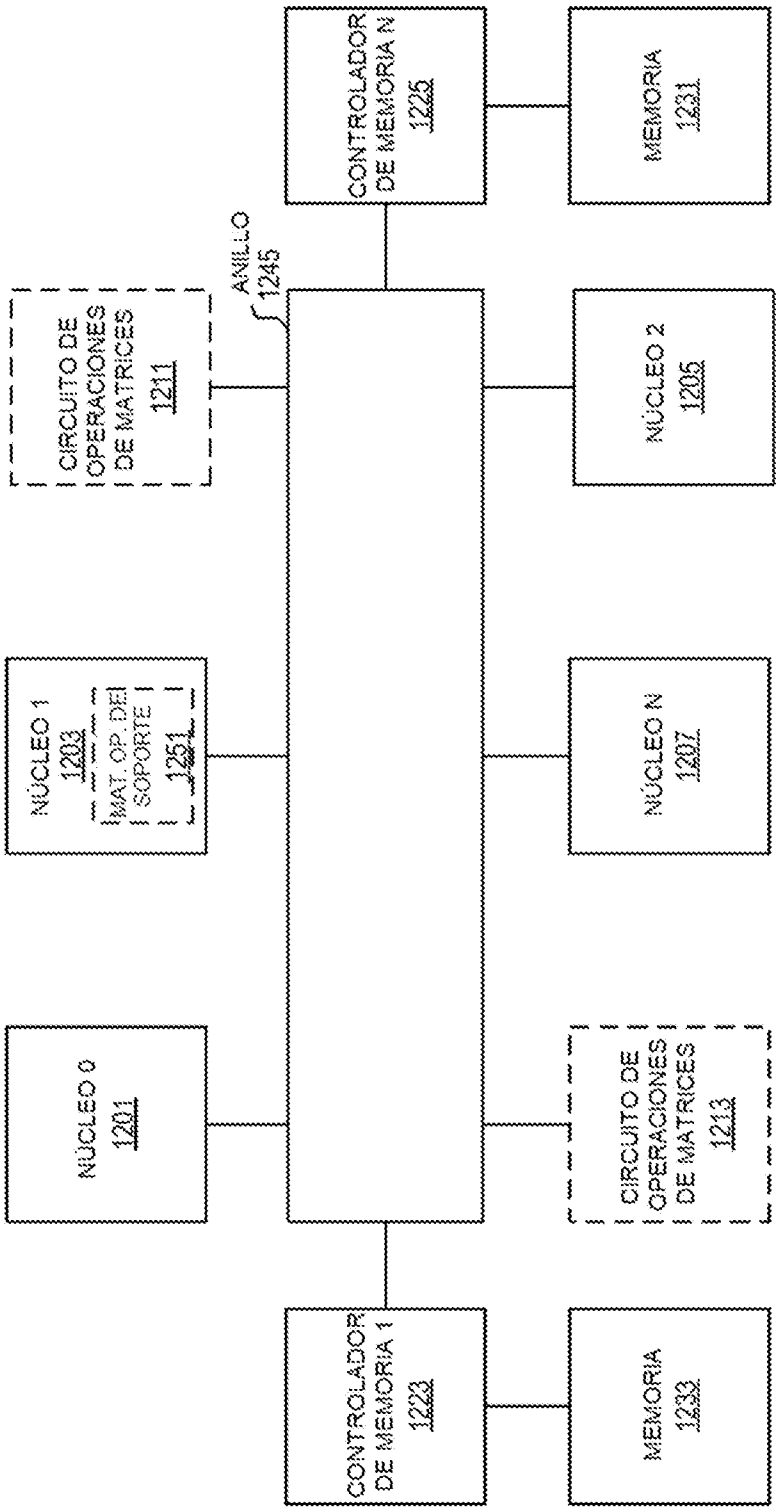

CONTROLADOR DE MEMORIA N 1225
MEMORIA 1231
ANILLO 1245
CIRCUITO DE OPERACIONES DE MATRICES 1211
NÚCLEO 2 1205
NÚCLEO 1 1203
MAT. OP. DEL SOPORTE 1251
NÚCLEO N 1207
NÚCLEO 0 1201
CIRCUITO DE OPERACIONES DE MATRICES 1213
CONTROLADOR DE MEMORIA 1 1223
MEMORIA 1233

FIG. 12

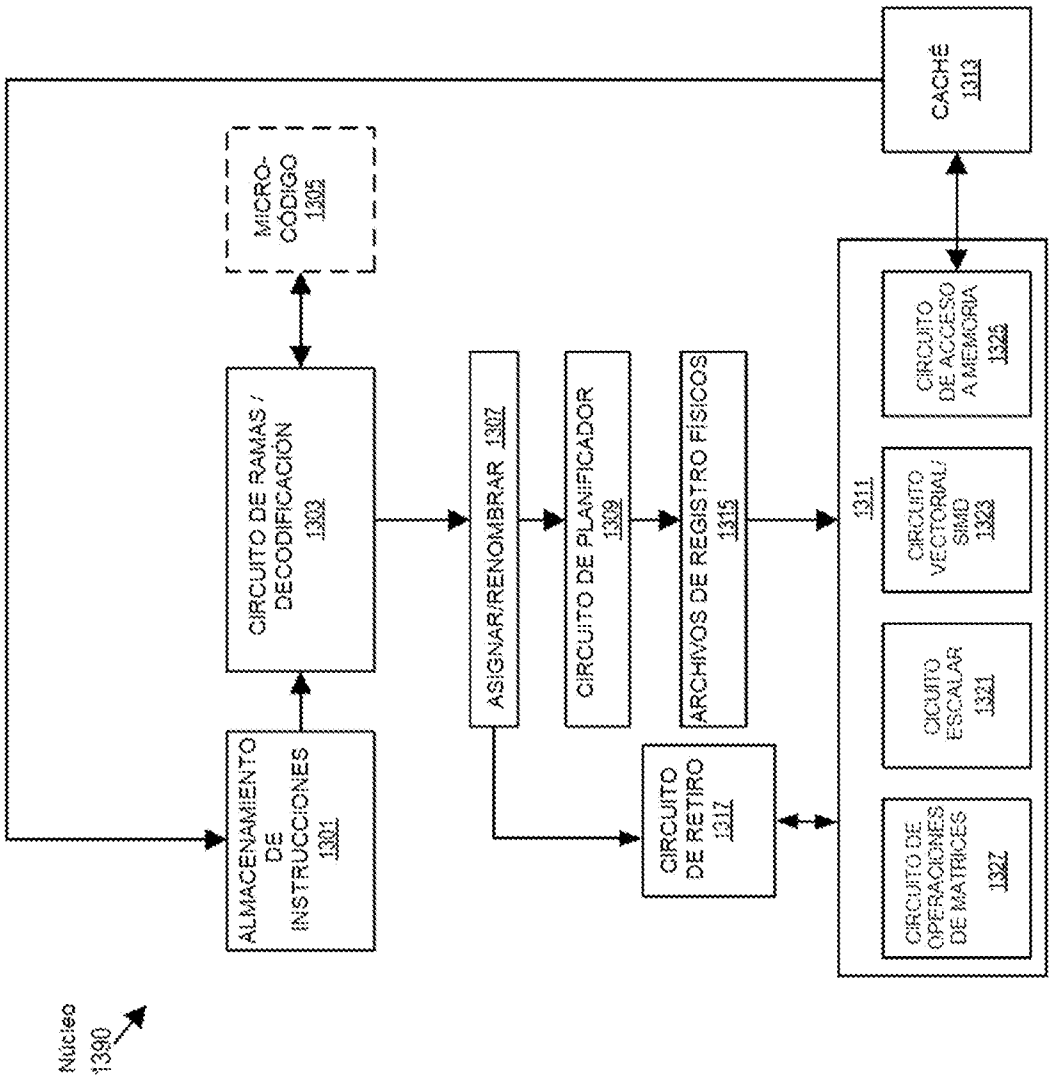

Núcleo
1390
ALMACENAMIENTO DE INSTRUCCIONES 1301
CIRCUITO DE RAMAS / DECODIFICACIÓN 1303
MICRO-CÓDIGO 1305
ASIGNAR/RENOMBRAR 1307
CIRCUITO DE PLANIFICADOR 1309
ARCHIVOS DE REGISTRO FÍSICOS 1315
CIRCUITO DE RETIRO 1317
1311
CIRCUITO DE OPERACIONES DE MATRICES 1327
CIRCUITO ESCALAR 1321
CIRCUITO VECTORIAL/ SIMD 1323
CIRCUITO DE ACCESO A MEMORIA 1325
CACHÉ 1313

FIG. 13

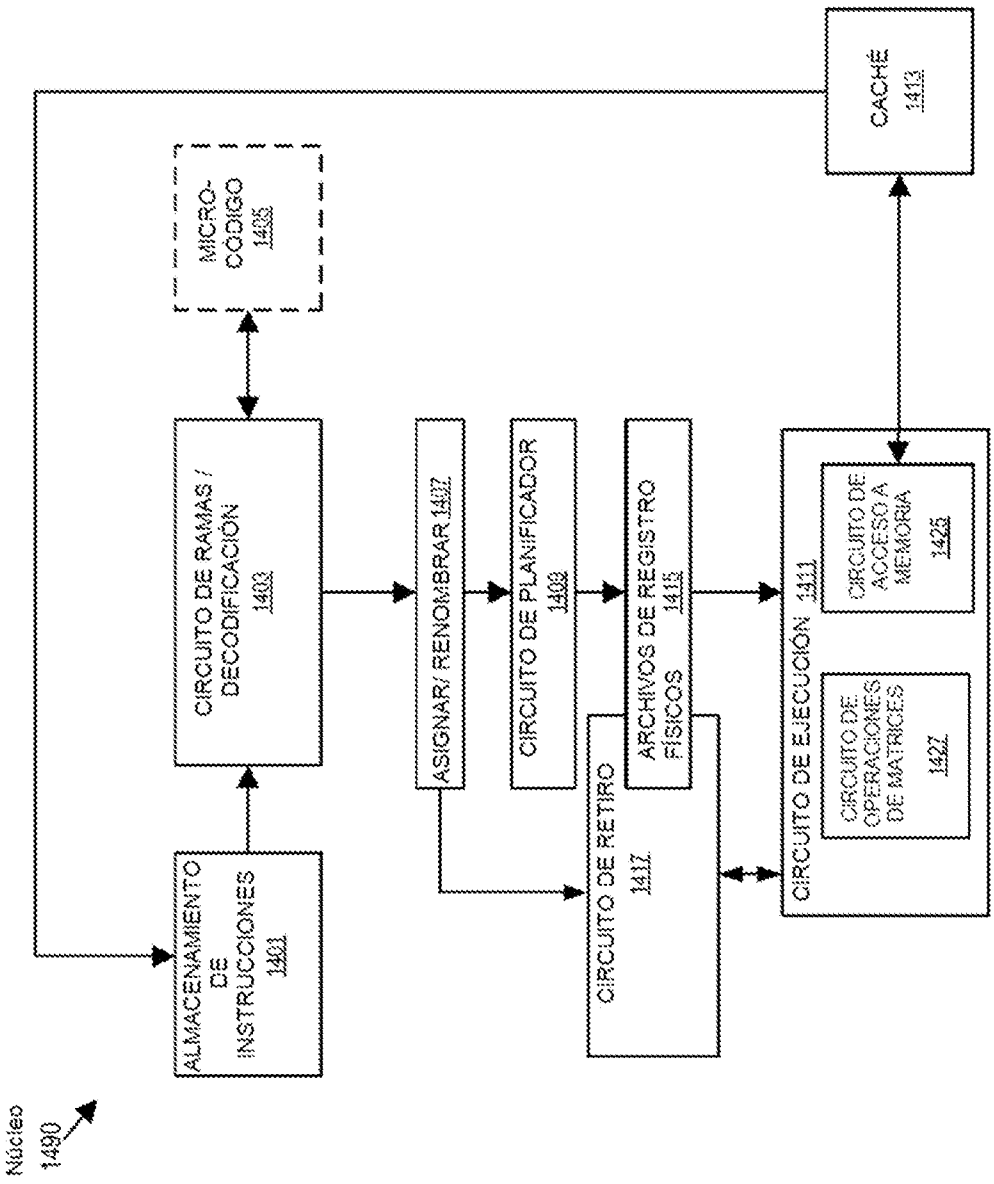

Núcleo
1490
ALMACENAMIENTO DE INSTRUCCIONES 1401
CIRCUITO DE RAMAS / DECODIFICACIÓN 1403
MICRO-CÓDIGO 1435
ASIGNAR/ RENOMBRAR 1407
CIRCUITO DE PLANIFICADOR 1409
ARCHIVOS DE REGISTRO FÍSICOS 1415
CIRCUITO DE RETIRO 1417
CIRCUITO DE EJECUCIÓN 1411
CIRCUITO DE ACCESO A MEMORIA 1425
CIRCUITO DE OPERACIONES DE MATRICES 1427
CACHÉ 1413

FIG. 14

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix}$$

DIR	VALOR
0	A_{11}
1	A_{12}
2	A_{13}
3	A_{21}
4	A_{22}
5	A_{23}

FILA PRINCIPAL

DIR	VALOR
0	A_{11}
1	A_{21}
2	A_{12}
3	A_{22}
4	A_{13}
5	A_{23}

COLUMNA PRINCIPAL

FIG. 15

```
TILECONFIG (RAX)
// SUPONER QUE ALGUNOS BUCLES EXTERIORES ACCIONAN LA TESELA DE CACHÉ (NO MOSTRADA)
{
TILELOAD TMM0, RSI+RDI // SRCDST, RSI APUNTA A C, RDI TIENE
TILELOAD TMM1, RSI+RDI+N // SEGUNDA TESELA DE C, SE DESPLIEGA EN DIMENSIÓN N DE SIMD
MOV KK, 0
LOOP:
TILELOAD TMM2, R8+R9 // SRC2 ES CARGA DE AVANCE DE A, REUTILIZADA PARA 2 INSTR. TMMA
TILELOAD TMM3, R10+R11 // SRC1 ES CARGA DE AVANCE DE B
TMMAPS TMM0, TMM2, TMM3 // ACTUALIZAR TESELA IZQUIERDA DE C
TILELOAD TMM3, R10+R11+N // SRC1 SE CARGA CON B DESDE LA TESELA MÁS A LA DERECHA
TMMAPS TMM1, TMM2, TMM3 // ACTUALIZAR TESLA DERECHA DE C
ADD R8, K // ACTUALIZAR PUNTEROS CON CONSTANTES CONOCIDAS FUERA DEL BUCLE
ADD R10, K*R11
ADD KK, K
CMP KK, LIMIT
JNE LOOP
TILESTORE RSI+RDI, TMM0 // ACTUALIZAR LA MATRIZ C EN MEMORIA
TILESTORE RSI+RDI+M, TMM1
} // FIN DE BUCLE EXTERIOR
TILERELEASE // DEVUELVE TESELAS AL ESTADO INIC
```

FIG. 16

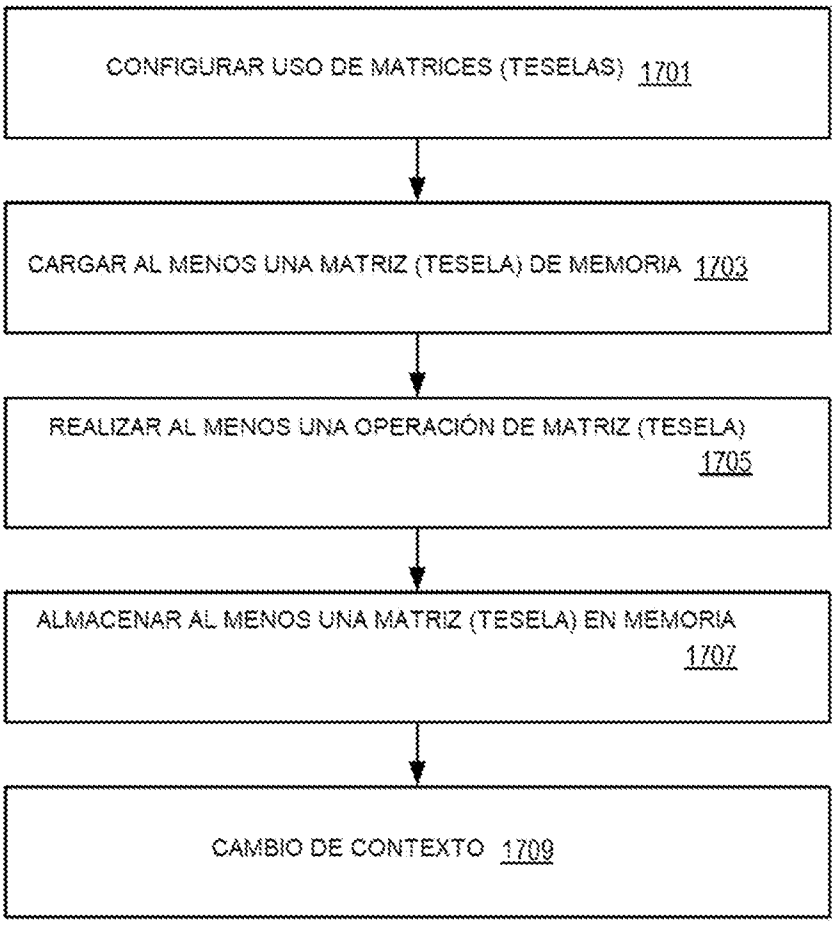

CONFIGURAR USO DE MATRICES (TESELAS) 1701
CARGAR AL MENOS UNA MATRIZ (TESELA) DE MEMORIA 1703
REALIZAR AL MENOS UNA OPERACIÓN DE MATRIZ (TESELA) 1705
ALMACENAR AL MENOS UNA MATRIZ (TESELA) EN MEMORIA 1707
CAMBIO DE CONTEXTO 1709

FIG. 17

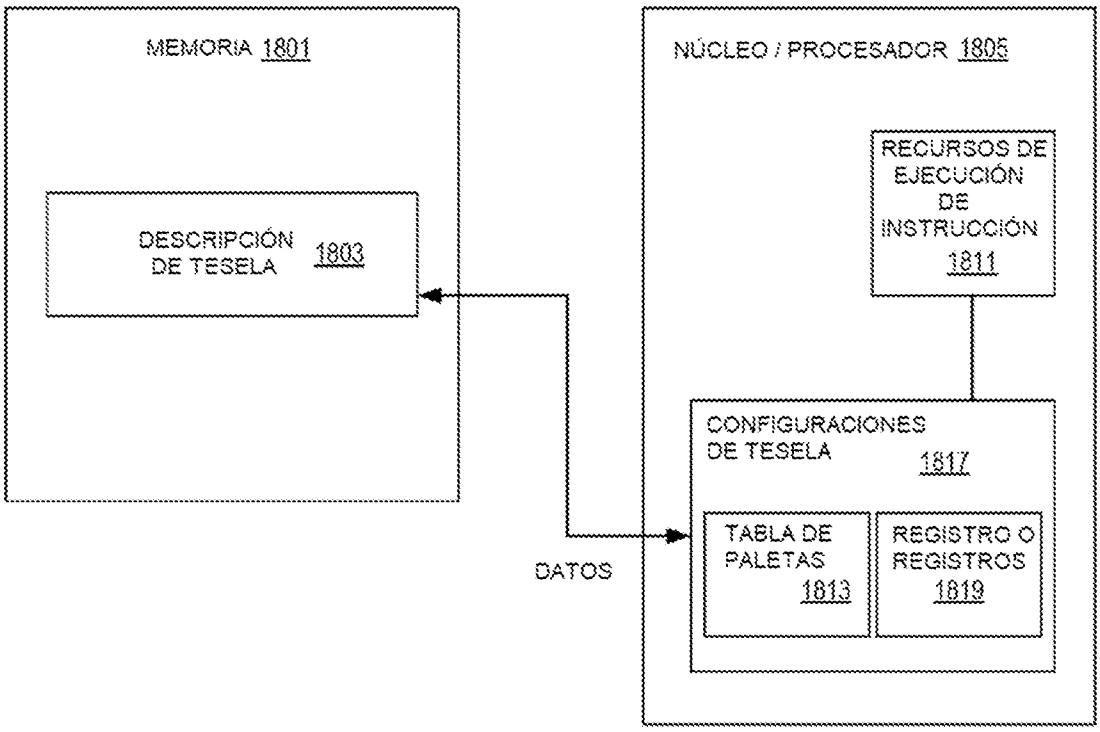

MEMORIA 1801
DESCRIPCIÓN DE TESELA 1803
NÚCLEO / PROCESADOR 1805
RECURSOS DE EJECUCIÓN DE INSTRUCCIÓN 1811
CONFIGURACIONES DE TESELA 1817
TABLA DE PALETAS 1813
REGISTRO O REGISTROS 1819
DATOS

FIG. 18

ID DE PALETA 1901	STARTM 1903
STARTP 1905	INDICADORES DE PAR 1907
0	0
0	0

. . .

0	0
TMM0 FILAS 1913	TMM0 COLUMNAS 1915
TMM1 FILAS	TMM1 COLUMNAS
* * *	
TMM15 FILAS	TMM15 COLUMNAS
0	

FIG. 19

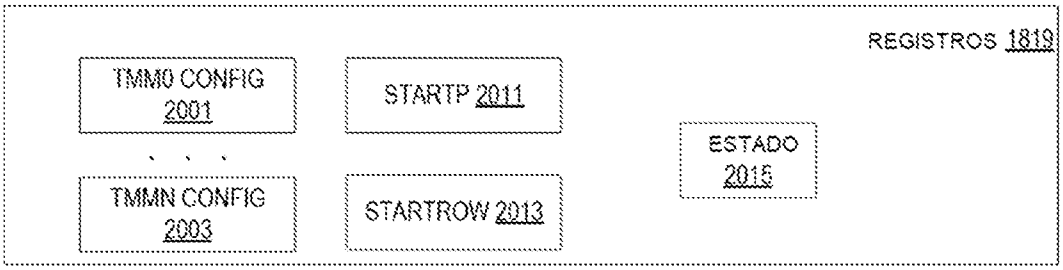


FIG. 20(A)

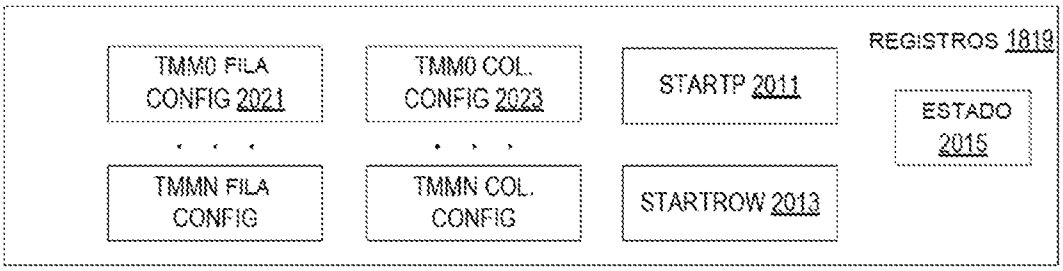


FIG. 20(B)

FIG. 20(C)

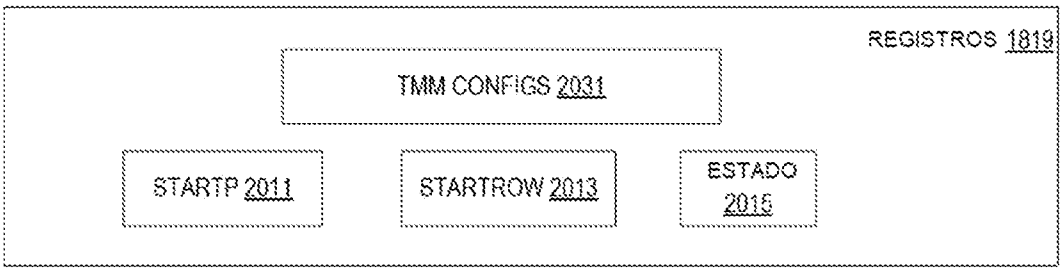


FIG. 20(D)

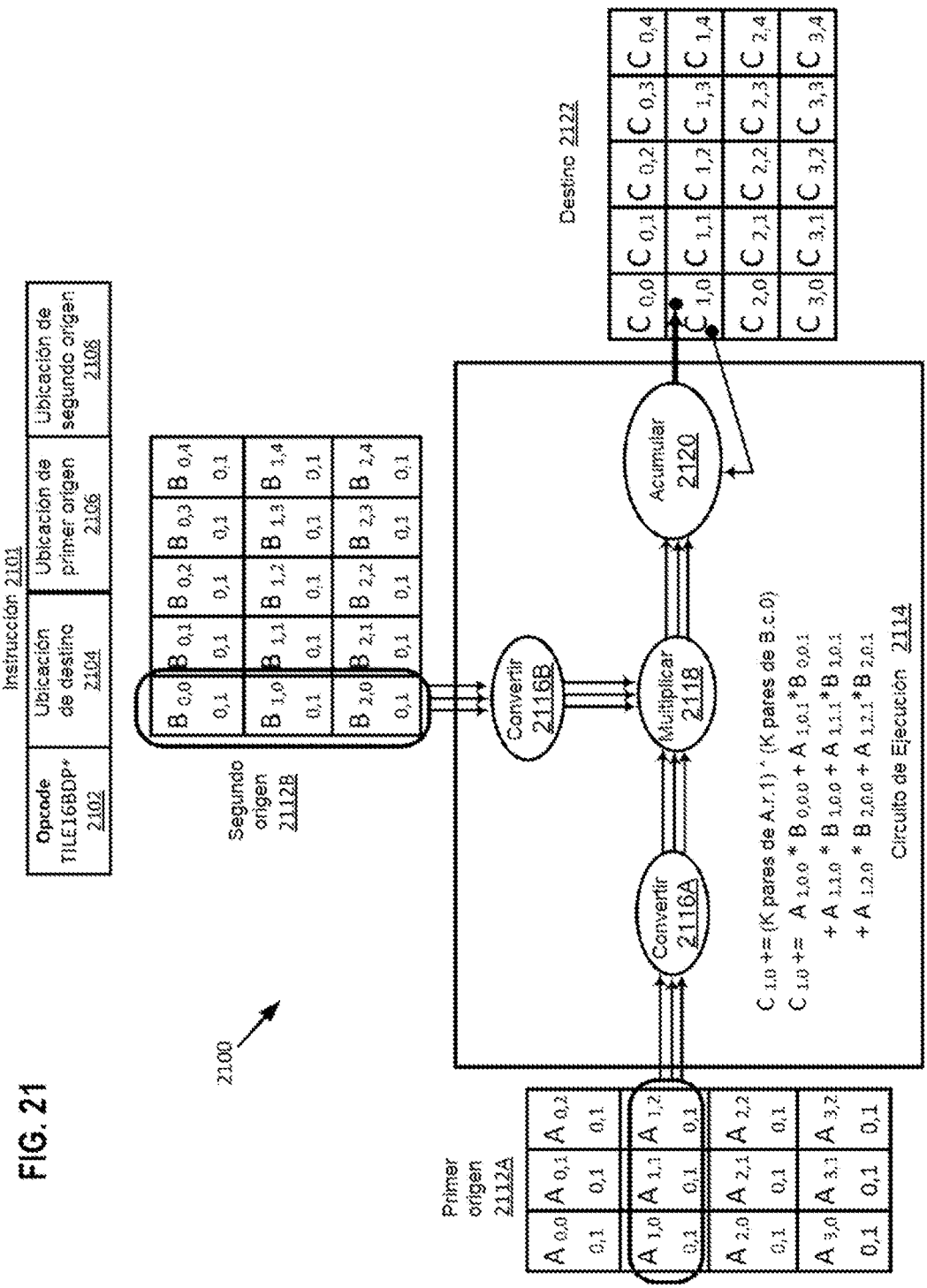

Instrucción 2101
Opcode TILE16BDP* 2102
Ubicación de destino 2104
Ubicación de primer origen 2106
Ubicación de segundo origen 2108
Segundo origen 2112B
Primer origen 2112A
Destino 2122
Convertir 2116B
Convertir 2116A
Multiplicar 2118
Acumular 2120
Circuito de Ejecución 2114
2100
C 1,0 += (K pares de A,r,1) ' (K pares de B,c,0)
C 1,0 += A 1,0,0 * B 0,0,0 + A 1,0,1 *B 0,0,1
+ A 1,1,0 * B 1,0,0 + A 1,1,1 *B 1,0,1
+ A 1,2,0 * B 2,0,0 + A 1,2,1 *B 2,0,1

FIG. 21

Instrucción 2201

Opcode TILE16BDP* 2202	Ubicación de destino 2204	Ubicación de primer origen 2206	Ubicación de segundo origen 2208

Pseudocódigo 2200

```
TILE16BDP tsrcdest, tsrc1, tsrc2
// C = m x n (tsrcdest), A = m x k (tsrc1),  B = k x n (tsrc2)

# los elementos src1 y src2 son pares de bfloat16
elements_src1 := tsrc1.colsb / 4
elements_src2 := tsrc2.colsb / 4
elements_dest := tsrcdest.colsb / 4
elements_temp := tsrcdest.colsb / 2 // Cuenta es en bfloat16 antes que horizontal

for m in 0 ... tsrcdest.rows-1:
  temp1[ 0 ... elements_temp-1 ] := 0
  for k in 0 ... elements_src1-1:
     for n in 0 ... elements_dest-1:

       // FP32 FMA estándar, DAZ=FTZ=1, redondeo RNE.
       // MXCSR ni se consulta ni se actualiza.
       // No saltan ni se señalan excepciones

       temp1.fp32[2*n+0] += make_fp32(tsrc1.row[m].bfloat16[2*k+0]) *
                    make_fp32(tsrc2.row[k].bfloat16[2*n+0])
       temp1.fp32[2*n+1] += make_fp32(tsrc1.row[m].bfloat16[2*k+1]) *
                    make_fp32(tsrc2.row[k].bfloat16[2*n+1])

  for n in 0 ... elements_dest-1:
     // DAZ=FTZ=1, redondeo RNE.
     // MXCSR ni se consulta ni se actualiza.
     // No saltan ni se señalan excepciones
     f[n] := temp1.fp32[2*n] + temp1.fp32[2*n+1]
     srcdest.row[m].fp32[n] += f[n]
  write_row_and_zero(tsrcdest, m, tmp, tsrcdest.colsb)

zero_upper_rows(tsrcdest, tsrcdest.rows)
zero_tileconfig_start()
```

FIG. 22A

Instruction 2211

Opcode TILE16BDP* 2212	Ubicación de destino 2214	Ubicación de primer origen 2216	Ubicación de segundo origen 2218

Pseudocódigo 2210

```
TILE16BDP tsrcdest, tsrc1, tsrc2
// C = m x n (tsrcdest), A = m x k (tsrc1),  B = k x n (tsrc2)

# los elementos src1 y src2 son pares de bfloat16
elements_src1 := tsrc1.colsb / 4
elements_src2 := tsrc2.colsb / 4
elements_dest := tsrcdest.colsb / 4
elements_temp := tsrcdest.colsb / 2 // Cuenta es en bfloat16 antes que horizontal

for m in 0 ... tsrcdest.rows-1:
  temp1[ 0 ... elements_temp-1 ] := 0
  for k in 0 ... elements_src1-1:
     for n in 0 ... elements_dest-1:

       // FP32 FMA estándar, DAZ=FTZ=1, redondeo RNE.
       // MXCSR ni se consulta ni se actualiza.
       // No saltan ni se señalan excepciones

       temp1.fp32[2*n+1] += make_fp32(tsrc1.row[m].bfloat16[2*k+1]) *
                    make_fp32(tsrc2.row[k].bfloat16[2*n+1])
       temp1.fp32[2*n+0] += make_fp32(tsrc1.row[m].bfloat16[2*k+0]) *
                    make_fp32(tsrc2.row[k].bfloat16[2*n+0])

  for n in 0 ... elements_dest-1:
     // DAZ=FTZ=1, redondeo RNE.
     // MXCSR ni se consulta ni se actualiza.
     // No saltan ni se señalan excepciones
     f[n] := temp1.fp32[2*n] + temp1.fp32[2*n+1]
     srcdest.row[m].fp32[n] += f[n]
  write_row_and_zero(tsrcdest, m, tmp, tsrcdest.colsb)

zero_upper_rows(tsrcdest, tsrcdest.rows)
zero_tileconfig_start()
```

FIG. 22B

Pseudocódigo para funciones auxiliares 2220

```
define make_fp32(x):
      // El parámetro x es bfloat16. Empaquetarlo en los 16b superiores de un dword
      // El patrón de bits es un valor fp32 legal. Devolver que dword:= 0 del patrón de bits
     dword[31:16] := x return
   dword



define write_row_and_zero(treg, r, data, nbytes):
    for j in 0 ... nbytes-1:
        treg.row[r].byte[j] := data.byte[j]
    //establecer a cero el resto de la fila
    for j in nbytes ... palette_table[tileconfig.palette_id].bytes_per_row-1:
        treg.row[r].byte[j] := 0



define zero_upper_rows(treg, r):
       for i in r ... palette_table[tileconfig.palette_id].max_rows-1:
           for j in 0 ... palette_table[tileconfig.palette_id].bytes_per_row-1:
               treg.row[i].byte[j] := 0



define zero_tileconfig_start():
  tileconfig.startRow := 0
```

FIG. 22C

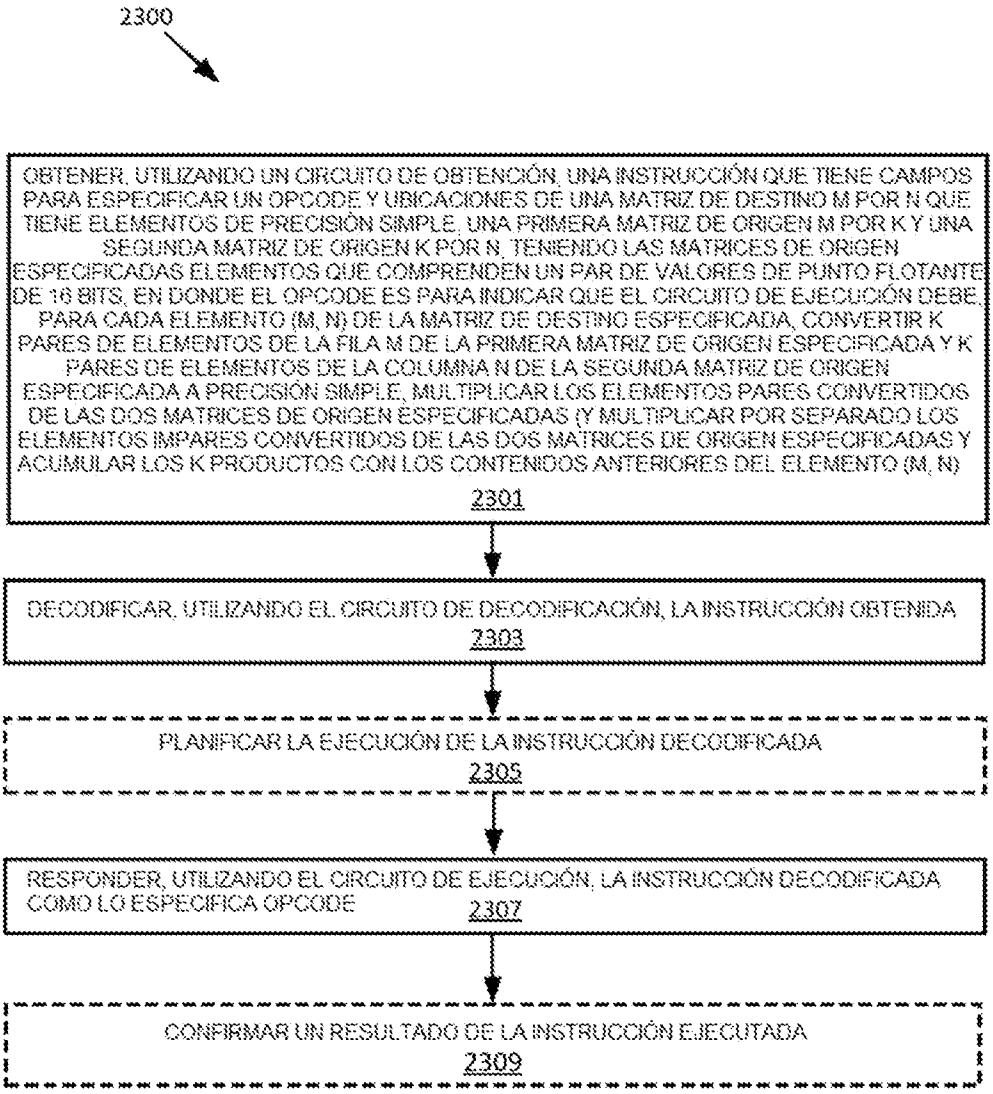

2300
OBTENER, UTILIZANDO UN CIRCUITO DE OBTENCIÓN, UNA INSTRUCCIÓN QUE TIENE CAMPOS PARA ESPECIFICAR UN OPCODE Y UBICACIONES DE UNA MATRIZ DE DESTINO M POR N QUE TIENE ELEMENTOS DE PRECISIÓN SIMPLE, UNA PRIMERA MATRIZ DE ORIGEN M POR K Y UNA SEGUNDA MATRIZ DE ORIGEN K POR N, TENIENDO LAS MATRICES DE ORIGEN ESPECIFICADAS ELEMENTOS QUE COMPRENDEN UN PAR DE VALORES DE PUNTO FLOTANTE DE 16 BITS, EN DONDE EL OPCODE ES PARA INDICAR QUE EL CIRCUITO DE EJECUCIÓN DEBE, PARA CADA ELEMENTO (M, N) DE LA MATRIZ DE DESTINO ESPECIFICADA, CONVERTIR K PARES DE ELEMENTOS DE LA FILA M DE LA PRIMERA MATRIZ DE ORIGEN ESPECIFICADA Y K PARES DE ELEMENTOS DE LA COLUMNA N DE LA SEGUNDA MATRIZ DE ORIGEN ESPECIFICADA A PRECISIÓN SIMPLE, MULTIPLICAR LOS ELEMENTOS PARES CONVERTIDOS DE LAS DOS MATRICES DE ORIGEN ESPECIFICADAS (Y MULTIPLICAR POR SEPARADO LOS ELEMENTOS IMPARES CONVERTIDOS DE LAS DOS MATRICES DE ORIGEN ESPECIFICADAS Y ACUMULAR LOS K PRODUCTOS CON LOS CONTENIDOS ANTERIORES DEL ELEMENTO (M, N)
2301
DECODIFICAR, UTILIZANDO EL CIRCUITO DE DECODIFICACIÓN, LA INSTRUCCIÓN OBTENIDA
2303
PLANIFICAR LA EJECUCIÓN DE LA INSTRUCCIÓN DECODIFICADA
2305
RESPONDER, UTILIZANDO EL CIRCUITO DE EJECUCIÓN, LA INSTRUCCIÓN DECODIFICADA COMO LO ESPECIFICA OPCODE
2307
CONFIRMAR UN RESULTADO DE LA INSTRUCCIÓN EJECUTADA
2309

FIG. 23

Producto Escalar de Tesela de 16 bits (TILE16BOP) Instrucción 2400

Opcode TILE16BOP *	Ubicación de destino	Ubicación de primer origen	Ubicación de segundo origen	Formato de elemento de origen	K	M (Filas Origen 1)	N (Columnas Origen 2)
2402	2404	2406	2408	2410	2412	2414	2416

FIG. 24

FIG. 25A

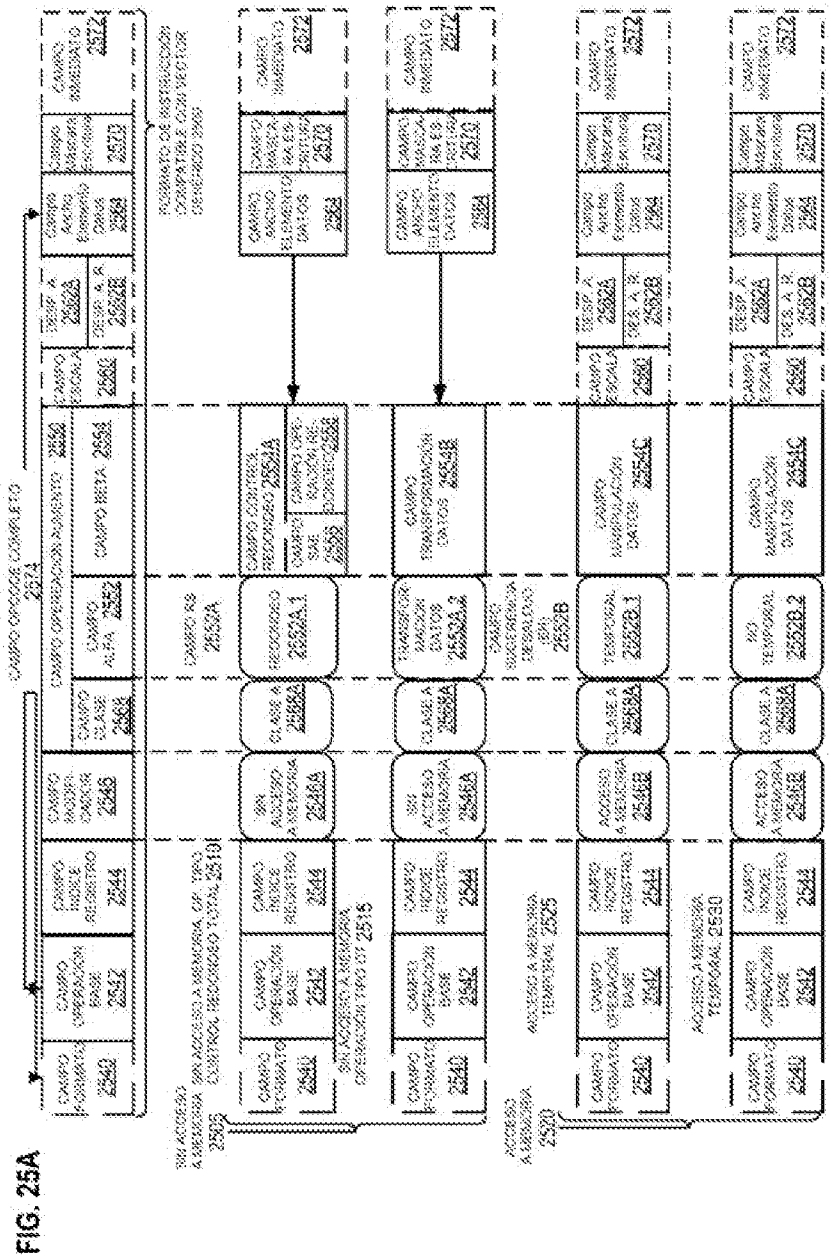
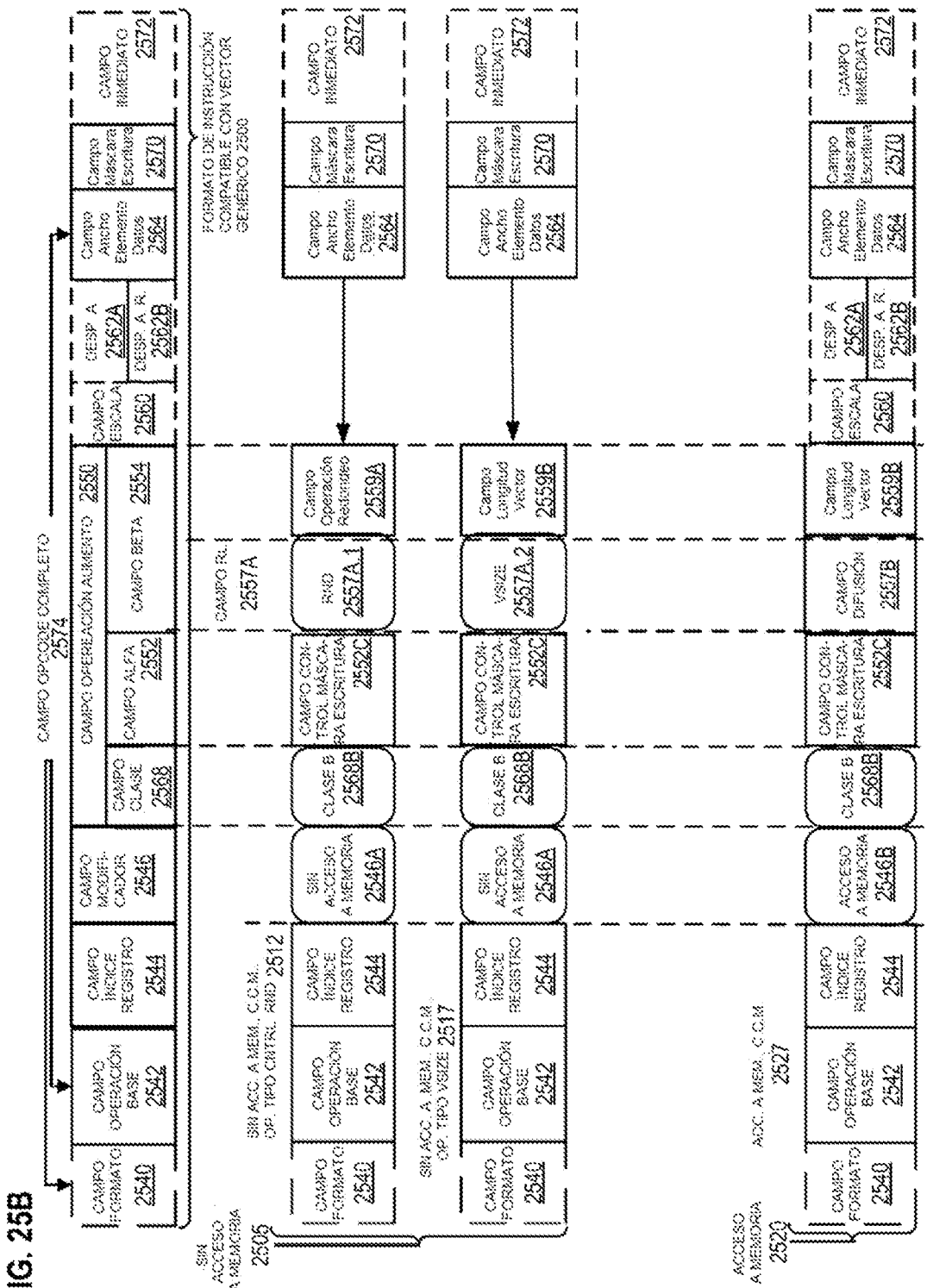


FIG. 25B


CAMPO OPCODE COMPLETO 2574
FORMATO DE INSTRUCCIÓN COMPATIBLE CON VECTOR GENÉRICO 2500
SIN ACCESO A MEMORIA 2505
ACCESO A MEMORIA 2520
SIN ACC. A MEM., C.O.M., OP. TIPO CNTRL, RMD 2512
SIN ACC. A MEM., C.O.M., OP. TIPO VSIZE 2517
ACC. A MEM., C.O.M. 2527
CAMPO FORMATO 2540
CAMPO OPERACIÓN BASE 2542
CAMPO ÍNDICE REGISTRO 2544
CAMPO MODIFICADOR 2546
SIN ACCESO A MEMORIA 2546A
ACCESO A MEMORIA 2546B
CAMPO CLASE 2568
CLASE B 2568B
CAMPO OPERACIÓN AUMENTO 2550
CAMPO ALFA 2552
CAMPO BETA 2554
CAMPO CONTROL MÁSCARA ESCRITURA 2552C
CAMPO RL 2557A
RMD 2557A.1
VSIZE 2557A.2
CAMPO DIFUSIÓN 2567B
CAMPO ESCALA 2560
DESP. A 2562A
DESP. A R. 2562B
CAMPO ANCHO ELEMENTO DATOS 2564
CAMPO MÁSCARA ESCRITURA 2570
CAMPO INMEDIATO 2572
CAMPO OPERACIÓN RESTRINGIDO 2559A
CAMPO LONGITUD VECTOR 2559B

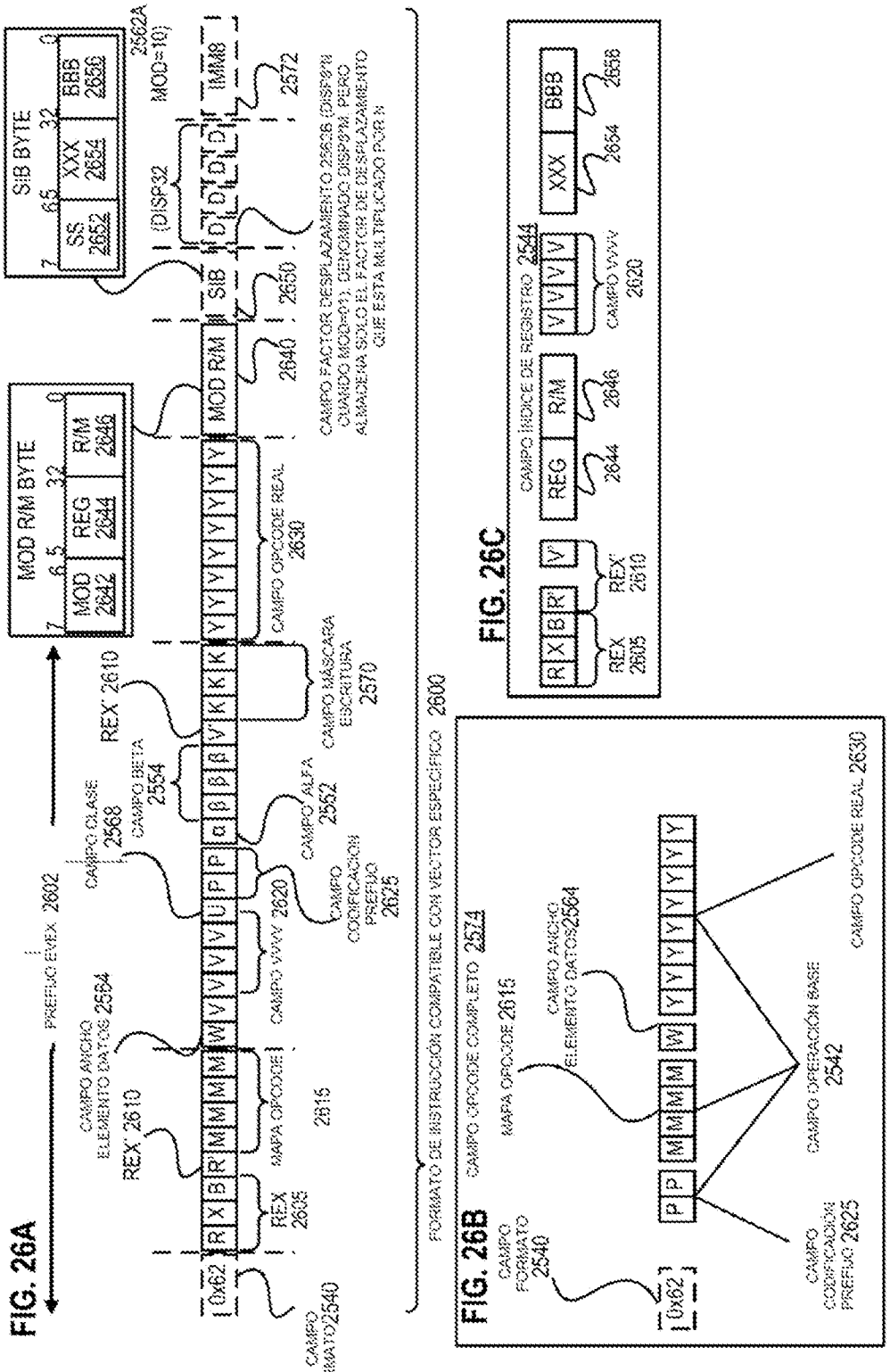


FIG. 26A
FORMATO DE INSTRUCCIÓN COMPATIBLE CON VECTOR ESPECÍFICO 2600
CAMPO FORMATO 2540
PREFIJO EVEX 2602
CAMPO CLASE 2568
CAMPO BETA 2554
REX 2610
CAMPO ANCHO ELEMENTO DATOS 2564
REX 2605
MAPA OPCODE 2615
CAMPO VVVV 2620
CAMPO ALFA 2552
CAMPO CODIFICACIÓN PREFIJO 2625
CAMPO MÁSCARA ESCRITURA 2570
CAMPO OPCODE REAL 2630
MOD R/M BYTE
MOD 2642
REG 2644
R/M 2646
SIB BYTE
SS 2652
XXX 2654
BBB 2656
(DISP32 2562A MOD=10)
SIB 2650
2640
2572
CAMPO FACTOR DESPLAZAMIENTO 2562B (DISP8 CUANDO MOD=01), DENOMINADO DISP8*N, PERO ALMACENA SÓLO EL FACTOR DE DESPLAZAMIENTO QUE ESTÁ MULTIPLICADO POR N
FIG. 26B
CAMPO FORMATO 2540
CAMPO OPCODE COMPLETO 2574
MAPA OPCODE 2615
CAMPO ANCHO ELEMENTO DATOS 2564
CAMPO OPERACIÓN BASE 2542
CAMPO CODIFICACIÓN PREFIJO 2625
CAMPO OPCODE REAL 2630
FIG. 26C
CAMPO ÍNDICE DE REGISTRO 2544
REX 2605
REX 2610
REG 2644
R/M 2646
CAMPO VVVV 2620
XXX 2654
BBB 2656

FIG. 26D

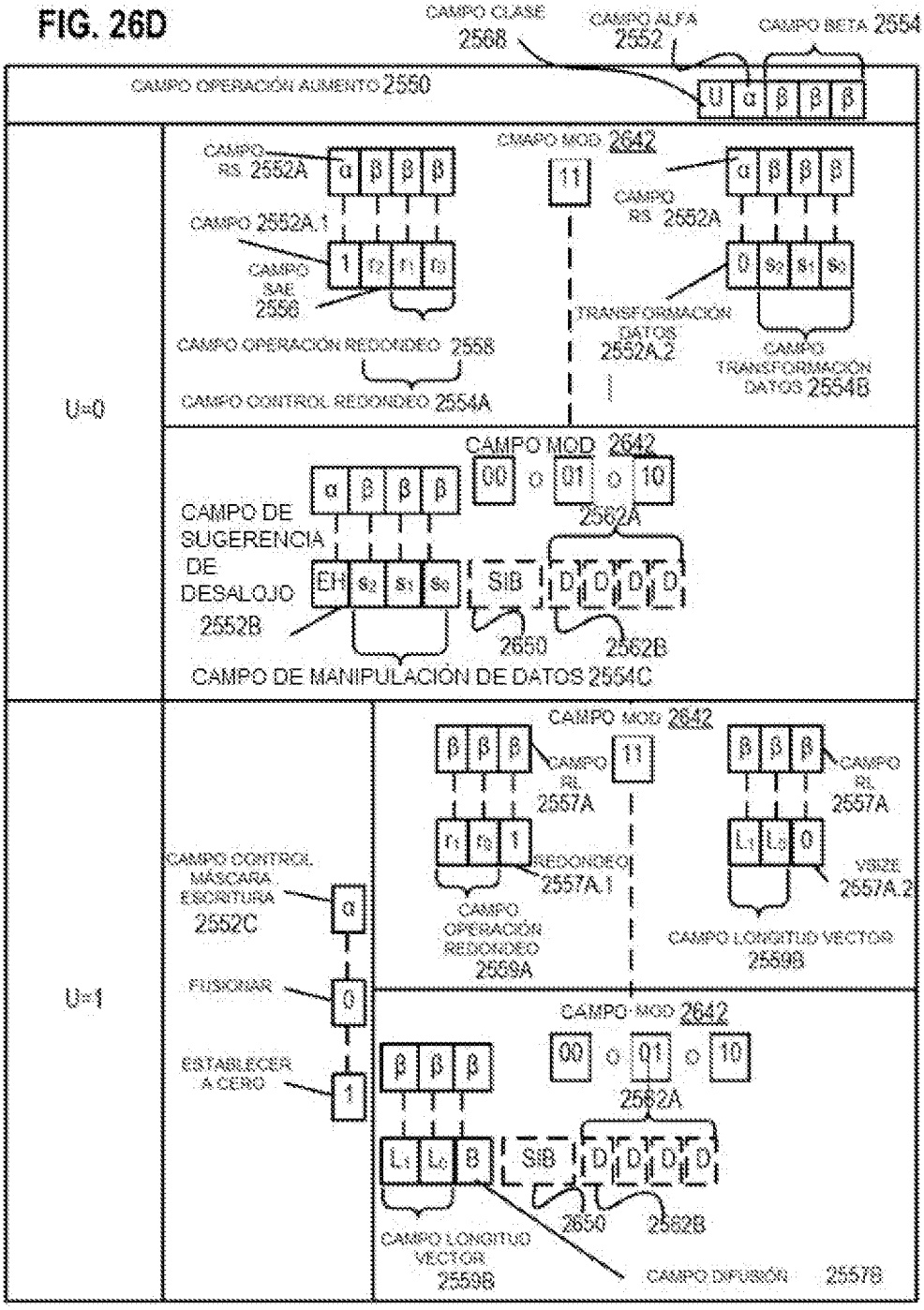
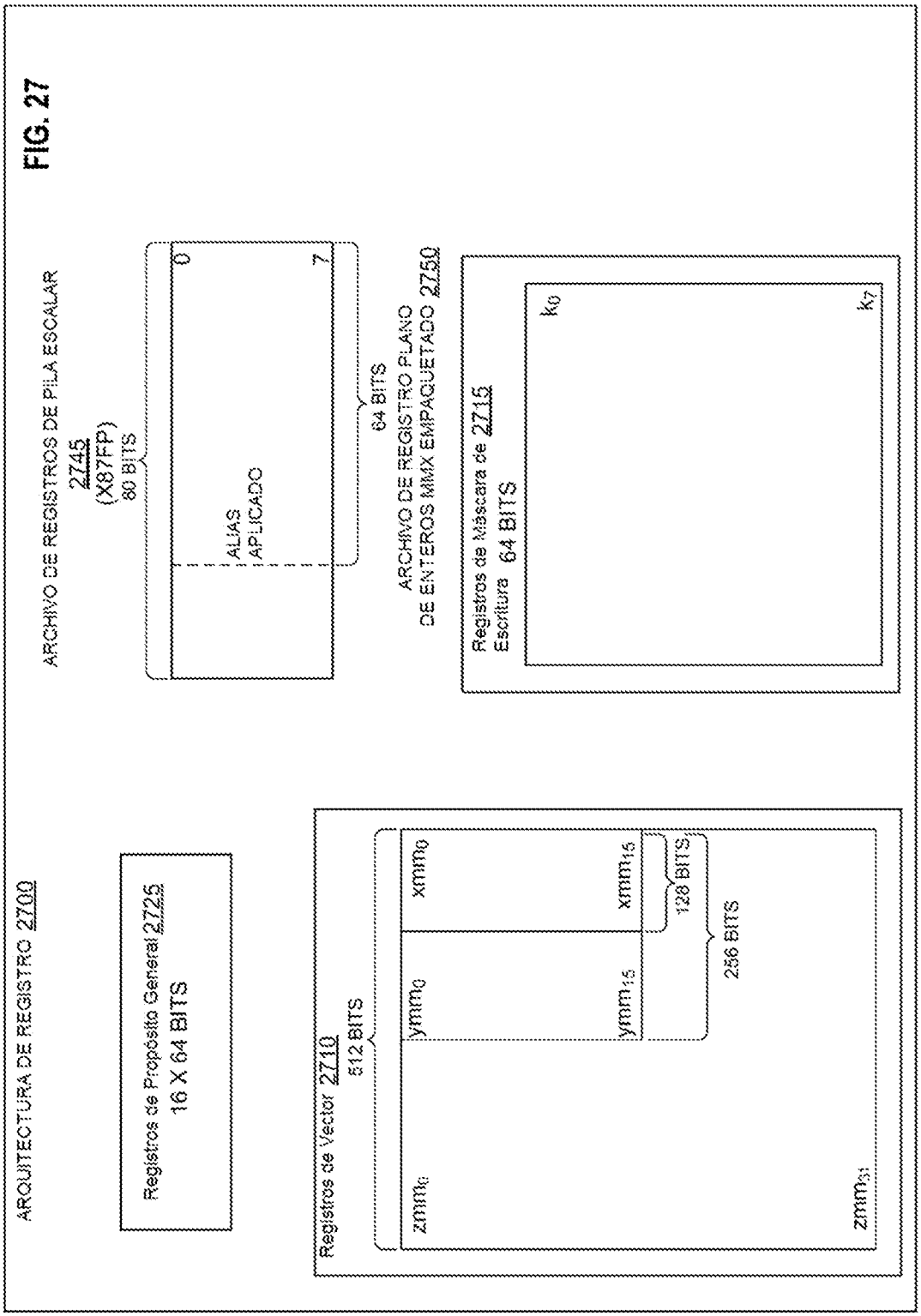

CAMPO CLASE 2568
CAMPO ALFA 2552
CAMPO BETA 2554
CAMPO OPERACIÓN AUMENTO 2550
U
α β β β
U=0
CAMPO RS 2552A
CAMPO 2552A.1
CAMPO SAE 2556
CAMPO OPERACIÓN REDONDEO 2558
CAMPO CONTROL REDONDEO 2554A
CAMPO MOD 2642
11
CAMPO RS 2552A
TRANSFORMACIÓN DATOS 2552A.2
CAMPO TRANSFORMACIÓN DATOS 2554B
CAMPO MOD 2642
00 o 01 o 10
2562A
CAMPO DE SUGERENCIA DE DESALOJO 2552B
EH
SIB
D D D D
2560
2562B
CAMPO DE MANIPULACIÓN DE DATOS 2554C
U=1
CAMPO MOD 2642
CAMPO RL 2557A
CAMPO RL 2557A
REDONDEO 2557A.1
CAMPO OPERACIÓN REDONDEO 2559A
VIEJE 2557A.2
CAMPO LONGITUD VECTOR 2559B
CAMPO CONTROL MÁSCARA ESCRITURA 2552C
FUSIONAR
ESTABLECER A CERO
CAMPO MOD 2642
00 o 01 o 10
2562A
SIB
D D D D
2560
2562B
CAMPO LONGITUD VECTOR 2559B
CAMPO DIFUSIÓN 2557B

FIG. 27


ARQUITECTURA DE REGISTRO 2700
Registros de Propósito General 2725
16 X 64 BITS
ARCHIVO DE REGISTROS DE PILA ESCALAR
2745
(X87FP)
80 BITS
0
7
ALIAS
APLICADO
64 BITS
ARCHIVO DE REGISTRO PLANO
DE ENTEROS MMX EMPAQUETADO 2750
Registros de Máscara de 2715
Escritura 64 BITS
k0
k7
Registros de Vector 2710
512 BITS
zmm0
ymm0
xmm0
ymm15
xmm15
128 BITS
256 BITS
zmm31

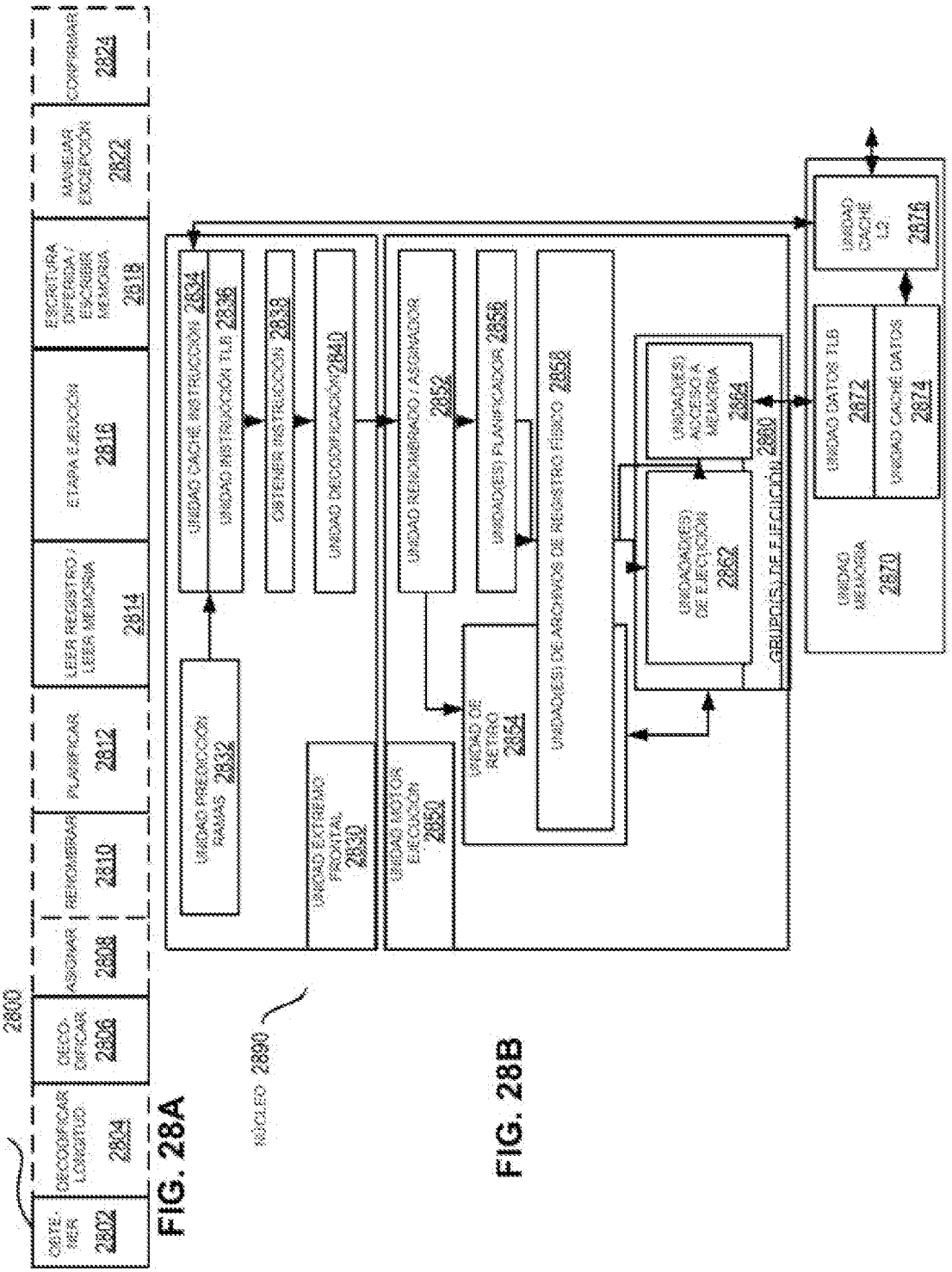


FIG. 28A

FIG. 28B

FIG. 29A

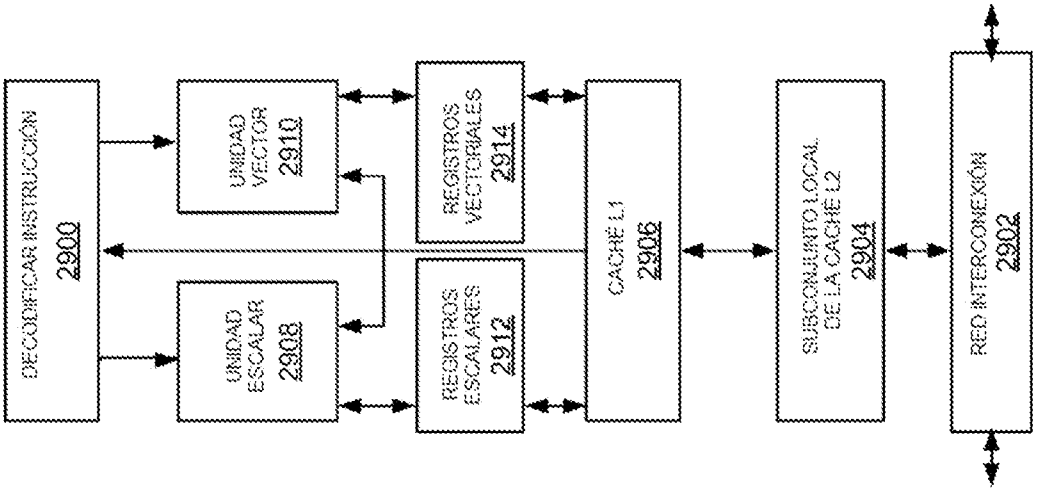
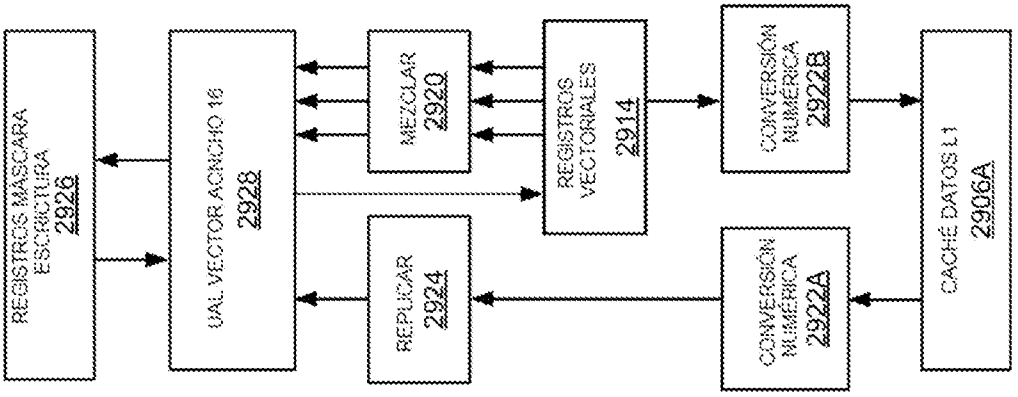


FIG. 29B



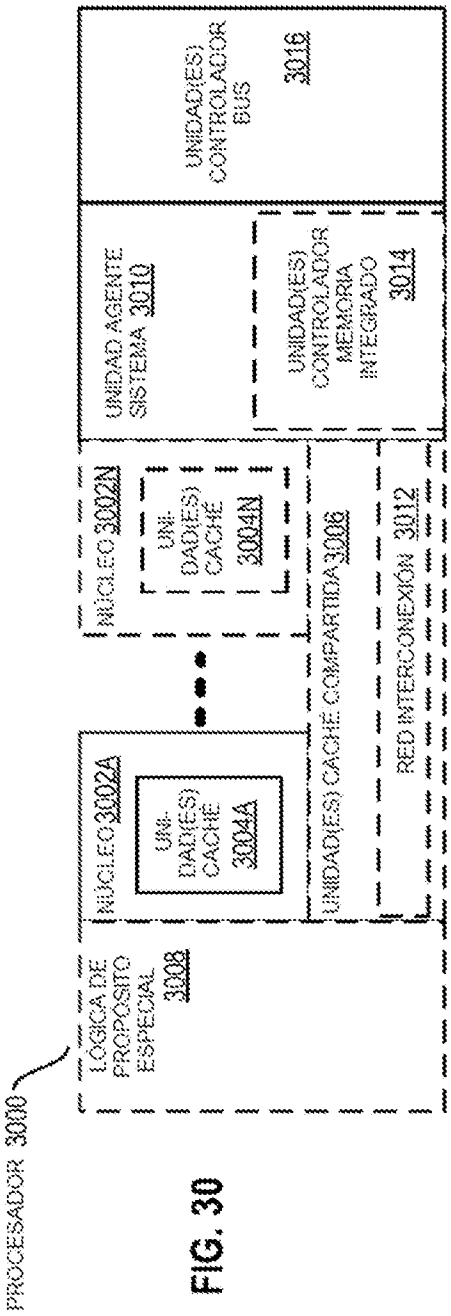

PROCESADOR 3000
LÓGICA DE PROPÓSITO ESPECIAL 3008
NÚCLEO 3002A
UNI-DAD(ES) CACHÉ 3004A
NÚCLEO 3002N
UNI-DAD(ES) CACHÉ 3004N
UNIDAD(ES) CACHÉ COMPARTIDA 3006
RED INTERCONEXIÓN 3012
UNIDAD AGENTE SISTEMA 3010
UNIDAD(ES) CONTROLADOR MEMORIA INTEGRADO 3014
UNIDAD(ES) CONTROLADOR BUS 3016

FIG. 30

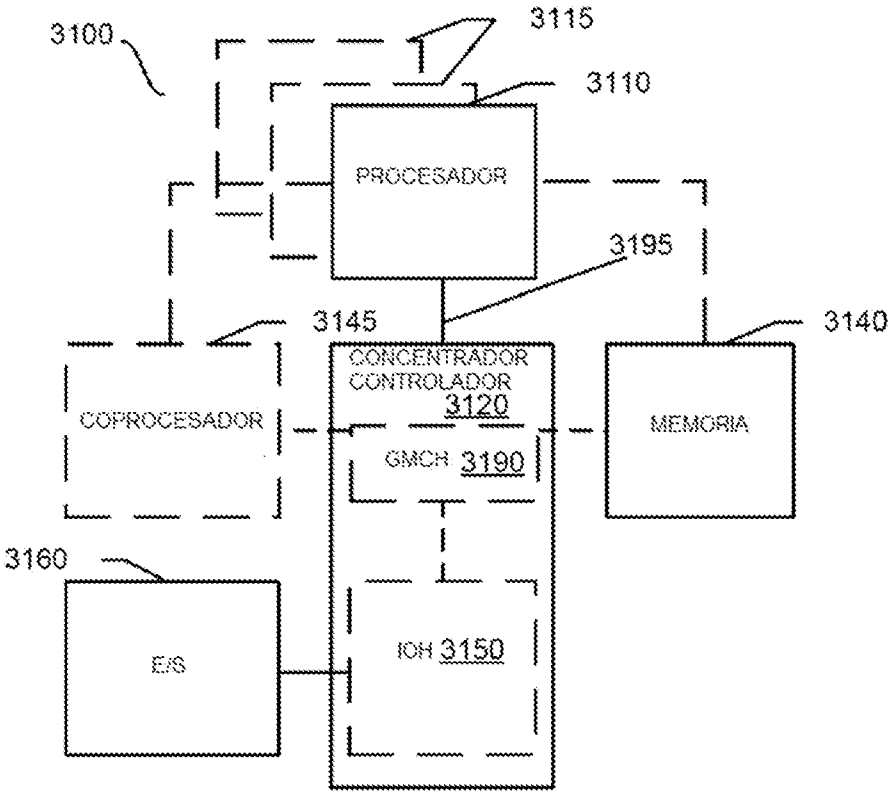

3100
3115
3110
PROCESADOR
3195
3145
3140
CONCENTRADOR
CONTROLADOR
3120
COPROCESADOR
MEMORIA
GMCH 3190
3160
E/S
ICH 3150

FIG. 31

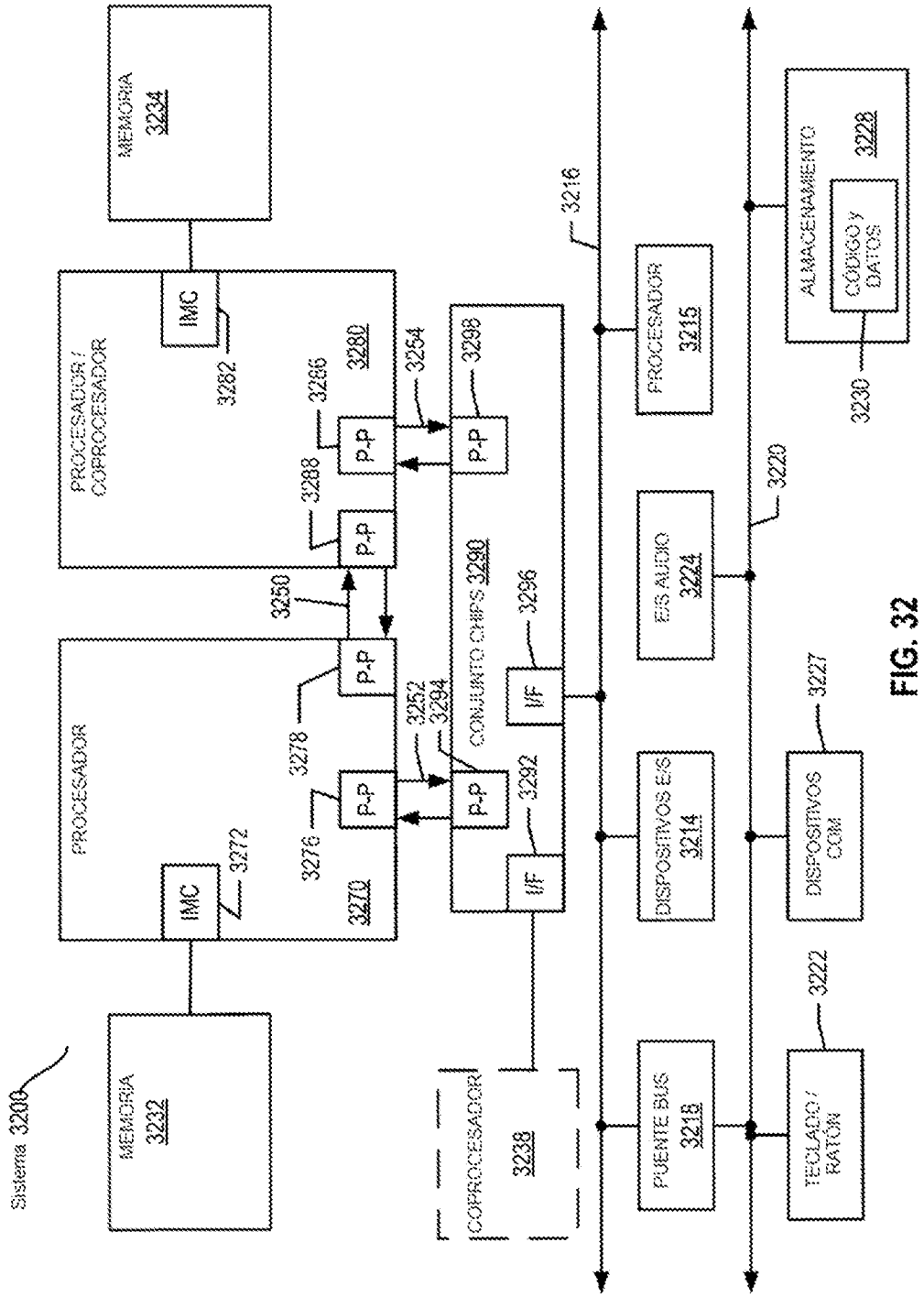

Sistema 3200
MEMORIA 3232
MEMORIA 3234
PROCESADOR 3270
IMC 3272
PROCESADOR / COPROCESADOR
IMC 3282
P-P 3276
P-P 3278
P-P 3288
P-P 3286
3280
3250
3252
3254
3294
3296
3298
CONJUNTO CHIPS 3290
I/F 3292
I/F 3296
COPROCESADOR 3238
3216
PUENTE BUS 3218
DISPOSITIVOS E/S 3214
PROCESADOR 3215
E/S AUDIO 3224
3220
TECLADO / RATÓN 3222
DISPOSITIVOS COM 3227
ALMACENAMIENTO 3228
CÓDIGO Y DATOS
3230

FIG. 32

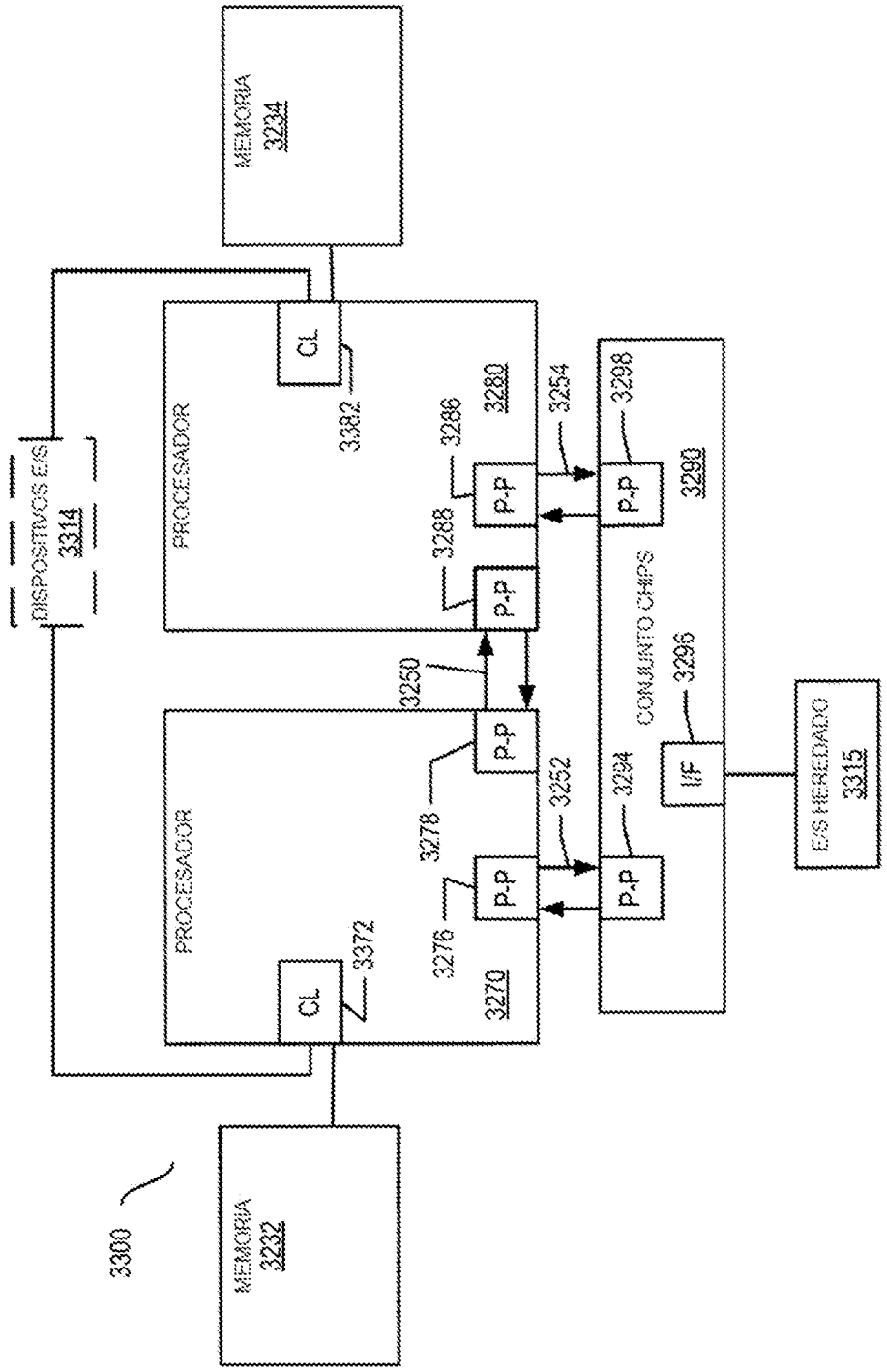

3300
MEMORIA
3232
MEMORIA
3234
DISPOSITIVOS E/S
3314
PROCESADOR
CL
3372
3382
PROCESADOR
3270
3280
3286
3288
3250
3278
3276
3252
3254
P-P
CONJUNTO CHIPS
3290
3294
3296
3298
I/F
E/S HEREDADO
3315

FIG. 33

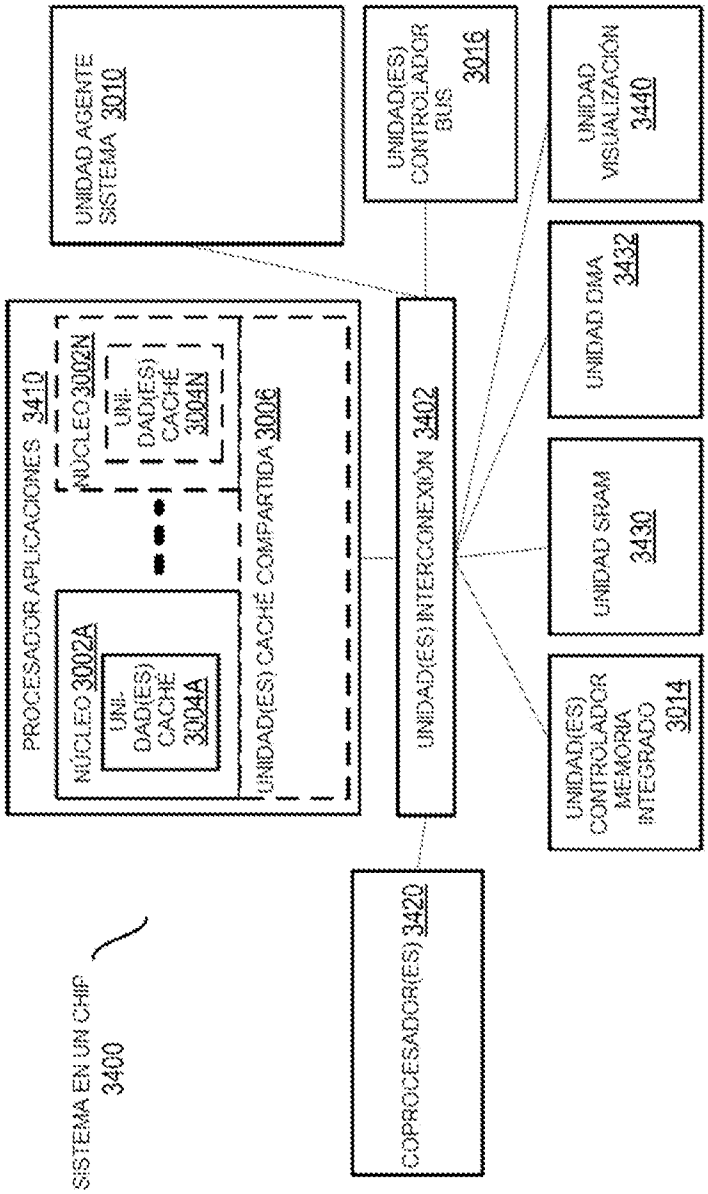

SISTEMA EN UN CHIP
3400
PROCESADOR APLICACIONES 3410
NÚCLEO 3002A
UNI-DAD(ES) CACHÉ 3004A
NÚCLEO 3002N
UNI-DAD(ES) CACHÉ 3004N
UNIDAD(ES) CACHÉ COMPARTIDA 3006
UNIDAD AGENTE SISTEMA 3010
UNIDAD(ES) CONTROLADOR BUS 3016
UNIDAD VISUALIZACIÓN 3440
UNIDAD DMA 3432
UNIDAD SRAM 3430
UNIDAD(ES) CONTROLADOR MEMORIA INTEGRADO 3014
UNIDAD(ES) INTERCONEXIÓN 3402
COPROCESADOR(ES) 3420

FIG. 34

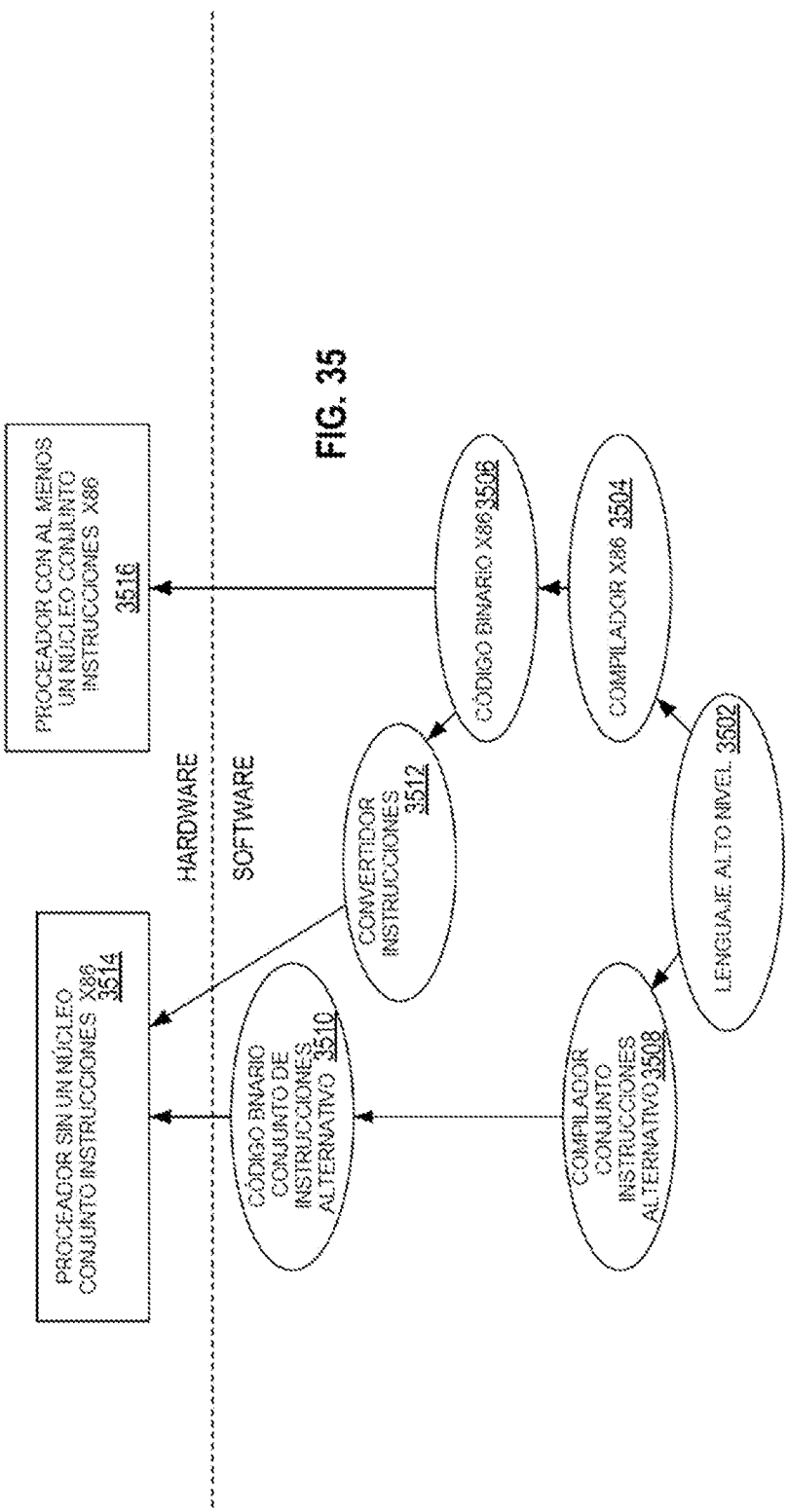

PROCESADOR CON AL MENOS UN NÚCLEO CONJUNTO INSTRUCCIONES X86 3516
HARDWARE
SOFTWARE
PROCESADOR SIN UN NÚCLEO CONJUNTO INSTRUCCIONES X86 3514
CÓDIGO BINARIO CONJUNTO DE INSTRUCCIONES ALTERNATIVO 3510
CONVERTIDOR INSTRUCCIONES 3512
CÓDIGO BINARIO X86 3506
COMPILADOR X86 3504
COMPILADOR CONJUNTO INSTRUCCIONES ALTERNATIVO 3508
LENGUAJE ALTO NIVEL 3502

FIG. 35