

19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **3 027 613**

51 Int. Cl.:

G06F 9/30

(2008.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

96 Fecha de presentación y número de la solicitud europea: **20.02.2020** **E 22208838 (7)**

97 Fecha y número de publicación de la concesión europea: **12.03.2025** **EP 4184319**

54 Título: **Método y aparato para realizar operaciones de reducción sobre una pluralidad de valores de elementos de datos**

30 Prioridad:

27.03.2019 US 201916366155

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

16.06.2025

73 Titular/es:

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

72 Inventor/es:

**HUGHES, CHRISTOPHER;
PEARCE, JONATHAN;
LUEH, GUEI-YUAN;
OULD-AHMED-VALL, ELMOUSTAPHA;
PARRA, JORGE;
SURTI, PRASOONKUMAR;
VINOD, KRISHNA y
ZOHAR, RONEN**

74 Agente/Representante:

LEHMANN NOVO, María Isabel

ES 3 027 613 T3

Aviso: En el plazo de nueve meses a contar desde la fecha de publicación en el Boletín Europeo de Patentes, de la mención de concesión de la patente europea, cualquier persona podrá oponerse ante la Oficina Europea de Patentes a la patente concedida. La oposición deberá formularse por escrito y estar motivada; sólo se considerará como formulada una vez que se haya realizado el pago de la tasa de oposición (art. 99.1 del Convenio sobre Concesión de Patentes Europeas).

DESCRIPCIÓN

Método y aparato para realizar operaciones de reducción sobre una pluralidad de valores de elementos de datos

5 **CAMPO TÉCNICO**

Las realizaciones de la invención se refieren al campo de la arquitectura de procesadores informáticos; y, más específicamente, para realizar operaciones de reducción sobre una pluralidad de valores de elementos de datos.

10 **TÉCNICA ANTERIOR**

Un conjunto de instrucciones, o arquitectura de conjunto de instrucciones (ISA), es la parte de la arquitectura del ordenador relacionada con la programación, que incluye los tipos de datos nativos, las instrucciones, la arquitectura de registro, los modos de direccionamiento, la arquitectura de memoria, el manejo de interrupciones y excepciones, y la entrada y salida externa (E/S). Cabe señalar que el término "instrucción" generalmente se refiere en el presente documento a macroinstrucciones, es decir, instrucciones que se proporcionan al procesador para su ejecución, en contraposición a microinstrucciones o microoperaciones, que son el resultado del decodificador de un procesador decodificando macroinstrucciones. Las microinstrucciones o microoperaciones se pueden configurar para ordenar a una unidad de ejecución en el procesador que realice operaciones para implementar la lógica asociada con la macroinstrucción.

La ISA se distingue de la microarquitectura, que es el conjunto de técnicas de diseño de procesadores utilizadas para implementar el conjunto de instrucciones. Los procesadores con diferentes microarquitecturas pueden compartir un conjunto de instrucciones común. Por ejemplo, los procesadores Intel® Pentium 4, los procesadores Intel® Core™ y los procesadores de Advanced Micro Devices, Inc. de Sunnyvale CA implementan versiones casi idénticas del conjunto de instrucciones x86 (con algunas extensiones que se han agregado con versiones más recientes), pero tienen diferentes diseños internos. Por ejemplo, la misma arquitectura de registro de la ISA puede implementarse de diferentes maneras en diferentes microarquitecturas usando técnicas bien conocidas, incluidos registros físicos dedicados, uno o más registros físicos asignados dinámicamente usando un mecanismo de cambio de nombre de registro (por ejemplo, el uso de un Register Alias Table (RAT), un Reorder Buffer (ROB) y un archivo de registro de retiro). A menos que se especifique lo contrario, las frases arquitectura de registro, archivo de registro y registro se utilizan en el presente documento para referirse a lo que es visible para el software/programador y la manera en que las instrucciones especifican los registros. Cuando se requiera una distinción, se utilizará el adjetivo "lógico", "arquitectónico" o "visible para software" para indicar registros/archivos en la arquitectura de registro, mientras que se utilizarán diferentes adjetivos para designar registros en una microarquitectura determinada (por ejemplo, registro físico, memoria intermedia de reorden, registro de retiro, grupo de registros).

Un conjunto de instrucciones incluye uno o más formatos de instrucciones. Un formato de instrucción dado define diversos campos (número de bits, ubicación de bits) para especificar, entre otras cosas, la operación a realizar y el operando u operandos en los que va a realizarse esa operación. Algunos formatos de instrucción se desglosan aún más mediante la definición de plantillas de instrucción (o subformatos). Por ejemplo, las plantillas de instrucción de un formato de instrucción dado pueden definirse para tener diferentes subconjuntos de los campos del formato de instrucción (los campos incluidos típicamente están en el mismo orden, pero al menos algunos tienen posiciones de bits diferentes porque hay menos campos incluidos) y/o definirse para tener un campo dado interpretado de manera diferente. Una instrucción dada se expresa usando un formato de instrucción dado (y, si está definido, en una de las plantillas de instrucción dadas de ese formato de instrucción) y especifica la operación y los operandos. Un flujo de instrucciones es una secuencia específica de instrucciones, donde cada instrucción en la secuencia es una ocurrencia de una instrucción en un formato de instrucción (y, si se define, una determinada de las plantillas de instrucción de ese formato de instrucción).

El documento US 8.200.940 B1 se refiere a realizar operaciones de reducción en un sistema SIMT (única instrucción de múltiples hilos) con uno o más hilos de ejecución deshabilitados. Por ejemplo, un sistema y método para realizar con éxito operaciones de reducción en un sistema SIMD de múltiples hilos mientras uno o más hilos están deshabilitados permite que las operaciones de reducción se realicen sin una penalización de rendimiento en comparación con realizar la misma operación con todos los hilos habilitados. Los datos de origen para cada cálculo intermedio de la operación de reducción se remapean mediante una barra transversal configurable según sea necesario para evitar el uso de datos no válidos de los hilos deshabilitados. La función de remapeo es transparente para el usuario y posibilita una ejecución correcta de operaciones de reducción invariantes al orden y operaciones de reducción de prefijo dependientes del orden.

El documento US 8.392.669 B1 se refiere a sistemas y métodos para coalescer accesos a memoria de hilos paralelos de un grupo de hilos. Para cada solicitud de aplicación de lectura que atiende un grupo de hilos, la interfaz del núcleo genera una entrada en la tabla de solicitudes pendientes (PRT) y una o más solicitudes de acceso a memoria. La interfaz del núcleo determina el número de solicitudes de acceso a memoria y el tamaño de cada solicitud de acceso a memoria basándose en la dispersión de las direcciones de acceso a memoria en la solicitud de aplicación. Cada solicitud de acceso a memoria especifica los hilos concretos que atiende la solicitud de acceso a memoria. La entrada

de PRT registra el número de solicitudes de acceso a memoria pendientes. A medida que la interfaz de memoria completa cada solicitud de acceso a memoria, la interfaz del núcleo utiliza la información en la solicitud de acceso a memoria y la entrada de PRT correspondiente para enrutar los datos devueltos. Cuando se han completado todas las solicitudes de acceso a memoria asociadas con una entrada de PRT particular, la interfaz del núcleo satisface la solicitud de aplicación correspondiente y libera la entrada de PRT.

SUMARIO

La invención se define en las reivindicaciones independientes. Las reivindicaciones dependientes definen realizaciones de las mismas.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

La invención se puede entender mejor haciendo referencia a la siguiente descripción y a los dibujos adjuntos que se utilizan para mostrar realizaciones de la invención. En los dibujos:

Las **Figuras 1A y 1B** son diagramas de bloques que muestran un formato de instrucción genérico compatible con vectores y plantillas de instrucción del mismo de acuerdo con realizaciones de la invención.

Las **Figuras 2A-D** son diagramas de bloques que muestran un formato de instrucción compatible con vectores específicos a modo de ejemplo de acuerdo con realizaciones de la invención.

La **Figura 3** es un diagrama de bloques de una arquitectura de registro de acuerdo con una realización de la invención.

La **Figura 4A** es un diagrama de bloques que muestra tanto un canal ejemplar en orden como un canal ejemplar de emisión/ejecución desordenada y cambio de nombre de registro de acuerdo con realizaciones de la invención.

La **Figura 4B** es un diagrama de bloques que muestra tanto una realización ejemplar de un núcleo de arquitectura en orden y un núcleo ejemplar de arquitectura de emisión/ejecución desordenada y cambio de nombre de registro que se incluirá en un procesador de acuerdo con realizaciones de la invención.

La **Figura 5A** es un diagrama de bloques de un único núcleo de procesador, junto con su conexión a la red de interconexión integrada y con su subconjunto local de la caché de Nivel 2 (L2), de acuerdo con realizaciones de la invención.

La **Figura 5B** es una vista ampliada de parte del núcleo del procesador en la Figura 5A de acuerdo con realizaciones de la invención.

La **Figura 6** es un diagrama de bloques de un procesador de un solo núcleo y un procesador multinúcleo con controlador de memoria y gráficos integrados de acuerdo con realizaciones de la invención.

La **Figura 7** muestra un diagrama de bloques de un sistema de acuerdo con una realización de la presente invención.

La **Figura 8** muestra un diagrama de bloques de un segundo sistema de acuerdo con una realización de la presente invención.

La **Figura 9** muestra un diagrama de bloques de un tercer sistema de acuerdo con una realización de la presente invención.

La **Figura 10** muestra un diagrama de bloques de un sistema en un chip (SoC) de acuerdo con una realización de la presente invención.

La **Figura 11** muestra 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 en instrucciones binarias en un conjunto de instrucciones de destino de acuerdo con realizaciones de la invención.

La **Figura 12** es un diagrama de bloques que ilustra un sistema informático configurado para implementar uno o más aspectos de las realizaciones descritas en el presente documento.

Las **Figuras 13A-13D** ilustran componentes de procesador paralelo, de acuerdo con una realización.

Las **Figuras 14A-14B** son diagramas de bloques de multiprocesadores de gráficos, de acuerdo con realizaciones.

Las **Figuras 15A-15F** ilustran una arquitectura ilustrativa en la que una pluralidad de GPU están comunicativamente acopladas a una pluralidad de procesadores de múltiples núcleos.

La **Figura 16** ilustra un canal de procesamiento de gráficos, de acuerdo con una realización.

La **Figura 17** muestra operaciones de reducción de árbol sobre una pluralidad de elementos vectoriales almacenados en un registro de entradas tras ejecutar una instrucción única según una realización de la invención.

La **Figura 18** muestra operaciones de reducción sobre una pluralidad de elementos vectoriales almacenados en múltiples registros de entradas tras ejecutar una instrucción única según una realización de la invención.

La **Figura 19** muestra operaciones de reducción sobre una pluralidad de elementos vectoriales almacenados en múltiples registros de entradas tras usar sincronización ejecutando una instrucción única según una realización de la invención.

La **Figura 20** es un diagrama de flujo que muestra operaciones de reducción sobre una pluralidad de elementos vectoriales tras ejecutar una instrucción única según una realización de la invención.

DESCRIPCIÓN DETALLADA

En la siguiente descripción, se exponen numerosos detalles específicos. Sin embargo, se entiende que las realizaciones de la invención se pueden poner en práctica 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.

Se pueden usar texto entre corchetes y bloques con bordes discontinuos (tales como guiones grandes, guiones pequeños, punto-guion y puntos) para ilustrar operaciones opcionales que añaden características adicionales a las realizaciones de la invención. Sin embargo, no se debe interpretar que dicha notación significa que estas son las únicas opciones u operaciones opcionales, y/o que los bloques con bordes sólidos no son opcionales en algunas realizaciones de la invención.

Las referencias en la memoria descriptiva a "una sola realización", "una realización", "una realización de ejemplo", etc., indican que la realización descrita puede incluir una propiedad, estructura o característica particular, pero cada realización puede no necesariamente incluir la propiedad, estructura o característica particular. Además, tales expresiones no se refieren necesariamente a la misma realización. Además, cuando se describe una propiedad, estructura o característica particular en relación con una realización, se afirma que está dentro del conocimiento de un experto en la técnica modificar dicha propiedad, estructura o característica en conexión con otras realizaciones, descritas o no explícitamente.

La siguiente descripción y reivindicaciones pueden utilizar los términos "acoplado" y "conectado", junto con sus derivados. Estos términos no pretenden ser sinónimos entre sí. "Acoplado" se usa para indicar que dos o más elementos, que pueden o no estar en contacto físico o eléctrico directo entre sí, cooperan o interactúan entre sí. "Conectado" se usa para indicar el establecimiento de comunicaciones entre dos o más elementos que están acoplados entre sí. Un "conjunto", tal como se utiliza en el presente documento, se refiere a cualquier número entero positivo de elementos, incluido un elemento.

Un conjunto de instrucciones incluye uno o más formatos de instrucciones. Un formato de instrucción dado define diversos campos (número de bits, ubicación de los bits) para especificar, entre otras cosas, la operación a realizar (código de operación) y los operandos en los que se realizará esa operación. Algunos formatos de instrucción se desglosan aún más mediante la definición de plantillas de instrucción (o subformatos). Por ejemplo, las plantillas de instrucción de un formato de instrucción dado pueden definirse para tener diferentes subconjuntos de los campos del formato de instrucción (los campos incluidos típicamente están en el mismo orden, pero al menos algunos tienen posiciones de bits diferentes porque hay menos campos incluidos) y/o definirse para tener un campo dado interpretado de manera diferente. Por lo tanto, cada instrucción de una ISA se expresa utilizando un formato de instrucción dado (y, si se define, en una de las plantillas de instrucción de ese formato de instrucción) e incluye campos para especificar la operación y los operandos. Por ejemplo, una instrucción ADD de ejemplo tiene un código de operación específico y un formato de instrucción que incluye un campo de código de operación para especificar ese código de operación y campos de operando para seleccionar operandos (fuente1/destino y fuente2); y una aparición de esta instrucción ADD en una secuencia de instrucciones tendrá contenidos específicos en los campos de operandos que seleccionan operandos específicos. Se ha lanzado y/o publicado un conjunto de extensiones SIMD referidas a extensiones vectoriales avanzadas (AVX) (AVX1 y AVX2) y que utiliza el esquema de codificación de extensiones vectoriales (VEX).

Formatos de instrucción ejemplares

Las realizaciones de las instrucciones descritas en el presente documento pueden realizarse en diferentes formatos. Además, a continuación, se detallan sistemas, arquitecturas y canales ilustrativos. Se pueden ejecutar realizaciones de las instrucciones en dichos sistemas, arquitecturas y canales, pero no se limitan a las detalladas.

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 tanto las operaciones vectoriales como las escalares se soportan a través del formato de instrucción compatible con vectores, realizaciones alternativas usan únicamente operaciones vectoriales en el formato de instrucción compatible con vectores.

Las **Figuras 1A-1B** son diagramas de bloques que muestran un formato de instrucción genérico compatible con vectores y plantillas de instrucción del mismo de acuerdo con realizaciones de la invención. La **Figura 1A** es un diagrama de bloques que muestra un formato de instrucción genérico compatible con vectores y plantillas de instrucción de clase A del mismo de acuerdo con realizaciones de la invención; mientras que la **Figura 1B** es un diagrama de bloques que muestra el formato de instrucción genérico compatible con vectores y sus plantillas de instrucción de clase B de acuerdo con realizaciones de la invención. Específicamente, un formato de instrucción compatible con vectores genérico 100 para el que se definen plantillas de instrucción de clase A y clase B, ambas de las cuales incluyen las plantillas de instrucción sin acceso a memoria 105 y las plantillas de instrucción con acceso a memoria 120. El término genérico en el contexto del formato de instrucción compatible con vectores se refiere al formato de instrucción que no está vinculado a ningún conjunto de instrucciones específico.

Si bien se describirán realizaciones de la invención 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 anchuras (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 consiste en 16 elementos de tamaño de palabra doble o, como alternativa, 8 elementos de tamaño de palabra cuádruple); una longitud (o tamaño) de operando de vector de 64 bytes con anchuras (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 anchuras (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 de vector de 16 bytes con anchuras (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 anchuras de elementos de datos (por ejemplo, anchuras de elementos de datos de 128 bits (16 bytes)).

Las plantillas de instrucción de clase A en la **Figura 1A** incluyen: 1) dentro de las plantillas de instrucción sin acceso a memoria 105 se muestra una plantilla de instrucción de operación de tipo de control de redondeo completo sin acceso a memoria 110 y una plantilla de instrucción de operación de tipo de transformación de datos sin acceso a memoria 115; y 2) dentro de las plantillas de instrucción con acceso a memoria 120 se muestra una plantilla de instrucción temporal con acceso a memoria 125 y una plantilla de instrucción no temporal con acceso a memoria 130. Las plantillas de instrucción de clase B en la **Figura 1B** incluyen: 1) dentro de las plantillas de instrucción sin acceso a memoria 105 se muestra una plantilla de instrucción de operación de tipo de control de redondeo parcial, control de máscara de escritura, sin acceso a memoria 112 y una plantilla de instrucción de operación de tipo vsize de control de máscara de escritura sin acceso a memoria 117; y 2) dentro de las plantillas de instrucción con acceso a memoria 120 se muestra una plantilla de instrucción de control de máscara de escritura con acceso a memoria 127.

El formato de instrucción compatible con vectores genérico 100 incluye los siguientes campos enumerados a continuación en el orden mostrado en las **Figuras 1A-1B**.

Campo de formato 140 - 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 ocurrencias 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 de base 142 - su contenido distingue diferentes operaciones de base.

Campo de índice de registro 144 - su contenido, directamente o a través de la generación de direcciones, especifica las ubicaciones de los operandos de origen y destino, ya sea en los registros o en la memoria. Estos incluyen un número suficiente de bits para seleccionar N registros de un archivo de registros PxQ (por ejemplo, 32x512, 16x128, 32x1024, 64x1024). Mientras que en una realización N puede ser hasta tres registros de origen y uno de destino, realizaciones alternativas pueden soportar más o menos registros de origen y de destino (por ejemplo, pueden soportar hasta dos orígenes donde uno de estos orígenes también actúa como el destino, pueden soportar hasta tres orígenes donde uno de estos orígenes también actúa como el destino, puede soportar hasta dos orígenes y un destino).

Campo de modificador 146 - su contenido distingue las ocurrencias de instrucciones en el formato de instrucción de vector genérico que especifican el acceso a memoria de aquellas que no lo hacen; es decir, entre plantillas de instrucción sin acceso a memoria 105 y plantillas de instrucción con acceso a memoria 120. Las operaciones con acceso a memoria leen y/o escriben en la jerarquía de la memoria (en algunos casos, especificando las direcciones de origen y/o destino usando valores en los registros), mientras que las operaciones sin acceso a memoria no lo hacen (por ejemplo, el origen y los destinos son registros). Mientras que en una realización este campo también selecciona

entre tres formas diferentes de realizar cálculos de direcciones de memoria, las realizaciones alternativas pueden soportar más, menos o diferentes formas de realizar cálculos de direcciones de memoria.

5 Campo de operación de aumento 150 - 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 de la invención, este campo se divide en un campo de clase 168, un campo alfa 152 y un campo beta 154. El campo de operación de aumento 150 permite realizar grupos comunes de operaciones en una sola instrucción en lugar de 2, 3 o 4 instrucciones.

10 Campo de escala 160 - 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 $2^{\text{escala}} * \text{índice} + \text{base}$).

15 Campo de desplazamiento 162A - su contenido se usa como parte de la generación de direcciones de memoria (por ejemplo, para la generación de direcciones que usa $2^{\text{escala}} * \text{índice} + \text{base} + \text{desplazamiento}$).

20 Campo de factor de desplazamiento 162B (obsérvese que la yuxtaposición del campo de desplazamiento 162A directamente sobre el campo de factor de desplazamiento 162B 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 utiliza $2^{\text{escala}} * \text{índice} + \text{base} + \text{desplazamiento escalado}$). 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 lo determina el hardware del procesador en tiempo de ejecución en base al campo de código de operación completo 174 (descrito en el presente documento) y el campo de manipulación de datos 154C. El campo de desplazamiento 162A y el campo de factor de desplazamiento 162B son opcionales en el sentido de que no se usan para las plantillas de instrucción sin acceso a memoria 105 y/o diferentes realizaciones pueden implementar únicamente uno o ninguno de los dos.

30 Campo de anchura de elemento de datos 164 - su contenido distingue cuál de un número de anchuras de elementos de datos va a usarse (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 únicamente se soporta una anchura de elemento de datos y/o se soportan anchuras de elementos de datos usando algún aspecto de los códigos de operación.

35 Campo de máscara de escritura 170 - su contenido controla, basándose en la posición del elemento de datos, si esa posición del elemento de datos en el operando de vector de destino refleja el resultado de la operación de base y la operación de aumento. Las plantillas de instrucción de clase A soportan la fusión de máscaras de escritura, mientras que las plantillas de instrucción de clase B soportan tanto la fusión como la puesta a cero de máscaras de escritura. 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 las máscaras vectoriales de puesta a cero permiten poner a cero 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 una realización, un elemento del destino se establece a 0 cuando el bit de máscara correspondiente tiene un valor 0. Un subconjunto de esta funcionalidad es la capacidad de controlar la longitud del vector de la operación que se realiza (es decir, el intervalo de elementos que se modifican, desde el primero hasta el último); sin embargo, no es necesario que los elementos que se modifican sean consecutivos. Por tanto, el campo de máscara de escritura 170 permite operaciones vectoriales parciales, incluyendo cargas, almacenamientos, aritméticas, lógicas, etc. Si bien se describen realizaciones de la invención en las que el contenido del campo de máscara de escritura 170 selecciona uno de un número de registros de máscara de escritura que contiene la máscara de escritura que se va a utilizar (y por lo tanto el contenido del campo de máscara de escritura 170 identifica indirectamente ese enmascaramiento que se va a realizar), realizaciones alternativas en lugar o adicionalmente permiten que el contenido del campo de escritura de máscara 170 especifique directamente el enmascaramiento a realizar.

55 Campo inmediato 172 - 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 compatible con vectores genéricos que no soporta inmediato y no está presente en instrucciones que no usan un inmediato.

60 Campo de clase 168 - su contenido distingue entre diferentes clases de instrucciones. Con referencia a las **Figuras 1A-B**, el contenido de este campo selecciona entre instrucciones de clase A y clase B. En las **Figuras 1A-B**, se utilizan cuadrados de esquinas redondeadas para indicar que un valor específico está presente en un campo (por ejemplo, clase A 168A y clase B 168B para el campo de clase 168 respectivamente en las **Figuras 1A-B**).

65 **Plantillas de instrucción de clase A**

En el caso de las plantillas de instrucción sin acceso a memoria de clase A 105, el campo alfa 152 se interpreta como un campo de RS 152A, cuyo contenido distingue cuál de los diferentes tipos de operación de aumento va a realizarse (por ejemplo, la redondeo 152A.1 y la transformación de datos 152A.2 se especifican respectivamente para las plantillas de instrucción de operación de tipo de redondeo sin acceso a memoria 110 y operación de tipo de transformación de datos sin acceso a memoria 115), mientras que el campo beta 154 distingue cuál de las operaciones del tipo especificado va a realizarse. En las plantillas de instrucción sin acceso a memoria 105, el campo de escala 160, el campo de desplazamiento 162A y el campo de factor de desplazamiento 162B (a veces denominado campo de escala de desplazamiento) no están presentes.

Plantillas de instrucción 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 110, el campo beta 154 se interpreta como un campo de control de redondeo 154A, cuyo contenido o contenidos proporcionan redondeo estático. Si bien en las realizaciones descritas de la invención el campo de control circular 154A incluye un campo suprimir todas las excepciones (coma flotante) (SAE) 156 y un campo de control de operación de redondeo 158, realizaciones alternativas pueden admitir la codificación de ambos conceptos en el mismo campo o solo tener uno u otro de estos conceptos/campos (por ejemplo, puede tener solo el campo de control de operación de redondeo 158).

Campo de SAE 156 - su contenido distingue si se debe o no deshabilitar el informe de eventos de excepción; cuando el contenido del campo de SAE 156 indica que la supresión está habilitada, una instrucción dada no informa ningún tipo de bandera de excepción de coma flotante y no genera ningún manejador de excepción de coma flotante.

Campo de control de operación de redondeo 158 - 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 más cercano). Por lo tanto, el campo de control de operación de redondeo 158 permite el cambio del modo de redondeo por instrucción. En una realización de la invención donde un procesador incluye un registro de control para especificar modos de redondeo, el contenido del campo de control de operación de redondeo 158 anula ese valor de registro.

Plantillas de instrucción sin acceso a memoria - operación de tipo de transformación de datos

En la plantilla de instrucción de operación de tipo de transformación de datos sin acceso a memoria 115, el campo beta 154 se interpreta como un campo de transformación de datos 154B, 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 120 de clase A, el campo alfa 152 se interpreta como un campo de sugerencia de desalojo 152B, cuyo contenido distingue cuál de las sugerencias de desalojo se va a usar (en la **Figura 1A**, temporal 152B.1 y no temporal 152B.2 se especifican respectivamente para la plantilla de instrucción de acceso a memoria, temporal 125 y la plantilla de instrucción de acceso a memoria, no temporal 130), mientras que el campo beta 154 se interpreta como un campo de manipulación de datos 154C, 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 instrucción con acceso a memoria 120 incluyen el campo de escala 160 y, opcionalmente, el campo de desplazamiento 162A o el campo de factor de desplazamiento 162B.

Las instrucciones de memoria vectoriales realizan cargas y almacenes de vectores en la memoria, con soporte de conversión. Al igual que con las instrucciones vectoriales normales, las instrucciones de memoria vectorial transfieren datos desde/hacia la memoria en forma de elementos de datos, y los elementos que realmente se transfieren están dictados por el contenido de la máscara vectorial que se selecciona como máscara de escritura.

Plantillas de instrucción con acceso a memoria - temporal

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 instrucción con acceso a memoria - no temporal

Los datos no temporales son datos que es poco probable que se reutilicen lo suficientemente pronto como para beneficiarse del almacenamiento en caché en la caché de 1^{er} nivel y se les debe dar prioridad para la expulsión. Sin embargo, esto es una sugerencia, y diferentes procesadores pueden implementarla de diferentes maneras, incluso ignorando la sugerencia por completo.

Plantillas de instrucción de clase B

En el caso de las plantillas de instrucción de clase B, el campo alfa 152 se interpreta como un campo de control de máscara de escritura (Z) 152C, cuyo contenido distingue si el enmascaramiento de escritura controlado por el campo de máscara de escritura 170 debe ser una fusión o una puesta a cero.

En el caso de las plantillas de instrucción sin acceso a memoria 105 de clase B, parte del campo beta 154 se interpreta como un campo RL 157A, cuyo contenido distingue cuál de los diferentes tipos de operación de aumento se van a realizar (por ejemplo, redondeo 157A.1 y la longitud del vector (VSIZE) 157A.2 se especifican respectivamente para la plantilla de instrucción sin acceso a memoria, control de máscara de escritura, plantilla de instrucción de operación de tipo de control de redondeo parcial 112 y sin acceso a memoria, control de máscara de escritura, operación de tipo VSIZE 117), mientras que el resto del campo beta 154 distingue cuál de las operaciones del tipo especificado se va a realizar. En las plantillas de instrucción sin acceso a memoria 105, el campo de escala 160, el campo de desplazamiento 162A y el campo de factor de desplazamiento 162B no están presentes.

En la plantilla de instrucción sin acceso a memoria, control de máscara de escritura, operación de tipo de control de redondeo parcial 110, el resto del campo beta 154 se interpreta como un campo de control de operación de redondeo 159A y el informe de eventos de excepción está deshabilitado (una instrucción dada no informa un tipo de la bandera de excepción de punto flotante y no genera un controlador de excepción de punto flotante).

Campo de control de operación de redondeo 159A - al igual que el campo de control de operación de redondeo 158, 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 más cercano). Por lo tanto, el campo de control de operación de redondeo 159A permite el cambio del modo de redondeo por instrucción. En una realización de la invención donde un procesador incluye un registro de control para especificar modos de redondeo, el contenido del campo de control de operación de redondeo 159 anula ese valor de registro.

En la plantilla de instrucción de operación de tipo VSIZE 117, control de máscara de escritura, sin acceso a memoria, el resto del campo beta 154 se interpreta como un campo de longitud de vector 159B, cuyo contenido distingue cuál de varias 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 120 de clase B, parte del campo beta 154 se interpreta como un campo de difusión 157B, 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 154 se interpreta como el campo de longitud vectorial 159B. Las plantillas de instrucción con acceso a memoria 120 incluyen el campo de escala 160 y, opcionalmente, el campo de desplazamiento 162A o el campo de escala de desplazamiento 162B.

Con respecto al formato de instrucción compatible con vectores genéricos 100, se muestra un campo de código de operación completo 174 que incluye el campo de formato 140, el campo de operación de base 142 y el campo de anchura de elemento de datos 164. Aunque se muestra una realización donde el campo de código de operación completo 174 incluye todos estos campos, el campo de código de operación completo 174 incluye menos de todos estos campos en realizaciones que no los soportan todos. El campo de código de operación completo 174 proporciona el código de operación (opcode).

El campo de operación de aumento 150, el campo de anchura de elemento de datos 164 y el campo de máscara de escritura 170 permiten que estas características se especifiquen en una base por instrucciones en el formato de instrucción compatible con vectores genérico.

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

Las diversas plantillas de instrucción que se encuentran dentro de la clase A y la clase B son beneficiosas en diferentes situaciones. En algunas realizaciones de la invención, 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 desordenado de propósito general de alto rendimiento destinado a computación de propósito general puede soportar solo la clase B, un núcleo destinado principalmente a gráficos y/o computación científica (rendimiento) puede soportar solo la clase A, y un núcleo destinado a ambas puede admitir ambas (por supuesto, un núcleo que tenga alguna combinación de plantillas e instrucciones de ambas clases, pero no todas las plantillas e instrucciones de ambas clases está dentro del alcance de la invención). Además, un solo procesador puede incluir múltiples núcleos, todos los cuales soportan la misma clase o en los que diferentes núcleos soportan diferentes clases. Por ejemplo, en un procesador con gráficos separados y núcleos de uso general, uno de los núcleos de 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 de alto rendimiento con ejecución fuera de orden y cambio de nombre de registro destinados a computación de propósito general que soportan solo clase B. Otro procesador que no tiene un núcleo de gráficos separado, puede incluir uno más núcleos de propósito general ordenados o desordenados que soportan 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 de la invención. 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, que incluyen: 1) una forma que tiene únicamente instrucciones de la clase o clases soportadas por el procesador objetivo para su ejecución; o 2) una forma que tiene rutinas alternativas escritas usando diferentes combinaciones de las instrucciones de todas las clases y que tiene un código de flujo de control que selecciona las rutinas a ejecutar basándose en las instrucciones soportadas por el procesador que actualmente está ejecutando el código.

Las **Figuras 2A-D** son diagramas de bloques que muestran un formato de instrucción compatible con vectores específico ilustrativo de acuerdo con realizaciones de la invención. La **Figura 2** muestra un formato de instrucción compatible con vectores específico 200 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 específicos 200 se puede usar para ampliar el conjunto de instrucciones x86 y, por lo tanto, algunos de los campos son similares o iguales a aquellos usados en el conjunto de instrucciones x86 existente y la extensión del mismo (por ejemplo, AVX). Este formato sigue siendo consistente con el campo de codificación de prefijo, el campo de bytes de código de operación 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. Se muestran los campos de la **Figura 1** a los que se mapean los campos de la **Figura 2**.

Debe entenderse que, aunque las realizaciones de la invención se describen con referencia al formato de instrucción compatible con vectores específicos 200 en el contexto del formato de instrucción compatible con vectores genérico 100 con fines ilustrativos, la invención no se limita al formato de instrucción compatible con vectores específico. 200 excepto donde se reivindique. Por ejemplo, el formato de instrucción compatible con vectores genérico 100 contempla una diversidad de tamaños posibles para los diversos campos, mientras que el formato de instrucción compatible con vectores específicos 200 se muestra teniendo campos de tamaños específicos. A modo de ejemplo específico, aunque el campo de ancho de elemento de datos 164 se muestra como un campo de un bit en el formato de instrucción compatible con vectores específicos 200, la invención no está tan limitada (es decir, el formato de instrucción genérico compatible con vectores 100 contempla otros tamaños del campo de ancho del elemento de datos 164).

El formato de instrucción compatible con vectores genérico 100 incluye los siguientes campos enumerados a continuación en el orden mostrado en la **Figura 2A**.

Prefijo EVEX (Bytes 0-3) 202: está codificado en formato de cuatro bytes.

Campo de formato 140 (EVEX Byte 0, bits [7:0]) - el primer byte (EVEX Byte 0) es el campo de formato 140 y contiene 0x62 (el valor único usado para distinguir el formato de instrucción compatible con vectores en una realización de la invención).

El segundo-cuarto bytes (EVEX Bytes 1-3) incluyen un número de campos de bits que proporcionan una capacidad específica.

Campo REX 205 (EVEX byte 1, bits [7-5]): consta de un campo de bits EVEX.R (EVEX byte 1, bit [7] - R), campo de bits EVEX.X (EVEX byte 1, bit [6] - X), y 1157BEX 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' 210 - esta es la primera parte del campo REX' 210 y es el campo de bits EVEX.R' (EVEX Byte 1, bit [4] - R') que se usa para codificar los 16 superiores o los 16 inferiores del conjunto de 32 registros extendido. En una realización de la invención, este bit, junto con otros como se indica a continuación, se almacena en formato de bits invertidos para distinguir (en el bien conocido modo x86 de 32 bits) de la instrucción BOUND, cuyo byte de código de operación 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 de la invención no almacenan este 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 215 (EVEX byte 1, bits [3:0] - mmmm): su contenido codifica un byte de código de operación inicial implícito (0F, 0F 38 o 0F 3).

El campo de anchura de elemento de datos 164 (EVEX byte 2, bit [7] - W) - se representa mediante 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 220 (EVEX Byte 2, bits [6:3]-VVVV): el rol de EVEX.VVVV puede incluir lo siguiente: 1) EVEX.VVVV codifica el primer operando del registro fuente, especificado en forma invertida (complemento a 1) y es válido para instrucciones con 2 o más operandos fuente; 2) EVEX.VVVV codifica el operando del registro de destino, especificado en forma de complemento a 1 para ciertos desplazamientos vectoriales; o 3) EVEX.VVVV no codifica un operando, el

campo está reservado y debe contener 1111b. Por lo tanto, el campo EVEX.VVVV 220 codifica los 4 bits de orden inferior del primer especificador de registro fuente almacenado en forma invertida (complemento a 1). Dependiendo de la instrucción, se usa un campo de bits EVEX adicional diferente para extender el tamaño del especificador a 32 registros.

EVEX.U 168 campo de Clase (EVEX byte 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 225 (EVEX byte 2, bits [1:0]-pp) - proporciona bits adicionales para el campo de operación de base. Además de proporcionar soporte para las instrucciones SSE heredadas en el formato de prefijo EVEX, esto también tiene la ventaja de compactar el prefijo de SIMD (en lugar de requerir un byte para expresar el prefijo de SIMD, el prefijo de 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 al PLA del decodificador (para que el 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 del código de operación, ciertas realizaciones se expanden de manera similar para mantener la consistencia, pero permiten que estos prefijos de SIMD heredados especifiquen diferentes significados. 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 152 (EVEX byte 3, bit [7] - EH; también conocido como EVEX.EH, EVEX.rs, EVEX.RL, control de máscara de escritura EVEX y EVEX.N; también se muestra con α), como se describió anteriormente, este campo es específico del contexto.

Campo beta 154 (EVEX byte 3, bits [6:4]-SSS, también conocido como EVEX.s₂₋₀, EVEX.r₂₋₀, EVEX.rr1, EVEX.LL0, EVEX.LLB; también se muestra con $\beta\beta\beta$), como se describió anteriormente, este campo es específico del contexto.

Campo REX' 210 - este es el resto del campo REX' y es el campo de bits EVEX.V' (EVEX Byte 3, bit [3] - V') que se puede usar para codificar los 16 superiores o los 16 inferiores del conjunto de 32 registros extendido. Este bit se almacena en formato de bits invertidos. 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 170 (EVEX byte 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 de la invención, el valor específico EVEX.kkk=000 tiene un comportamiento especial que implica que no se usa máscara de escritura alguna para la instrucción particular (esto se puede implementar de una diversidad de formas, incluyendo el uso de una máscara de escritura cableada físicamente a todo unos o hardware que sortea el hardware de enmascaramiento).

El campo de código de operación real 230 (byte 4) también se conoce como byte de código de operación. Parte del código de operación se especifica en este campo.

El campo MOD R/M 240 (byte 5) incluye el campo MOD 242, el campo Reg 244 y el campo R/M 246. Como se describió anteriormente, el contenido del campo MOD 242 distingue entre operaciones con acceso a memoria y operaciones sin acceso a memoria. La función del campo Reg 244 se puede resumir en dos situaciones: codificar el operando del registro de destino o un operando del registro de origen, o tratarse como una extensión de código de operación y no usarse para codificar un operando de instrucción. La función del campo R/M 246 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.

Byte de escala, índice, base (SIB) (Byte 6): como se describió anteriormente, el contenido el campo de escala 150 se utiliza para la generación de direcciones de memoria. SIB.xxx 254 y SIB.bbb 256 - el contenido de estos campos ha sido mencionado anteriormente con respecto a los índices de registro Xxxx y Bbbb.

Campo de desplazamiento 162A (Bytes 7-10) - cuando el campo MOD 242 contiene 10, los bytes 7-10 son el campo de desplazamiento 162A, y funciona igual que el desplazamiento de 32 bits heredado (disp32) y funciona con granularidad de byte.

Campo de factor de desplazamiento 162B (Byte 7) - cuando el campo MOD 242 contiene 01, el byte 7 es el campo de factor de desplazamiento 162B. 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. Dado que disp8 es un signo extendido, solo puede direccionar entre -128 y 127 compensaciones de bytes; en términos de líneas de caché de 64 bytes, disp8 usa 8 bits que se pueden establecer en solo cuatro valores realmente útiles: -128, -64, 0 y 64; dado que 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 162B es una reinterpretación de disp8; cuando se usa el campo de

factor de desplazamiento 162B, 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 $\text{disp8} \times N$. Esto reduce la longitud de instrucción promedio (un solo byte de lo usado para el desplazamiento, pero con un rango mucho mayor). Tal desplazamiento comprimido se basa en la suposición de que el desplazamiento efectivo es un múltiplo de la granularidad del acceso a memoria y, por lo tanto, no es necesario codificar los bits de bajo orden redundantes de la compensación de dirección. En otras palabras, el campo de factor de desplazamiento 162B sustituye el desplazamiento de 8 bits del conjunto de instrucciones x86 heredado. Por lo tanto, el campo de factor de desplazamiento 162B 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 $\text{disp8} \times N$. En otras palabras, no hay cambios en las reglas de codificación o longitudes de codificación, sino solo en la interpretación del valor de desplazamiento por el hardware (que necesita escalar el desplazamiento por el tamaño del operando de memoria para obtener una compensación de dirección por bytes).

El campo inmediato 172 funciona como se ha descrito anteriormente.

Campo de opcode completo

La **Figura 2B** es un diagrama de bloques que muestra los campos del formato de instrucción compatible con vectores específicos 200 que constituyen el campo de código de operación completo 174 de acuerdo con una realización de la invención. Específicamente, el campo de código de operación completo 174 incluye el campo de formato 140, el campo de operación de base 142 y el campo de anchura de elemento de datos (W) 164. El campo de operación de base 142 incluye el campo de codificación de prefijo 225, el campo de mapa de código de operación 215 y el campo de código de operación real 230.

Campo de índice de registro

La **Figura 2C** es un diagrama de bloques que muestra los campos del formato de instrucción compatible con vectores específicos 200 que componen el campo de índice de registro 144 de acuerdo con una realización de la invención. Específicamente, el campo de índice de registro 144 incluye el campo REX 205, el campo REX' 210, el campo MODR/M.reg 244, el campo MODR/M.r/m 246, el campo VVVV 220, el campo xxx 254 y el campo bbb 256.

Campo de operación de aumento

La **Figura 2D** es un diagrama de bloques que muestra los campos del formato de instrucción compatible con vectores específicos 200 que componen el campo de operación de aumento 150 de acuerdo con una realización de la invención. Cuando el campo clase (U) 168 contiene 0, significa EVEX.U0 (clase A 168A); cuando contiene 1, significa EVEX.U1 (clase B 168B). Cuando U=0 y el campo MOD 242 contiene 11 (lo que significa una operación sin acceso a memoria), el campo alfa 152 (EVEX byte 3, bit [7] - EH) se interpreta como el campo de rs 152A. Cuando el campo de RS 152A contiene un 1 (redondeo 152A.1), el campo beta 154 (EVEX byte 3, bits [6:4]-SSS) se interpreta como el campo de control de redondeo 154A. El campo de control de redondeo 154A incluye un campo SAE 156 de un bit y un campo de operación de redondeo 158 de dos bits. Cuando el campo rs 152A contiene un 0 (transformación de datos 152A.2), el campo beta 154 (byte 3 EVEX, bits [6:4]- SSS) se interpreta como un campo de transformación de datos de tres bits 154B. Cuando U=0 y el campo MOD 242 contiene 00, 01 o 10 (lo que significa una operación con acceso a memoria), el campo alfa 152 (EVEX byte 3, bit [7] - EH) se interpreta como el campo de sugerencia de expulsión (EH) 152B y el campo beta 154 (EVEX byte 3, bits [6:4]-SSS) se interpreta como un campo de manipulación de datos de tres bits 154C.

Cuando U=1, el campo alfa 152 (EVEX byte 3, bit [7] - EH) se interpreta como el campo de control de máscara de escritura (Z) 152C. Cuando U=1 y el campo MOD 242 contiene 11 (lo que significa una operación sin acceso a memoria), parte del campo beta 154 (byte de EVEX 3, bit [4] - S₀) se interpreta como el campo de RL 157A; cuando contiene un 1 (redondeo 157A.1) el resto del campo beta 154 (byte de EVEX 3, bit [6:5] - S₂₋₁) se interpreta como el campo de operación de redondeo 159A, mientras que cuando el campo de RL 157A contiene un 0 (VSIZE 157.A2) el resto del campo beta 154 (byte de EVEX 3, bit [6:5] - S₂₋₁) se interpreta como el campo de longitud de vector 159B (byte de EVEX 3, bit [6:5] - L₁₋₀). Cuando U=1 y el campo MOD 242 contiene 00, 01 o 10 (lo que significa una operación con acceso a memoria), el campo beta 154 (EVEX byte 3, bits [6:4]- SSS) se interpreta como el campo de longitud de vector 159B (EVEX byte 3, bit [6:5]- L₁₋₀) y el campo de difusión 157B (EVEX byte 3, bit [4]- B).

La **Figura 3** es un diagrama de bloques de una arquitectura de registro 300 de acuerdo con una realización de la invención. En la realización mostrada, hay 32 registros vectoriales 310 que tienen 512 bits de ancho; a estos registros se hace referencia como zmm0 a zmm31. Los 256 bits de orden inferior de los 16 registros zmm inferiores se superponen en 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 en los registros xmm0-15. El formato de instrucción compatible con vectores específicos 200 opera en estos archivos de registro superpuestos como se muestra en la siguiente tabla.

Longitud vectorial ajustable	Clase	Operaciones	Registros
Plantillas de instrucción que no incluyen el campo de longitud de vector 159B	A (Figura 1A; U=0)	110, 115, 125, 130	registros zmm (la longitud del vector es de 64 bytes)
	B (Figura 1B; U=1)	112	registros zmm (la longitud del vector es de 64 bytes)
Plantillas de instrucción que incluyen el campo de longitud de vector 159B	B (Figura 1B; U=1)	117, 127	registros zmm, ymm o xmm (la longitud del vector es de 64 bytes, 32 bytes o 16 bytes) dependiendo del campo de longitud del vector 159B

En otras palabras, el campo de longitud de vector 159B 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 de la longitud precedente; y las plantillas de instrucción sin el campo de longitud de vector 159B operan en la longitud de vector máxima. Además, en una realización, las plantillas de instrucción de clase B del formato de instrucción compatible con vectores específicos 200 operan en datos de coma flotante de precisión sencilla/doble escalar o empaquetados y datos de entero escalar o empaquetados. 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 315 - en la realización mostrada, 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 315 tienen un tamaño de 16 bits. Como se describió anteriormente, en una realización de la invención, el registro de máscara vectorial k0 no se puede utilizar como máscara de escritura; cuando la codificación que normalmente indicaría k0 se usa para una máscara de escritura, selecciona una máscara de escritura cableada de 0xFFFF, deshabilitando efectivamente la máscara de escritura para esa instrucción.

Registros de propósito general 325 - en la realización mostrada, hay dieciséis registros de propósito general de 64 bits que se usan 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) 345, en el que se asigna un alias al archivo de registro plano entero empaquetado MMX 350; en la realización mostrada, la pila x87 es una pila de ocho elementos utilizada para realizar operaciones escalares de punto flotante 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 con datos enteros empaquetados de 64 bits, así como para contener operandos para algunas operaciones realizadas entre los registros MMX y XMM.

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

Arquitecturas de procesador y tipos de datos ejemplares

La **Figura 4A** es un diagrama de bloques que muestra tanto un canal ejemplar en orden como un canal ejemplar de emisión/ejecución desordenada y cambio de nombre de registro de acuerdo con realizaciones de la invención. La **Figura 4B** es un diagrama de bloques que muestra tanto una realización ejemplar de un núcleo de arquitectura en orden y un núcleo ejemplar de arquitectura de emisión/ejecución desordenada y cambio de nombre de registro que se incluirá en un procesador de acuerdo con realizaciones de la invención. Los recuadros con líneas continuas en las **Figuras 4A-B** muestran la canalización ordenada y el núcleo ordenado, mientras que la adición opcional de los recuadros con líneas discontinuas muestra el cambio de nombre del registro, la canalización y el núcleo de emisión/ejecución desordenada. Dado que el aspecto ordenado es un subconjunto del aspecto desordenado, se describirá el aspecto desordenado.

En la **Figura 4A**, una canalización de procesador 400 incluye una etapa de recuperación 402, una etapa de decodificación de longitud 404, una etapa de decodificación 406, una etapa de asignación 408, una etapa de cambio de nombre 10, una etapa de planificación (también conocida como despacho o emisión) 12, una etapa de lectura de registro/lectura de memoria 14, una etapa de ejecución 16, una etapa de reescritura/escritura de memoria 18, una etapa de manejo de excepciones 22 y una etapa de confirmación 24. Cada etapa puede incluir circuitos para realizar las operaciones en la etapa, y los circuitos puede denominarse circuito de búsqueda, circuito de decodificación, circuito de programación, circuito de ejecución, etc. para indicar que se utilizan circuitos de hardware para implementar estas etapas.

La **Figura 4B** muestra el núcleo de procesador 490 que incluye una unidad de extremo frontal 430 acoplada a una unidad de motor de ejecución 450, y ambas están acopladas a una unidad de memoria 470. El núcleo 490 puede ser un núcleo de computación de conjunto de instrucciones reducido (RISC), un conjunto de instrucciones complejo de

computación (CISC), un núcleo de palabra de instrucción muy larga (VLIW) o un tipo de núcleo híbrido o alternativo. Como otra opción más, el núcleo 490 puede ser un núcleo de propósito especial, tal como, por ejemplo, un núcleo de red o comunicación, motor de compresión, núcleo de coprocesador, núcleo de unidad de procesamiento de gráficos de cálculo de propósito general (GPGPU), núcleo de gráficos o similares.

La unidad frontal 430 incluye una unidad de predicción de rama 432 acoplada a una unidad de caché de instrucciones 434, que está acoplada a una memoria intermedia de recuperación de traducción de instrucciones (TLB) 436, que está acoplada a una unidad de recuperación de instrucción 438, que está acoplada a una unidad de decodificación 440. La unidad de decodificación 440 (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, o que de otro modo reflejan o se derivan de las instrucciones originales. La unidad de decodificación 440 puede implementarse usando varios mecanismos diferentes. Ejemplos de mecanismos adecuados incluyen, entre otros, tablas de consulta, implementaciones de hardware, matrices lógicas programables (PLA), memorias de solo lectura (ROM) de microcódigo, etc. En una realización, el núcleo 490 incluye una ROM de microcódigo u otros medio que almacena microcódigo para ciertas macroinstrucciones (por ejemplo, en la unidad de decodificación 440 o de otro modo dentro de la unidad frontal 30). La unidad de decodificación 440 está acoplada a una unidad de cambio de nombre/asignación 452 en la unidad de motor de ejecución 450.

La unidad de motor de ejecución 450 incluye la unidad de cambio de nombre/asignación 452 acoplada a una unidad de retiro 454 y un conjunto de una o más unidades de planificación 456. La unidad de planificación 456 representa cualquier número de planificadores diferentes, incluyendo estaciones de reserva, ventana de instrucción central, etc. La o las unidades de planificación 456 están acopladas a la o las unidades de archivos de registro físico 458. Cada una de las unidades de archivos de registro físico 458 representa uno o más archivos de registro físico, diferentes de los cuales almacenan uno o más tipos de datos diferentes, 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 va a ejecutar), etc. En una realización, la unidad de archivos de registro físico 458 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 vectorial y registros de propósito general. La o las unidades de archivos de registro físico 458 se superpone con la unidad 454 de retiro para mostrar varias formas en las que se puede implementar el cambio de nombre de registro y la ejecución desordenada (por ejemplo, usando una o más memorias intermedias de reorden y uno o más archivos de registro de retiro; usando uno o más archivos futuros, una o más memorias intermedias de historial y uno o más archivos de registro de retiro; usando mapas de registros y un conjunto de registros; etc.). La unidad de retiro 454 y la o las unidades de archivos de registro físico 458 están acopladas al o a los grupos de ejecución 460. El o los grupos de ejecución 460 incluye un conjunto de una o más unidades de ejecución 462 y un conjunto de una o más unidades con acceso a memoria 464. Las unidades de ejecución 462 pueden realizar varias operaciones (por ejemplo, desplazamientos, suma, resta, multiplicación) y en varios tipos de datos (por ejemplo, punto flotante escalar, entero empaquetado, punto flotante empaquetado, entero vectorial, vector punto flotante). Si bien algunas realizaciones pueden incluir varias unidades de ejecución dedicadas a funciones o conjuntos de funciones específicas, otras realizaciones pueden incluir solo una unidad de ejecución o múltiples unidades de ejecución que realizan todas las funciones. La o las unidades de planificación 456, la o las unidades de archivos de registro físico 458 y el o los grupos de ejecución 460 se muestran como posiblemente plurales porque ciertas realizaciones crean canales separados para ciertos tipos de datos/operaciones (por ejemplo, un canal de entero escalar, un canal de punto flotante escalar/entero empaquetado/punto flotante empaquetado/entero vectorial/punto flotante vectorial y/o un canal con acceso a memoria, cada una de las cuales tiene su propia unidad de planificación, unidad de archivos de registro físico y/o grupo de ejecución - y en el caso de un canal con acceso a memoria separado, se implementan ciertas realizaciones en las que solo el grupo de ejecución de este canal tiene la o las unidades con acceso a memoria 464). También debe entenderse que cuando se utilizan canales separados, una o más de estas canales pueden estar desordenados en servicio/ejecución y el resto ordenados.

El conjunto de unidades con acceso a memoria 464 está acoplado a la unidad de memoria 470, que incluye una unidad TLB de datos 472 acoplada a una unidad de caché de datos 474 acoplada a una unidad de caché de nivel 2 (L2) 476. En una realización ilustrativa, las unidades con acceso a memoria 464 puede 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 472 en la unidad de memoria 470. La unidad de caché de instrucciones 434 está además acoplada a una unidad de caché de nivel 2 (L2) 476 en la unidad de memoria 470. La unidad de caché L2 476 está acoplada a uno o más niveles de caché y eventualmente a una memoria principal.

A modo de ejemplo, la arquitectura central ilustrativa de cambio de nombre de registro, emisión/ejecución desordenada puede implementar el canal 400 de la siguiente manera: 1) la recuperación de instrucciones 438 realiza las etapas de recuperación y decodificación de longitud 402 y 404; 2) la unidad de decodificación 440 realiza la etapa de decodificación 406; 3) la unidad de cambio de nombre/asignación 452 realiza la etapa de asignación 408 y la etapa de cambio de nombre 410; 4) la o las unidades de planificación 456 realizan la etapa de planificación 412; 5) la o las unidades de archivos de registro físico 458 y la unidad de memoria 470 realizan la etapa 414 de lectura de registro/lectura de memoria; el grupo de ejecución 460 realiza la etapa de ejecución 416; 6) la unidad de memoria 470 y la o las unidades de archivos de registro físico 458 realizan la etapa 418 de reescritura/escritura en memoria; 7)

varias unidades pueden estar involucradas en la etapa de manejo de excepciones 422; y 8) la unidad de retiro 454 y la o las unidades de archivos de registro físico 458 realizan la etapa de confirmación 424.

El núcleo 490 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 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), incluidas las instrucciones descritas en el presente documento. En una realización, el núcleo 490 incluye lógica para soportar una extensión del conjunto de instrucciones de datos empaquetados (por ejemplo, AVX1, AVX2, extensiones de matriz avanzadas (AMX) y/o alguna forma del formato de instrucción genérico compatible con vectores (U=0 y/o U=1), que se describe a continuación), permitiendo así que las operaciones utilizadas por muchas aplicaciones multimedia se realicen utilizando datos empaquetados.

Debe entenderse que el núcleo puede soportar subprocesos múltiples (ejecutar dos o más conjuntos paralelos de operaciones o subprocesos), y puede hacerlo de diversas maneras, incluyendo subprocesos múltiples en intervalos de tiempo, subprocesos múltiples simultáneos (donde un único núcleo físico proporciona un núcleo lógico para cada uno de los subprocesos que el núcleo físico está subprocesando de forma múltiple simultáneamente), o una combinación de los mismos (por ejemplo, recuperación y decodificación en intervalos de tiempo y subprocesos múltiples simultáneos a partir de entonces, como en la tecnología Intel® Hyperthreading).

Si bien el cambio de nombre de registros se describe en el contexto de la ejecución desordenada, debe entenderse que el cambio de nombre de registros puede usarse en una arquitectura ordenada. Si bien la realización ilustrada del procesador también incluye unidades de caché de datos e instrucciones separadas 34/474 y una unidad de caché L2 compartida 476, 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 caché puede ser externa al núcleo y/o al procesador.

Las **Figuras 5A-B** muestran un diagrama de bloques de una arquitectura de núcleo ordenado ilustrativa más específica, cuyo núcleo sería uno de varios bloques lógicos (que incluyen 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 en anillo) con alguna lógica de función fija, interfaces de E/S de memoria y otra lógica de E/S necesaria, de acuerdo con la aplicación.

La **Figura 5A** es un diagrama de bloques de un único núcleo de procesador, junto con su conexión a la red de interconexión integrada 502 y con su subconjunto local de la memoria caché de nivel 2 (L2) 504, de acuerdo con realizaciones de la invención. En una realización, un decodificador de instrucciones 500 soporta el conjunto de instrucciones x86 con una extensión del conjunto de instrucciones de datos empaquetados. Una caché L1 506 permite accesos de baja latencia a la memoria caché en las unidades escalares y vectoriales. Mientras que en una realización (para simplificar el diseño), una unidad escalar 508 y una unidad vectorial 510 usan conjuntos de registros separados (respectivamente, registros escalares 512 y registros vectoriales 514) y los datos transferidos entre ellos se escriben en la memoria y luego se vuelven a leer desde una memoria caché de nivel 1 (L1) 506, realizaciones alternativas de la invención 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 L2 504 es parte de una caché L2 global que está dividida 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 504. Los datos leídos por un núcleo de procesador se almacenan en su subconjunto de caché L2 504 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 504 y se eliminan de otros subconjuntos, si es necesario. La red en anillo garantiza la coherencia de los datos compartidos. La red en anillo es bidireccional para permitir que agentes tales como núcleos de procesador, caché L2 y otros bloques lógicos se comuniquen entre sí dentro del chip. Cada ruta de datos del anillo tiene 1012 bits de anchura por dirección.

La **Figura 5B** es una vista ampliada de parte del núcleo del procesador en la **Figura 5A** de acuerdo con realizaciones de la invención. La **Figura 5B** incluye una caché de datos L1 506A parte de la caché L2 504, así como más detalles con respecto a la unidad de vector 510 y los registros vectoriales 514. Específicamente, la unidad vectorial 510 es una unidad de procesamiento vectorial (VPU) de 16 de ancho (véase la ALU 528 de 16 de ancho), que ejecuta una o más instrucciones flotantes de precisión de enteros, sencilla y doble. La VPU soporta el mezclado de las entradas de registro con la unidad de mezcla 520, la conversión numérica con las unidades de conversión numérica 522A-B y la replicación con la unidad de replicación 524 en la entrada de memoria. Los registros de máscara de escritura 526 permiten predecir escrituras vectoriales resultantes.

La **Figura 6** es un diagrama de bloques de un procesador 600 que puede tener más de un núcleo, puede tener un controlador de memoria integrado y puede tener gráficos integrados de acuerdo con realizaciones de la invención. Los recuadros con líneas continuas en la **Figura 6** muestran un procesador 600 con un solo núcleo 602A, un agente de sistema 610, un conjunto de una o más unidades de controlador de bus 616, mientras que la adición opcional de los recuadros con líneas discontinuas muestra un procesador alternativo 600 con múltiples núcleos 602A-N (que puede incluir una o más unidades de caché 604A-N), un conjunto de una o más unidades de controlador de memoria integradas 614 en la unidad de agente de sistema 610, y lógica de propósito especial 608 (por ejemplo, lógica de gráficos integrada).

Por lo tanto, diferentes implementaciones del procesador 600 pueden incluir: 1) una CPU con la lógica de propósito especial 608 siendo gráficos integrados y/o lógica científica (de rendimiento) (que puede incluir uno o más núcleos), y siendo los núcleos 602A-N uno o más núcleos de propósito general (por ejemplo, núcleos ordenados de propósito general, núcleos desordenados de propósito general, una combinación de los dos); 2) un coprocesador con núcleos 602A-N que son un gran número de núcleos de propósito especial destinados principalmente a gráficos y/o científicos (rendimiento); y 3) un coprocesador con los núcleos 602A-N siendo un gran número de núcleos ordenados de propósito general. Así, el procesador 600 puede ser un procesador de propósito general, coprocesador o procesador de propósito especial, tal como, por ejemplo, un procesador de red o comunicación, motor de compresión, procesador de gráficos, GPGPU (unidad de procesamiento de gráficos de propósito general), un coprocesador de alto rendimiento de muchos núcleos integrados (MIC) (incluidos 30 o más núcleos), procesador integrado o similar. El procesador puede implementarse en uno o más chips. El procesador 600 puede ser parte y/o puede implementarse en uno o más sustratos usando cualquiera de un número de 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 o una o más unidades de caché compartidas 606 y memoria externa (no mostrada) acoplada al conjunto de unidades de controlador de memoria integrado 614. El conjunto de unidades de caché compartida 606 puede incluir una o más cachés de nivel medio, tales como de nivel 2 (L2), nivel 3 (L3), nivel 4 (L4) u otros niveles de caché, una caché de último nivel (LLC) y/o combinaciones de las mismas. Mientras que en una realización una unidad de interconexión basada en anillo 612 interconecta la lógica de propósito especial 608, el conjunto de unidades de caché compartidas 606 y la unidad de agente del sistema 610/unidades de controlador de memoria integrada 614, realizaciones alternativas pueden usar cualquier número de técnicas bien conocidas para interconectar tales unidades. En una realización, se mantiene la coherencia entre una o más unidades de caché compartidas 606 y núcleos 602A-N.

En algunas realizaciones, uno o más de los núcleos 602A-N son capaces de realizar subprocesos múltiples. El agente de sistema 610 incluye aquellos componentes que coordinan y operan los núcleos 602A-N. La unidad de agente de sistema 610 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 602A-N y la lógica de propósito especial 608. La unidad de visualización es para controlar una o más pantallas conectadas externamente.

Los núcleos 602A-N pueden ser homogéneos o heterogéneos en términos de conjunto de instrucciones de arquitectura; es decir, dos o más de los núcleos 602A-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.

Las **Figuras 7-10** son diagramas de bloques de arquitecturas informáticas ilustrativas. Otros diseños y configuraciones de sistema conocidos en la técnica para portátiles, equipos de sobremesa, PC portátiles, asistentes digitales personales, estaciones de trabajo de ingeniería, servidores, dispositivos de red, concentradores de red, conmutadores, procesadores integrados, procesadores de señales digitales (DSP), dispositivos de gráficos, dispositivos de videojuegos, decodificadores de salón, microcontroladores, teléfonos móviles, reproductores multimedia portátiles, dispositivos de mano y diversos otros dispositivos electrónicos, también son adecuados. En general, son generalmente adecuados una gran diversidad de sistemas o dispositivos electrónicos que pueden incorporar un procesador y/u otra lógica de ejecución como se divulga en el presente documento.

Con referencia ahora a la **Figura 7**, se muestra un diagrama de bloques de un sistema 700 de acuerdo con una realización de la presente invención. El sistema 700 puede incluir uno o más procesadores 710, 715, que están acoplados a un concentrador de controlador 720. En una realización, el concentrador de controlador 720 incluye un concentrador de controlador de memoria de gráficos (GMCH) 790 y un concentrador de entrada/salida (IOH) 750 (que pueden estar en chips separados); el GMCH 790 incluye controladores de memoria y gráficos a los que están acoplados la memoria 740 y un coprocesador 745; el IOH 750 acopla dispositivos de entrada/salida (E/S) 760 al GMCH 790. Alternativamente, uno o ambos controladores de memoria y gráficos están integrados dentro del procesador (como se describe en el presente documento), la memoria 740 y el coprocesador 745 están acoplado directamente al procesador 710, y al concentrador de controlador 720 en un único chip con el IOH 750.

La naturaleza opcional de los procesadores adicionales 715 se indica en la **Figura 7** con líneas discontinuas. Cada procesador 710, 715 puede incluir uno o más de los núcleos de procesamiento descritos en el presente documento y puede ser alguna versión del procesador 600.

5 La memoria 740 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 720 se comunica con el o los procesadores 710, 715 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 795.

10 En una realización, el coprocesador 745 es un procesador de propósito especial, tal como, por ejemplo, un procesador MIC de alto rendimiento, un procesador de red o comunicación, motor de compresión, procesador de gráficos, GPGPU, procesador integrado o similar. En una realización, el concentrador de controlador 720 puede incluir un acelerador de gráficos integrado.

15 Puede haber una variedad de diferencias entre los procesadores 710, 715 en términos de un espectro de métricas de mérito que incluyen características arquitectónicas, microarquitectónicas, térmicas, de consumo de energía y similares.

20 En una realización, el procesador 710 ejecuta instrucciones que controlan operaciones de procesamiento de datos de un tipo general. Incrustadas dentro de las instrucciones puede haber instrucciones de coprocesador. El procesador 710 reconoce estas instrucciones de coprocesador como de un tipo que debería ser ejecutado por el coprocesador adjunto 745. En consecuencia, el procesador 710 emite estas instrucciones de coprocesador (o señales de control que representan instrucciones de coprocesador) en un bus del coprocesador u otra interconexión, al coprocesador 745. Los coprocesadores 745 aceptan y ejecutan las instrucciones de coprocesador recibidas.

25 Con referencia ahora a la **Figura 8**, se muestra un diagrama de bloques de un primer sistema 800 ilustrativo más específico de acuerdo con una realización de la presente invención. Como se muestra en la **Figura 8**, el sistema multiprocesador 800 es un sistema de interconexión punto a punto e incluye un primer procesador 870 y un segundo procesador 880 acoplados a través de una interconexión punto a punto 850. Cada uno de los procesadores 870 y 880 puede ser alguna versión del procesador 600. En una realización de la invención, los procesadores 870 y 880 son respectivamente procesadores 710 y 715, mientras que el coprocesador 838 es el coprocesador 745. En otra realización, los procesadores 870 y 880 son respectivamente el procesador 710 y el coprocesador 745.

30 Los procesadores 870 y 880 se muestran incluyendo las unidades de controlador de memoria integrada (IMC) 872 y 882, respectivamente. El procesador 870 también incluye como parte de sus unidades de controlador de bus, interfaces punto a punto (P-P) 876 y 878; de manera similar, el segundo procesador 880 incluye interfaces P-P 886 y 888. Los procesadores 870, 880 pueden intercambiar información a través de una interfaz punto a punto (P-P) 850 usando circuitos de interfaz P-P 878, 888. Como se muestra en la Figura 8, los IMC 872 y 882 acoplan los procesadores a las respectivas memorias, en concreto, una memoria 832 y una memoria 834, que pueden ser porciones de la memoria principal conectadas localmente a los respectivos procesadores.

35 Cada uno de los procesadores 870, 880 puede intercambiar información con un conjunto de chips 890 a través de interfaces P-P 852, 854 individuales usando circuitos de interfaz punto a punto 876, 894, 886, 898. El conjunto de chips 890 puede opcionalmente intercambiar información con el coprocesador 838 a través de una interfaz de alto rendimiento 839. En una realización, el coprocesador 838 es un procesador de propósito especial, tal como, por ejemplo, un procesador MIC de alto rendimiento, un procesador de red o comunicación, motor de compresión, procesador de gráficos, GPGPU, procesador integrado o similar.

40 Se puede incluir una caché compartida (no mostrada) en cualquiera de los procesadores o fuera de ambos procesadores, pero conectada con los procesadores a través de la interconexión P-P, de modo que la información de la caché local de uno o ambos procesadores puede almacenarse en la caché compartida si se coloca un procesador en un modo de bajo consumo.

45 El conjunto de chips 890 se puede acoplar a un primer bus 816 a través de una interfaz 896. En una realización, el primer bus 816 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 así limitado.

50 Como se muestra en la **Figura 8**, diversos dispositivos de E/S 814 pueden acoplarse al primer bus 816, junto con un puente de bus 818 que acopla el primer bus 816 a un segundo bus 820. En una realización, uno o más procesadores adicionales 815, tales como coprocesadores, procesadores MIC de alto rendimiento, GPGPU, aceleradores (tales como, por ejemplo, aceleradores de gráficos o unidades de procesamiento de señales digitales (DSP)), conjuntos de puertas programables en campo o cualquier otro procesador, están acoplados al primer bus 816. En una realización, el segundo bus 820 puede ser un bus de recuento bajo de pines (LPC). Se pueden acoplar diversos dispositivos a un segundo bus 820 que incluye, por ejemplo, un teclado y/o ratón 822, dispositivos de comunicaciones 827 y una unidad de almacenamiento 828 tal como una unidad de disco u otro dispositivo de almacenamiento masivo que puede incluir

instrucciones/códigos y datos 830, en una realización. Además, se puede acoplar una E/S de audio 824 al segundo bus 820. Obsérvese que son posibles otras arquitecturas. Por ejemplo, en lugar de la arquitectura de punto a punto de la **Figura 8**, un sistema puede implementar un bus multipunto u otra arquitectura de este tipo.

5 Con referencia ahora a la **Figura 9**, se muestra un diagrama de bloques de un segundo sistema 900 ilustrativo más específico de acuerdo con una realización de la presente invención. Elementos similares en las **Figuras 8 y 9** llevan números de referencia similares, y ciertos aspectos de la **Figura 8** se han omitido de la **Figura 9** para evitar complicar otros aspectos de la **Figura 9**.

10 La **Figura 9** muestra que los procesadores 970, 980 pueden incluir memoria integrada y lógica de control de E/S ("CL") 972 y 982, respectivamente. Por lo tanto, la CL 972, 982 incluyen unidades controladoras de memoria integradas e incluyen lógica de control de E/S. La **Figura 9** muestra que no solo las memorias 932, 934 están acopladas a la CL 972, 982, sino que también los dispositivos de E/S 914 también están acoplados a la lógica de control 972, 982. Los dispositivos de E/S heredados 915 están acoplados al conjunto de chips 990.

15 Con referencia ahora a la **Figura 10**, se muestra un diagrama de bloques de un SoC 1000 de acuerdo con una realización de la presente invención. Elementos similares en la **Figura 6** llevan los mismos números de referencia. Además, los recuadros con línea discontinua son características opcionales en los SoC más avanzados. En la **Figura 10**, una unidad o unidades de interconexión 1002 están acopladas a: un procesador de aplicaciones 1010 que incluye un conjunto de uno o más núcleos 1022A-N y una unidad o unidades de caché compartida 1006; una unidad de agente de sistema 1008; una unidad o unidades de controlador de bus 1016; una unidad o unidades de controlador de memoria integrado 1014; un conjunto o uno o más coprocesadores 1020 que pueden incluir lógica de gráficos integrada, un procesador de imágenes, un procesador de audio y un procesador de vídeo; una unidad de memoria de acceso aleatorio estática (SRAM) 1030; una unidad de acceso directo a memoria (DMA) 1032; y una unidad de visualización 1040 para acoplarse a una o más pantallas externas. En una realización, los coprocesadores 1020 incluyen un procesador de propósito especial, tal como, por ejemplo, un procesador de red o comunicación, motor de compresión, GPGPU, un procesador MIC de alto rendimiento, procesador integrado o similar.

20 Las realizaciones de los mecanismos divulgados en el presente documento pueden implementarse en hardware, software, firmware o una combinación de tales enfoques de implementación. Las realizaciones de la invención 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 (que incluye memoria y/o elementos de almacenamiento volátiles y no volátiles), al menos un dispositivo de entrada y al menos un dispositivo de salida.

30 El código de programa, tal como el código 830 mostrado en la **Figura 8**, 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 puede aplicarse a uno o más dispositivos de salida, de forma conocida. Para los propósitos de esta solicitud, un sistema de procesamiento incluye cualquier sistema que tenga un procesador, tal como, por ejemplo; un procesador de señales digitales (DSP), un microcontrolador, un circuito integrado de específico de la aplicación (ASIC) o un microprocesador.

35 El código del programa puede implementarse en un lenguaje de programación orientado a objetos o procedural de alto nivel para comunicarse con un sistema de procesamiento. El código del programa también puede implementarse en lenguaje ensamblador o 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.

40 Uno o más aspectos de al menos una realización pueden implementarse mediante instrucciones representativas almacenadas en un medio legible por máquina que representa diversa lógica dentro del procesador que, cuando las lee una máquina, hace que la máquina fabrique lógica para realizar las técnicas descritas en el presente documento. Tales representaciones, conocidas como "núcleos de IP", pueden almacenarse en un medio legible por máquina tangible y suministrarse a diversos clientes o instalaciones de fabricación para cargarlas en las máquinas de fabricación que realmente hacen la lógica o el procesador.

45 Dichos medios de almacenamiento legibles por máquina pueden incluir, entre otros, 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, compactos, memorias de disco de solo lectura (CD-ROM), discos compactos regrabables (CD-RW), vídeo digital/disco versátil (DVD), Blu-ray (BD) y discos magnetoópticos, dispositivos semiconductores tales como memorias de solo lectura (ROM), memorias de acceso aleatorio (RAM) tales como memorias dinámicas de acceso aleatorio (DRAM), memorias estáticas de acceso aleatorio (SRAM), memorias de solo lectura programables y borrables (EPROM), memorias flash, memorias de solo lectura programables y borrables eléctricamente (EEPROM), memorias de cambio de fase (PCM), tarjetas magnéticas u ópticas, o cualquier otro tipo de soporte adecuado para almacenar instrucciones electrónicas.

50 En consecuencia, las realizaciones de la invención también incluyen medios legibles por máquina, tangibles, no transitorios que contienen instrucciones o 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. Tales realizaciones también pueden denominarse productos de programa.

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 de destino. Por ejemplo, el convertidor de instrucciones puede traducir (por ejemplo, usando traducción binaria estática, traducción binaria dinámica que incluye compilación dinámica), transformar, emular o convertir de otro modo una instrucción en una o más instrucciones para que las procese el núcleo. El convertidor de instrucciones puede implementarse en software, hardware, firmware o una combinación de los mismos. El convertidor de instrucciones puede estar en el procesador, fuera del procesador o en parte dentro y fuera del procesador.

La **Figura 11** 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 en instrucciones binarias en un conjunto de instrucciones de destino de acuerdo con realizaciones de la invención. En la realización ilustrada, el convertidor de instrucciones es un convertidor de instrucciones de software, aunque, como alternativa, el convertidor de instrucciones puede implementarse en software, firmware, hardware o diversas combinaciones de los mismos. La **Figura 11** muestra un programa en un lenguaje de alto nivel 1102 que puede compilarse usando un compilador x86 1104 para generar código binario x86 1106 que puede ejecutarse de forma nativa mediante un procesador con al menos un núcleo de conjunto de instrucciones x86 1116. El procesador con al menos un núcleo de conjunto de instrucciones x86 1116 representa cualquier procesador que puede realizar sustancialmente las mismas funciones que un procesador Intel con al menos un núcleo del conjunto de instrucciones x86 ejecutando o procesando de otro modo (1) de manera compatible una parte sustancial del conjunto de instrucciones del núcleo de conjunto de instrucciones Intel x86 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, para lograr sustancialmente el mismo resultado que un procesador Intel con al menos un núcleo de conjunto de instrucciones x86. El compilador x86 1104 representa un compilador que puede funcionar para generar código binario x86 1106 (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 1116. De manera similar, la **Figura 11** muestra que el programa en el lenguaje de alto nivel 1102 puede compilarse usando un compilador de conjunto de instrucciones alternativo 1108 para generar un código binario de conjunto de instrucciones alternativo 1110 que puede ejecutarse de forma nativa por un procesador sin al menos un núcleo de conjunto de instrucciones x86 1114 (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 1112 se usa para convertir el código binario x86 1106 en un código que el procesador puede ejecutar de forma nativa sin un núcleo de conjunto de instrucciones x86 1114. No es probable que este código convertido sea el mismo que el código binario del conjunto de instrucciones alternativo 1110 porque es difícil hacer un convertidor de instrucciones apto para esto; sin embargo, el código convertido conseguirá la operación general y estará compuesto por instrucciones del conjunto de instrucciones alternativo. Por lo tanto, el convertidor de instrucciones 1112 representa software, firmware, hardware o una combinación de los mismos que, mediante emulación, simulación o cualquier otro proceso, permite que un procesador u otro dispositivo electrónico que no tiene un procesador o núcleo de conjunto de instrucciones x86 ejecute el código binario x86 1106.

Sistemas informáticos adicionales: descripción general

La **Figura 12** es un diagrama de bloques que ilustra un sistema informático 1200 configurado para implementar uno o más aspectos de las realizaciones descritas en el presente documento. El sistema informático 1200 incluye un subsistema de procesamiento 1201 que tiene uno o más procesadores 1202 y una memoria de sistema 1204 que se comunica por medio de una ruta de interconexión que puede incluir un concentrador de memoria 1205. El concentrador de memoria 1205 puede ser un componente separado dentro de un componente de conjunto de chips o puede integrarse dentro de los uno o más procesadores 1202. El concentrador de memoria 1205 se acopla con un subsistema de E/S 1211 mediante un enlace de comunicación 1206. El subsistema de E/S 1211 incluye un concentrador de E/S 1207 que puede habilitar al sistema informático 1200 para recibir entrada desde uno o más dispositivo o dispositivos de entrada 1208. Adicionalmente, el concentrador de E/S 1207 puede permitir un controlador de visualización, que puede estar incluido en el uno o más procesadores 1202, para que proporcione salidas a uno o más dispositivos de visualización 1210A. En una realización, el uno o más dispositivos de visualización 1210A acoplados con el concentrador de E/S 1207 pueden incluir un dispositivo de visualización local, interno o integrado.

En una realización, el subsistema de procesamiento 1201 incluye uno o más procesador o procesadores paralelos 1212 acoplados al concentrador de memoria 1205 mediante un bus u otro enlace de comunicación 1213. El enlace de comunicaciones 1213 puede ser uno de cualquier número de tecnologías o protocolos de enlace de comunicaciones basados en normas, tales como, pero sin limitación, PCI Express, o puede ser una interfaz de comunicaciones o estructura de comunicaciones específica de distribuidor. En una realización, uno o más procesadores paralelos 1212 forman un sistema de procesamiento vectorial o paralelo enfocado de cálculo que incluye una gran cantidad de núcleos de procesamiento y/o grupos de procesamiento, tales como un procesador de muchos núcleos integrados (MIC). En una realización, el uno o más procesador o procesadores paralelos 1212 forman un subsistema de procesamiento de gráficos que puede emitir píxeles a uno del uno o más dispositivo o dispositivos de visualización 1210A acoplados mediante el concentrador de E/S 1207. El uno o más procesador o procesadores paralelos 1212 también pueden

incluir un controlador de visualización e interfaz de visualización (no mostrados) para permitir una conexión directa a uno o más dispositivo o dispositivos de visualización 1210B.

Dentro del subsistema de E/S 1211, una unidad de almacenamiento de sistema 1214 puede conectarse al concentrador de E/S 1207 para proporcionar un mecanismo de almacenamiento para el sistema informático 1200. Se puede usar un conmutador de E/S 1216 para proporcionar un mecanismo de interfaz para permitir conexiones entre el concentrador de E/S 1207 y otros componentes, tales como un adaptador de red 1218 y/o un adaptador de red inalámbrica 1219 que pueden estar integrados en la plataforma, y otros diversos dispositivos que puedan añadirse por medio de uno o más dispositivo o dispositivos de adición 1220. El adaptador de red 1218 puede ser un adaptador Ethernet u otro adaptador de red cableado. El adaptador de red inalámbrico 1219 puede incluir uno o más de un dispositivo de red de Wi-Fi, de Bluetooth, de comunicación de campo cercano (NFC) o de otro tipo que incluya una o más radios inalámbricas.

El sistema informático 1200 puede incluir otros componentes no mostrados de forma explícita, que incluyen conexiones USB u otros puertos, unidades de almacenamiento óptico, dispositivos de captura de vídeo y similares, que también se pueden conectar al concentrador de E/S 1207. Las rutas de comunicación que interconectan los diversos componentes en la **Figura 12** se pueden implementar usando cualquier protocolo adecuado, tal como protocolos (por ejemplo, PCI-Express) basados en PCI (Interconexión de Componentes Periféricos), o cualesquiera otras interfaces de comunicación de bus o de punto a punto y/o protocolo o protocolos, tal como la interconexión de alta velocidad NV-Link o protocolos de interconexión conocidos en la técnica.

En una realización, el uno o más procesador o procesadores paralelos 1212 incorporan circuitos optimizados para procesamiento de gráficos y vídeo, incluyendo, por ejemplo, circuitería de salida de vídeo, y constituyen una unidad de procesamiento de gráficos (GPU). En otra realización, uno o más procesadores paralelos 1212 incorporan circuitos optimizados para procesamiento de propósito general, preservando al mismo tiempo la arquitectura de cálculo subyacente, descrita con mayor detalle en el presente documento. En otra realización más, los componentes del sistema informático 1200 pueden integrarse con uno o más elementos del sistema en un único circuito integrado. Por ejemplo, el uno o más procesador o procesadores paralelos 1212, el concentrador de memoria 1205, el procesador o procesadores 1202 y el concentrador de E/S 1207 pueden integrarse en un circuito integrado de sistema en chip (SoC). Como alternativa, los componentes del sistema informático 1200 pueden integrarse en un único paquete para formar una configuración de sistema en paquete (SIP). En una realización, al menos una parte de los componentes del sistema informático 1200 se puede integrar en un módulo de múltiples chips (MCM), que se puede interconectar con otros módulos de múltiples chips en un sistema informático modular.

Se apreciará que el sistema informático 1200 mostrado en el presente documento es ilustrativo y que se permiten variaciones y modificaciones. La topología de conexión, incluidos el número y la disposición de puentes, el número de procesadores 1202 y el número de procesadores paralelos 1212, puede modificarse según se desee. Por ejemplo, en algunas realizaciones, la memoria de sistema 1204 está conectada al procesador o procesadores 1202 directamente en lugar de a través de un puente, mientras que otros dispositivos se comunican con la memoria de sistema 1204 mediante el concentrador de memoria 1205 y el procesador o procesadores 1202. En otras topologías alternativas, el procesador o procesadores paralelos 1212 están conectados al concentrador de E/S 1207 o directamente a uno del uno o más procesador o procesadores 1202, en lugar de al concentrador de memoria 1205. En otras realizaciones, el concentrador de E/S 1207 y el concentrador de memoria 1205 se pueden integrar en un único chip. Algunas realizaciones pueden incluir dos o más conjuntos de procesador o procesadores 1202 adjuntos mediante múltiples zócalos, que pueden acoplarse con dos o más instancias del procesador o procesadores paralelos 1212.

Algunos de los componentes particulares mostrados en la presente memoria son opcionales y pueden no estar incluidos en todas las implementaciones del sistema informático 1200. Por ejemplo, puede ser compatible cualquier número de tarjetas o periféricos adicionales o se pueden eliminar algunos componentes. Además, algunas arquitecturas pueden usar terminología diferente para componentes similares a los ilustrados en la **Figura 12**. Por ejemplo, el concentrador de memoria 1205 puede denominarse puente norte en algunas arquitecturas, mientras que el concentrador de E/S 1207 puede denominarse puente sur.

La **Figura 13A** ilustra un procesador paralelo 1300, de acuerdo con una realización. Los diversos componentes del procesador paralelo 1300 pueden implementarse usando uno o más dispositivos de circuito integrado, tales como procesadores programables, circuitos integrados específicos de la aplicación (ASIC) o matrices de puertas programables *in situ* (FPGA). El procesador paralelo 1300 ilustrado es una variante del uno o más procesadores paralelos 1212 mostrados en la **Figura 12**, de acuerdo con una realización.

En una realización, el procesador paralelo 1300 incluye una unidad de procesamiento paralelo 1302. La unidad de procesamiento paralelo incluye una unidad de E/S 1304 que permite una comunicación con otros dispositivos, que incluyen otras instancias de la unidad de procesamiento paralelo 1302. La unidad de E/S 1304 se puede conectar directamente a otros dispositivos. En una realización, la unidad de E/S 1304 se conecta con otros dispositivos mediante el uso de una interfaz de concentrador o de conmutador, tal como un concentrador de memoria 13405. Las conexiones entre el concentrador de memoria 13405 y la unidad de E/S 1304 forman un enlace de comunicación 13413. Dentro la unidad de procesamiento paralelo 1302, la unidad de E/S 1304 se conecta con una interfaz de anfitrión 1306 y una

barra transversal de memoria 1316, donde la interfaz de anfitrión 1306 recibe comandos dirigidos a realizar operaciones de procesamiento y la barra transversal de memoria 1316 recibe comandos dirigidos a realizar operaciones de memoria.

5 Cuando la interfaz de anfitrión 1306 recibe una memoria intermedia de comandos mediante la unidad de E/S 1304, la interfaz de anfitrión 1306 puede dirigir operaciones de trabajo para realizar aquellos comandos a un extremo frontal 1308. En una realización, el extremo frontal 1308 se acopla con un planificador 1310, que está configurado para distribuir comandos u otros elementos de trabajo a una matriz de agrupaciones de procesamiento 1312. En una
10 realización, el planificador 1310 garantiza que la matriz de agrupaciones de procesamiento 1312 está configurada apropiadamente y en un estado válido antes de que se distribuyan las tareas a las agrupaciones de procesamiento de la matriz de agrupaciones de procesamiento 1312. En una realización, el planificador 1310 se implementa mediante lógica de firmware que se ejecuta en un microcontrolador. El planificador implementado por microcontrolador 1310 puede configurarse para realizar operaciones de planificación y de distribución de trabajo complejas con granularidad gruesa y fina, lo que permite la anticipación rápida de prioridad y conmutación de contexto de hilos que se ejecutan
15 en la matriz de procesamiento 1312. En una realización, el software de anfitrión puede demostrar cargas de trabajo para la planificación en la matriz de procesamiento 1312 mediante uno de múltiples timbres de procesamiento de tráfico. Las cargas de trabajo, a continuación, pueden distribuirse automáticamente a través de la matriz de procesamiento 1312 por la lógica del planificador 1310 dentro del microcontrolador planificador.

20 La matriz de grupos de procesamiento 1312 puede incluir hasta "N" grupos de procesamiento (por ejemplo, el grupo 1314A, el grupo 1314B, hasta el grupo 1314N). Cada grupo 1314A-1314N de la matriz de grupo de procesamiento 1312 puede ejecutar una gran cantidad de subprocesos simultáneos. El planificador 1310 puede asignar trabajo a los grupos 1314A-1314N de la matriz de grupos de procesamiento 1312 usando diversos algoritmos de planificación y/o
25 distribución de trabajo, que pueden variar dependiendo de la carga de trabajo que surge para cada tipo de programa o cálculo. La planificación puede ser manejada dinámicamente por el planificador 1310 o puede ser asistida en parte por la lógica de compilador durante la compilación de la lógica de programa configurada para su ejecución por la matriz de grupos de procesamiento 1312. En una realización, diferentes grupos 1314A-1314N de la matriz de grupos de procesamiento 1312 pueden asignarse para procesar diferentes tipos de programas o para realizar diferentes tipos de cálculos.

30 La matriz de grupo de procesamiento 1312 se puede configurar para realizar varios tipos de operaciones de procesamiento en paralelo. En una realización, la matriz de grupo de procesamiento 1312 está configurada para realizar operaciones de cálculo paralelas de propósito general. Por ejemplo, la matriz de agrupaciones de procesamiento 1312 puede incluir lógica para ejecutar tareas de procesamiento que incluye filtración de datos de vídeo
35 y/o de audio, ejecución de operaciones de modelado, que incluye operaciones físicas y ejecución de transformaciones de datos.

En una realización, la matriz de agrupación de procesamiento 1312 está configurada para realizar operaciones de procesamiento de gráficos en paralelo. En realizaciones en las que el procesador paralelo 1300 está configurado para
40 realizar operaciones de procesamiento de gráficos, la matriz de agrupación de procesamiento 1312 puede incluir una lógica adicional para soportar la ejecución de tales operaciones de procesamiento de gráficos, incluyendo, pero sin limitación, una lógica de muestreo de textura para realizar operaciones de textura, así como una lógica de teselación y otra lógica de procesamiento de vértices. Adicionalmente, la matriz de agrupaciones de procesamiento 1312 puede configurarse para ejecutar programas sombreadores relacionados con el procesamiento de gráficos tales como, pero
45 sin limitación, sombreadores de vértices, sombreadores de teselación, sombreadores de geometría y sombreadores de píxeles. La unidad de procesamiento paralelo 1302 puede transferir datos desde la memoria de sistema por medio de la unidad de E/S 1304 para su procesamiento. Durante el procesamiento, los datos transferidos pueden almacenarse en una memoria en chip (por ejemplo, la memoria de procesador paralelo 1322) durante el procesamiento y, a continuación, escribirse en diferido en la memoria del sistema.

50 En una realización, cuando la unidad de procesamiento paralelo 1302 se usa para realizar procesamiento de gráficos, el planificador 1310 puede configurarse para dividir la carga de trabajo de procesamiento en tareas de tamaño aproximadamente igual, para permitir mejor la distribución de las operaciones de procesamiento de gráficos a múltiples grupos 1314A-1314N de la matriz de grupo de procesamiento 1312. En algunas realizaciones, partes de la matriz de
55 grupo de procesamiento 1312 se pueden configurar para realizar diferentes tipos de procesamiento. Por ejemplo, una primera parte puede estar configurada para realizar un sombreado de vértices y una generación de topología, una segunda parte puede estar configurada para realizar teselación y sombreado de geometría, y una tercera parte puede estar configurada para realizar sombreado de píxeles u otras operaciones de espacio de pantalla, para producir una imagen representada gráficamente para su visualización. Los datos intermedios producidos por una o más de las
60 agrupaciones 1314A-1314N se pueden almacenar en memorias intermedias para permitir que se transmitan los datos intermedios entre las agrupaciones 1314A-1314N para su procesamiento adicional.

Durante la operación, la matriz de agrupaciones de procesamiento 1312 puede recibir tareas de procesamiento que se van a ejecutar a través del planificador 1310, que recibe comandos que definen tareas de procesamiento desde el
65 extremo frontal 1308. Para operaciones de procesamiento de gráficos, las tareas de procesamiento pueden incluir índices de datos que hay que procesar, por ejemplo, datos de superficie (parche), datos de primitiva, datos de vértice

y/o datos de píxel, así como parámetros de estado y comandos que definen cómo han de procesarse los datos (por ejemplo, qué programa ha de ejecutarse). El planificador 1310 puede configurarse para extraer los índices que corresponden a las tareas o puede recibir los índices desde el extremo frontal 1308. El extremo frontal 1308 puede estar configurado para garantizar que la matriz de agrupaciones de procesamiento 1312 esté configurada en un estado válido antes de que se inicie la carga de trabajo especificada en memorias intermedias de comandos entrantes (por ejemplo, memorias intermedias de lotes, memorias intermedias de carga, etc.).

Cada una de las una o más instancias de la unidad de procesamiento paralelo 1302 puede acoplarse con memoria de procesador paralelo 1322. Puede accederse a la memoria de procesador paralelo 1322 mediante la barra transversal de memoria 1316, que puede recibir solicitudes de memoria desde la matriz de agrupaciones de procesamiento 1312, así como la unidad de E/S 1304. La barra transversal de memoria 1316 puede acceder a la memoria de procesador paralelo 1322 mediante una interfaz de memoria 1318. La interfaz de memoria 1318 puede incluir múltiples unidades de partición (por ejemplo, unidad de partición 1320A, unidad de partición 1320B, hasta unidad de partición 1320N) que pueden cada una acoplarse a una parte (por ejemplo, unidad de memoria) de la memoria de procesador paralelo 1322. En una implementación, el número de unidades de partición 1320A-1320N está configurado para que sea igual al número de unidades de memoria, de manera que una primera unidad de partición 1320A tiene una correspondiente primera unidad de memoria 1324A, una segunda unidad de partición 1320B tiene una correspondiente unidad de memoria 1324B y una unidad de partición de orden N 1320N tiene una correspondiente unidad de memoria de orden N 1324N. En otras realizaciones, el número de unidades de subdivisión 1320A-1320N puede no ser igual al número de dispositivos de memoria.

En diversas realizaciones, las unidades de memoria 1324A-1324N pueden incluir diversos tipos de dispositivos de memoria, incluyendo memoria de acceso aleatorio dinámica (DRAM) o memoria de acceso aleatorio de gráficos, tal como una memoria de acceso aleatorio de gráficos sincrónica (SGRAM), incluyendo una memoria de tasa de datos doble de gráficos (GDDR). En una realización, las unidades de memoria 1324A-1324N pueden incluir también una memoria aplada 3D, incluyendo, pero sin limitación, una memoria de alto ancho de banda (HBM). Los expertos en la técnica apreciarán que la implementación específica de las unidades de memoria 1324A-1324N puede variar y puede seleccionarse entre uno de varios diseños convencionales. Los objetivos de renderizado, tales como memoria intermedia de cuadros o mapas de textura, pueden almacenarse a través de las unidades de memoria 1324A-1324N, permitiendo que las unidades de partición 1320A-1320N escriban partes de cada objetivo de renderizado en paralelo para usar eficientemente el ancho de banda disponible de la memoria de procesador paralelo 1322. En algunas realizaciones, se puede excluir una instancia local de la memoria de procesador paralelo 1322 en favor de un diseño de memoria unificado que utiliza memoria de sistema junto con memoria caché local.

En una realización, una cualquiera de las agrupaciones 1314A-1314N de la matriz de agrupaciones de procesamiento 1312 puede procesar datos que se escribirán en cualquiera de las unidades de memoria 1324A-1324N dentro de la memoria de procesador paralelo 1322. La barra transversal de memoria 1316 puede estar configurada para transferir la salida de cada grupo 1314A-1314N en cualquier unidad de partición 1320A-1320N o en otro grupo 1314A-1314N, que puede realizar operaciones de procesamiento adicionales en la salida. Cada grupo 1314A-1314N puede comunicarse con la interfaz de memoria 1318 a través de la barra transversal de memoria 1316 para leer o escribir en diversos dispositivos de memoria externos. En una realización, la barra transversal de memoria 1316 tiene una conexión a la interfaz de memoria 1318 para comunicarse con la unidad de E/S 1304, así como una conexión a una instancia local de la memoria de procesador paralelo 1322, lo que posibilita que las unidades de procesamiento dentro de las diferentes agrupaciones de procesamiento 1314A-1314N se comuniquen con la memoria de sistema u otra memoria que no sea local a la unidad de procesamiento paralelo 1302. En una realización, la barra transversal de memoria 1316 puede usar canales virtuales para separar flujos de tráfico entre las agrupaciones 1314A-1314N y las unidades de subdivisión 1320A-1320N.

Aunque se ilustra una única instancia de la unidad de procesamiento paralelo 1302 dentro del procesador paralelo 1300, puede incluirse cualquier número de instancias de la unidad de procesamiento paralelo 1302. Por ejemplo, se pueden proporcionar múltiples instancias de la unidad de procesamiento paralelo 1302 en una única tarjeta de adición, o se pueden interconectar múltiples tarjetas de adición. Las diferentes instancias de la unidad de procesamiento paralelo 1302 pueden configurarse para interfuncionar incluso si las diferentes instancias tienen diferentes números de núcleos de procesamiento, diferentes cantidades de memoria de procesador paralelo local y/u otras diferencias de configuración. Por ejemplo, y en una realización, algunas instancias de la unidad de procesamiento paralelo 1302 pueden incluir unidades de coma flotante de precisión más alta con relación a otras instancias. Los sistemas que incorporan una o más instancias de la unidad de procesamiento paralelo 1302 o el procesador paralelo 1300 se pueden implementar en una diversidad de configuraciones y factores de forma, incluyendo, pero sin limitación, ordenadores personales de sobremesa, portátiles o de mano, servidores, estaciones de trabajo, consolas de juegos y/o sistemas embebidos.

La **Figura 13B** es un diagrama de bloques de una unidad de partición 1320, de acuerdo con una realización. En una realización, la unidad de subdivisión 1320 es una instancia de una de las unidades de subdivisión 1320A-1320N de la **Figura 13A**. Como se ilustra, la unidad de partición 1320 incluye una memoria caché L2 1321, una interfaz de memoria intermedia de trama 1325 y una ROP 1326 (unidad de operaciones de trama). La caché L2 1321 es una caché de lectura/escritura que está configurada para realizar operaciones de carga y almacenamiento recibidas desde la barra

transversal de memoria 1316 y la ROP 1326. Los fallos de lectura y las solicitudes de escritura urgentes se emiten por la caché L2 1321 a la interfaz de memoria intermedia de fotogramas 1325 para su procesamiento. Pueden enviarse también las actualizaciones a la memoria intermedia de fotograma mediante la interfaz de memoria intermedia de fotograma 1325 para su procesamiento. En una realización, la interfaz de memoria intermedia de fotogramas 1325 interactúa con una de las unidades de memoria en la memoria de procesador paralelo, tal como las unidades de memoria 1324A-1324N de la **Figura 13** (por ejemplo, dentro de la memoria de procesador paralelo 1322).

En aplicaciones de gráficos, la ROP 1326 es una unidad de procesamiento que realiza operaciones de trama tal como plantilla, prueba z, combinación y similares. La ROP 1326 emite entonces datos de gráficos procesados que se almacenan en una memoria de gráficos. En algunas realizaciones, la ROP 1326 incluye lógica de compresión para comprimir datos de profundidad o color que se escriben en la memoria y descomprimir datos de profundidad o color que se leen de la memoria. La lógica de compresión puede ser lógica de compresión sin pérdidas, que hace uso de uno o más de múltiples algoritmos de compresión. El tipo de compresión que se realiza por la ROP 1326 puede variar basándose en las características estadísticas de los datos que van a comprimirse. Por ejemplo, en una realización, se realiza una compresión de color delta sobre datos de profundidad y de color por mosaico.

En algunas realizaciones, la ROP 1326 se incluye dentro de cada agrupación de procesamiento (por ejemplo, la agrupación 1314A-1314N de la **Figura 13**) en lugar de dentro de la unidad de subdivisión 1320. En dicha realización, las solicitudes de lectura y escritura de datos de píxeles se transmiten a través de la barra transversal de memoria 1316 en lugar de datos de fragmentos de píxeles. Los datos de gráficos procesados pueden visualizarse en un dispositivo de visualización, tal como uno de uno o más dispositivo o dispositivos de visualización 1210 de la **Figura 12**, enrutarse para su procesamiento adicional por el procesador o procesadores 1202, o enrutarse para su procesamiento adicional por una de las entidades de procesamiento dentro del procesador paralelo 1300 de la **Figura 13A**.

La **Figura 13C** es un diagrama de bloques de un grupo de procesamiento 1314 dentro de una unidad de procesamiento en paralelo, de acuerdo con una realización. En una realización, la agrupación de procesamiento es una instancia de una de las agrupaciones de procesamiento 1314A-1314N de la **Figura 13**. El grupo de procesamiento 1314 puede configurarse para ejecutar muchos subprocesos en paralelo, donde el término "subproceso" se refiere a una instancia de un programa particular que se ejecuta en un conjunto particular de datos de entrada. En algunas realizaciones, se usan técnicas de emisión de instrucción de única instrucción de múltiples datos (SIMD) para soportar la ejecución paralela de un gran número de hilos sin proporcionar múltiples unidades de instrucción independientes. En otras realizaciones, se usan técnicas de única instrucción de múltiples hilos (SIMT) para soportar la ejecución paralela de un gran número de hilos generalmente sincronizados, usando una unidad de instrucciones común configurada para emitir instrucciones en un conjunto de motores de procesamiento dentro de cada una de las agrupaciones de procesamiento. A diferencia del régimen de ejecución de SIMD, donde todos los motores de procesamiento ejecutan habitualmente instrucciones idénticas, la ejecución de SIMT permite que diferentes hilos sigan más fácilmente rutas de ejecución divergentes a través de un programa de hilos dado. Los expertos en la materia entenderán que un régimen de procesamiento de SIMD representa un subconjunto funcional de un régimen de procesamiento de SIMT.

La operación de la agrupación de procesamiento 1314 puede controlarse mediante un gestor de canalizaciones 1332 que distribuye las tareas de procesamiento a procesadores paralelos de SIMT. El administrador de canalización 1332 recibe instrucciones del planificador 1310 de la **Figura 13** y gestiona la ejecución de esas instrucciones a través de un multiprocesador de gráficos 1334 y/o una unidad de textura 1336. El multiprocesador de gráficos 1334 ilustrado es una instancia ilustrativa de un procesador paralelo de SIMT. Sin embargo, se pueden incluir diversos tipos de procesadores paralelos de SIMT de arquitecturas diferentes dentro de la agrupación de procesamiento 1314. Puede incluirse una o más instancias del multiprocesador de gráficos 1334 dentro de una agrupación de procesamiento 1314. El multiprocesador de gráficos 1334 puede procesar datos y puede usarse una barra transversal de datos 1340 para distribuir los datos procesados a uno de múltiples posibles destinos, que incluyen otras unidades sombreadoras. El gestor de canalizaciones 1332 puede facilitar la distribución de datos procesados especificando destinos para que se distribuyan datos procesados mediante la barra transversal de datos 1340.

Cada multiprocesador de gráficos 1334 dentro de la agrupación de procesamiento 1314 puede incluir un conjunto idéntico de lógica de ejecución funcional (por ejemplo, unidades aritmético-lógicas, unidades de carga-almacén, etc.). La lógica de ejecución funcional puede configurarse de una manera canalizada en la que pueden emitirse nuevas instrucciones antes de que se hayan completado instrucciones previas. La lógica de ejecución funcional soporta una diversidad de operaciones que incluyen aritmética de números enteros y de coma flotante, operaciones de comparación, operaciones booleanas, desplazamiento de bits y de cálculo de diversas funciones algebraicas. En una realización, se puede aprovechar el mismo hardware de unidad funcional para realizar diferentes operaciones y puede estar presente cualquier combinación de unidades funcionales.

Las instrucciones transmitidas a la agrupación de procesamiento 1314 constituyen un hilo. Un conjunto de subprocesos ejecutándose a través del conjunto de motores de procesamiento en paralelo es un grupo de subprocesos. Un grupo de hilos ejecuta el mismo programa en diferentes datos de entrada. Cada hilo dentro de un grupo de hilos se puede asignar a un motor de procesamiento diferente dentro de un multiprocesador de gráficos 1334. Un grupo de hilos puede incluir menos hilos que el número de motores de procesamiento dentro del multiprocesador

de gráficos 1334. Cuando un grupo de hilos incluye menos hilos que el número de motores de procesamiento, uno o más de los motores de procesamiento pueden encontrarse en espera durante ciclos en los que se está procesando ese grupo de hilos. Un grupo de hilos puede incluir también más hilos que el número de motores de procesamiento dentro del multiprocesador de gráficos 1334. Cuando el grupo de hilos incluye más hilos que el número de motores de procesamiento dentro del multiprocesador de gráficos 1334, puede realizarse un procesamiento a lo largo de ciclos de reloj consecutivos. En una realización, pueden ejecutarse múltiples grupos de hilos concurrentemente en un multiprocesador de gráficos 1334.

En un ejemplo, el multiprocesador de gráficos 1334 incluye una memoria caché interna para realizar operaciones de carga y de almacenamiento. En una realización, el multiprocesador de gráficos 1334 puede prescindir de una caché interna y usar una memoria caché (por ejemplo, la caché L1 308) dentro de la agrupación de procesamiento 1314. Cada multiprocesador de gráficos 1334 también tiene acceso a memorias caché de L2 dentro de las unidades de partición (por ejemplo, unidades de partición 1320A-1320N de la Figura 13) que se comparten entre todas las agrupaciones de procesamiento 1314 y se pueden usar para transferir datos entre hilos. El multiprocesador de gráficos 1334 también puede acceder a memoria global fuera de chip, que puede incluir una o más memorias de procesador paralelo local y/o memoria del sistema. Puede usarse cualquier memoria externa a la unidad de procesamiento paralelo 1302 como memoria global. Las realizaciones en las que la agrupación de procesamiento 1314 incluye múltiples instancias del multiprocesador de gráficos 1334 pueden compartir instrucciones y datos comunes, que pueden almacenarse en la caché L1 308.

Cada agrupación de procesamiento 1314 puede incluir una MMU 1345 (unidad de gestión de memoria) que está configurada para mapear direcciones virtuales en direcciones físicas. En otras realizaciones, una o más instancias de la MMU 1345 pueden residir dentro de la interfaz de memoria 1318 de la **Figura 13**. La MMU 1345 incluye un conjunto de entradas de tabla de página (PTE) usadas para mapear una dirección virtual a una dirección física de un mosaico (más información sobre la generación de mosaicos) y, opcionalmente, un índice de línea de caché. La MMU 1345 puede incluir memorias intermedias de conversión adelantada (TLB) de dirección o memorias caché que pueden residir dentro del multiprocesador de gráficos 1334 o la memoria caché L1 o la agrupación de procesamiento 1314. La dirección física se procesa para distribuir la localidad de acceso a los datos de superficie para permitir un entrelazado eficiente de solicitudes entre unidades de subdivisión. El índice de línea de memoria caché se puede usar para determinar si una solicitud para una línea de memoria caché es un acierto o un fallo.

En aplicaciones de gráficos e informática, se puede configurar una agrupación de procesamiento 1314 de tal manera que cada multiprocesador de gráficos 1334 esté acoplado a una unidad de textura 1336 para realizar operaciones de mapeo de textura, p. ej., determinar posiciones de muestras de textura, leer datos de textura y filtrar los datos de textura. Los datos de textura se leen desde una caché L1 de textura interna (no mostrada) o, en algunas realizaciones, desde la caché L1 dentro del multiprocesador de gráficos 1334 y se recuperan desde una caché L2, memoria de procesador paralelo local o memoria de sistema, según sea necesario. Cada multiprocesador de gráficos 1334 emite tareas procesadas a la barra transversal de datos 1340 para proporcionar la tarea procesada a otra agrupación de procesamiento 1314 para su procesamiento adicional o para almacenar la tarea procesada en una caché L2, memoria de procesador paralelo local o memoria de sistema mediante la barra transversal de memoria 1316. Una preROP 1342 (unidad de operaciones previas a la rasterización) está configurada para recibir datos desde el multiprocesador de gráficos 1334, dirigir datos a las unidades de ROP, que pueden estar ubicadas con unidades de subdivisión como se describe en el presente documento (por ejemplo, las unidades de subdivisión 1320A-1320N de la **Figura 13**). La unidad preROP 1342 puede realizar optimizaciones para la combinación de colores, organizar datos de color de píxeles y realizar traducciones de direcciones.

Se apreciará que la arquitectura de núcleo descrita en el presente documento es ilustrativa y que son posibles variaciones y modificaciones. Cualquier número de unidades de procesamiento, por ejemplo, multiprocesador de gráficos 1334, unidades de textura 1336, preROP 1342, etc., puede incluirse dentro de una agrupación de procesamiento 1314. Además, aunque únicamente se muestra una agrupación de procesamiento 1314, la unidad de procesamiento paralelo, como se describe en el presente documento, puede incluir cualquier número de instancias de la agrupación de procesamiento 1314. En una realización, cada grupo de procesamiento 1314 puede configurarse para operar independientemente de otros grupos de procesamiento 1314 usando unidades de procesamiento separadas y distintas, caché L1, etc.

La **Figura 13D** muestra un multiprocesador de gráficos 1334, de acuerdo con una realización. En tal realización, el multiprocesador de gráficos 1334 se acopla con el gestor de canal 1332 del grupo de procesamiento 1314. El multiprocesador de gráficos 1334 tiene un canal de ejecución que incluye, entre otros, una caché de instrucciones 1352, una unidad de instrucciones 1354, una unidad de asignación de direcciones 1356, un archivo de registro 1358, uno o más núcleos de unidad de procesamiento de gráficos de propósito general (GPGPU) 1362, y una o más unidades de carga/almacenamiento 1366. Los núcleos GPGPU 1362 y las unidades de carga/almacenamiento 1366 están acoplados con la memoria caché 1372 y la memoria compartida 1370 a través de un Interconexión de memoria y caché 1368.

En una realización, la caché de instrucciones 1352 recibe un flujo de instrucciones para ejecutar desde el gestor de canal 1332. Las instrucciones son almacenadas en la caché de instrucciones 1352 y despachadas para su ejecución

por la unidad de instrucciones 1354. La unidad de instrucciones 1354 puede despachar instrucciones como grupos de subprocesos (por ejemplo, envoltorios), con cada subproceso del grupo de subprocesos asignado a una unidad de ejecución diferente dentro del núcleo de la GPGPU 1362. Una instrucción puede acceder a cualquiera de un espacio de direcciones local, compartido o global, especificando una dirección dentro de un espacio de direcciones unificado. La unidad de correlación de direcciones 1356 se puede usar para convertir direcciones del espacio de direcciones unificado en una dirección de memoria distinta a la que se puede acceder mediante las unidades de carga/almacenamiento 1366.

El archivo de registro 1358 proporciona un conjunto de registros para las unidades funcionales del multiprocesador de gráficos 1334. El archivo de registro 1358 proporciona almacenamiento temporal para los operandos conectados a las rutas de datos de las unidades funcionales (por ejemplo, núcleos de GPGPU 1362, unidades de carga/almacén 1366) del multiprocesador de gráficos 1334. En una realización, el archivo de registro 1358 se divide entre cada una de las unidades funcionales de manera que a cada unidad funcional se le asigna una porción dedicada del archivo de registro 1358. En una realización, el archivo de registro 1358 se divide entre las diferentes urdimbres que se ejecutan mediante el multiprocesador de gráficos 1334.

Cada núcleo de GPGPU 1362 puede incluir unidades de coma flotante (FPU) y/o unidades aritmético-lógicas (ALU) de números enteros que se usan para ejecutar instrucciones del multiprocesador de gráficos 1334. Los núcleos de GPGPU 1362 pueden ser similares en arquitectura o pueden diferir en arquitectura, según las realizaciones. Por ejemplo, y en una realización, una primera porción de los núcleos de GPGPU 1362 incluye una FPU de precisión sencilla y una ALU de números enteros, mientras que una segunda porción de los núcleos de GPGPU incluye una FPU de precisión doble. En una realización, las FPU pueden implementar la norma IEEE 754-2008 para aritmética de coma flotante o posibilitar aritmética de coma flotante de precisión variable. El multiprocesador de gráficos 1334 puede incluir adicionalmente una o más unidades de función fija o de función especial para realizar funciones específicas, tales como operaciones de copia de rectángulo o de mezcla de píxeles. En una realización, uno o más de los núcleos de GPGPU puede incluir también lógica de función fija o especial.

En una realización, los núcleos GPGPU 1362 incluyen lógica SIMD capaz de realizar una única instrucción en múltiples conjuntos de datos. En una realización, los núcleos GPGPU 1362 pueden ejecutar físicamente instrucciones SIMD4, SIMD8 y SINM16 y ejecutar lógicamente instrucciones SINM1, SIMD2 y SIMD32. Las instrucciones SIMD para los núcleos GPGPU pueden ser generadas en tiempo de compilación por un compilador sombreador o pueden generarse automáticamente cuando se ejecutan programas escritos y compilados para arquitecturas de único programa - múltiples datos (SPMD) o de SIMT. Múltiples hilos de un programa configurado para el modelo de ejecución de SIMT pueden ejecutarse mediante una única instrucción de SIMD. Por ejemplo, y en una realización, se pueden ejecutar en paralelo ocho hilos SIMT que realizan operaciones iguales o similares a través de una única unidad lógica SIMD8.

La interconexión de memoria y de caché 1368 es una red de interconexión que conecta cada una de las unidades funcionales del multiprocesador de gráficos 1334 al archivo de registros 1358 y a la memoria compartida 1370. En una realización, la interconexión de memoria y de caché 1368 es una interconexión de barra transversal que permite que la unidad de carga/almacenamiento 1366 implemente operaciones de carga y de almacenamiento entre la memoria compartida 1370 y el archivo de registros 1358. El archivo de registro 1358 puede funcionar a la misma frecuencia que los núcleos de GPGPU 1362, por lo tanto, la transferencia de datos entre los núcleos de GPGPU 1362 y el archivo de registro 1358 es de muy baja latencia. La memoria compartida 1370 puede usarse para permitir la comunicación entre hilos que se ejecutan en las unidades funcionales dentro del multiprocesador de gráficos 1334. La memoria caché 1372 puede usarse como una caché de datos, por ejemplo, para almacenar en caché datos de textura comunicados entre las unidades funcionales y la unidad de textura 1336. La memoria compartida 1370 también se puede utilizar como un programa gestionado en caché. Los hilos que se ejecutan en los núcleos GPGPU 1362 pueden almacenar datos programáticamente dentro de la memoria compartida además de los datos almacenados en caché automáticamente que se almacenan dentro de la memoria caché 1372.

Las **Figuras 14A-14B** ilustran multiprocesadores de gráficos adicionales, de acuerdo con realizaciones. Los multiprocesadores de gráficos 1425, 1450 ilustrados son variantes del multiprocesador de gráficos 1334 de la **Figura 13C**. Los multiprocesadores de gráficos ilustrados 1425, 1450 se pueden configurar como un multiprocesador de transmisión (SM) capaz de ejecutar simultáneamente una gran cantidad de subprocesos de ejecución.

La **Figura 14A** muestra un multiprocesador de gráficos 1425 de acuerdo con una realización adicional. El multiprocesador de gráficos 1425 incluye múltiples instancias adicionales de unidades de recursos de ejecución con respecto al multiprocesador de gráficos 234 de la **Figura 13D**. Por ejemplo, el multiprocesador de gráficos 1425 puede incluir múltiples instancias de la unidad de instrucción 1432A-1432B, del archivo de registro 1434A-1434B y de la o las unidades de textura 1444A-1444B. El multiprocesador de gráficos 1425 también incluye múltiples conjuntos de gráficos o unidades de ejecución de cálculo (por ejemplo, núcleo GPGPU 1436A-1436B, núcleo GPGPU 1437A-1437B, núcleo GPGPU 1438A-1438B) y múltiples conjuntos de unidades de carga/almacenamiento 1440A-1440B. En una realización, las unidades de recurso de ejecución tienen una caché de instrucciones común 1430, una memoria caché de textura y/o de datos 1442 y una memoria compartida 1446.

Los diversos componentes pueden comunicarse mediante una estructura de interconexión 1427. En una realización, la estructura de interconexión 1427 incluye uno o más conmutadores de barra transversal para permitir la comunicación entre los diversos componentes del multiprocesador de gráficos 1425. En una realización, el tejido de interconexión 1427 es una capa de tejido de red de alta velocidad separada sobre la que se apila cada componente del multiprocesador de gráficos 1425. Los componentes del multiprocesador de gráficos 1425 se comunican con componentes remotos mediante la estructura de interconexión 1427. Por ejemplo, cada uno de los núcleos de GPGPU 1436A-1436B, 1437A-1437B y 1478A-1438B puede comunicarse con la memoria compartida 1446 mediante el tejido de interconexión 1427. El tejido de interconexión 1427 puede arbitrar la comunicación dentro del multiprocesador de gráficos 1425 para garantizar una asignación de ancho de banda equitativa entre los componentes.

La **Figura 14B** muestra un multiprocesador de gráficos 1450 de acuerdo con una realización adicional. El procesador de gráficos incluye múltiples conjuntos de recursos de ejecución 1456A-1456D, donde cada conjunto de recursos de ejecución incluye múltiples unidades de instrucciones, archivos de registro, núcleos de GPGPU y unidades de carga-almacenamiento, como se ilustra en la **Figura 13D** y en la **Figura 14A**. Los recursos de ejecución 1456A-1456D pueden funcionar en conjunto con la(s) unidad(es) de textura 1460A-1460D para operaciones de textura, al tiempo que comparten una memoria caché de instrucciones 1454 y una memoria compartida 1462. En una realización, los recursos de ejecución 1456A-1456D pueden compartir una caché de instrucciones 1454 y memoria compartida 1462, así como múltiples instancias de una memoria caché de textura y/o datos 1458A-1458B. Los diversos componentes se pueden comunicar a través de un tejido de interconexión 1452 similar al tejido de interconexión 1427 de la **Figura 14A**.

Los expertos en la materia entenderán que la arquitectura descrita en las **Figuras 12, 13A-13D y 14A-14B** es descriptiva y no limitante en cuanto al alcance de las presentes realizaciones. Por lo tanto, las técnicas descritas en el presente documento pueden implementarse en cualquier unidad de procesamiento configurada apropiadamente, incluyendo, sin limitación, uno o más procesadores de aplicaciones móviles, una o más unidades centrales de procesamiento (CPU) de sobremesa o de servidor, incluyendo CPU de múltiples núcleos, una o más unidades de procesamiento paralelo, tales como la unidad de procesamiento paralelo 1302 de la **Figura 13**, así como uno o más procesadores de gráficos o unidades de procesamiento de propósito especial, sin apartarse del alcance de las realizaciones descritas en el presente documento.

En algunas realizaciones, un procesador paralelo o GPGPU como se describe en el presente documento está acoplado de manera comunicativa a núcleos de anfitrión/procesador para acelerar operaciones de gráficos, operaciones de aprendizaje automático, operaciones de análisis de patrones y diversas funciones de GPU de propósito general (GPGPU). La GPU puede acoplarse de manera comunicativa al procesador/núcleos de anfitrión a través de un bus u otra interconexión (por ejemplo, una interconexión de alta velocidad tal como PCIe o NVLink). En otras realizaciones, la GPU puede integrarse en el mismo paquete o chip que los núcleos y acoplarse de manera comunicativa a los núcleos a través de un bus/interconexión de procesador interno (es decir, internamente al paquete o chip). Independientemente de la manera en la que esté conectada la GPU, los núcleos de procesador pueden asignar trabajo a la GPU en forma de secuencias de comandos/instrucciones contenidas en un descriptor de trabajo. La GPU usa entonces circuitería/lógica dedicada para procesar de manera eficiente estos comandos/instrucciones.

Técnicas para la interconexión de GPU a un procesador de anfitrión

La **Figura 15A** ilustra una arquitectura ejemplar en la que una pluralidad de GPU 1510-1513 están acopladas comunicativamente a una pluralidad de procesadores multinúcleo 1505-1506 a través de enlaces 1540-1543 de alta velocidad (por ejemplo, buses, interconexiones punto a punto, etc.). En una realización, los enlaces de alta velocidad 1540-1543 soportan un rendimiento de comunicaciones de 4 GB/s, 30 GB/s, 80 GB/s o superior, dependiendo de la implementación. Se pueden utilizar diversos protocolos de interconexión que incluyen, entre otros, PCIe 4.0 o 5.0 y NVLink 2.0. Sin embargo, los principios subyacentes de la invención no están limitados a ningún protocolo o rendimiento de comunicaciones particular.

Además, en una realización, dos o más de las GPU 1510-1513 están interconectadas a través de enlaces de alta velocidad 1544-1545, que pueden implementarse usando protocolos/enlaces iguales o diferentes a los usados para los enlaces de alta velocidad 1540-1543. De manera similar, dos o más de los procesadores de múltiples núcleos 1505-1506 pueden conectarse a través del enlace de alta velocidad 1533, que pueden ser buses de multiprocesador simétrico (SMP) que operan a 20 GB/s, 30 GB/s, 120 GB/s o superior. Como alternativa, toda la comunicación entre los diversos componentes de sistema que se muestran en la **Figura 15A** se puede lograr usando los mismos protocolos/enlaces (por ejemplo, a través de una estructura de interconexión común). Sin embargo, como se mencionó, los principios subyacentes de la invención no se limitan a ningún tipo particular de tecnología de interconexión.

En una realización, cada procesador de múltiples núcleos 1505-1506 está acoplado de manera comunicativa a una memoria de procesador 1501-1502, por medio de interconexiones de memoria 1530-1531, respectivamente, y cada GPU 1510-1513 está acoplada de manera comunicativa a una memoria de GPU 1520-1523 a través de interconexiones de memoria de GPU 1550-1553, respectivamente. Las interconexiones de memoria 1530-1531 y 1550-1553 pueden utilizar las mismas tecnologías de acceso de memoria u otras diferentes. A modo de ejemplo, y no

de limitación, las memorias de procesador 1501-1502 y las memorias de GPU 1520-1523 pueden ser memorias volátiles, tal como memorias de acceso aleatorio dinámicas (DRAM) (incluyendo DRAM apiladas), SDRAM DDR de gráficos (GDDR) (por ejemplo, GDDR5, GDDR6), o Memoria de Alto Ancho de Banda (HBM) y/o pueden ser memorias no volátiles, tales como 3D XPoint o Nano-Ram. En una realización, una porción de las memorias puede ser memoria volátil y otra porción puede ser memoria no volátil (por ejemplo, usando una jerarquía de memoria de dos niveles (2LM)).

Como se describe a continuación, aunque los diversos procesadores 1505-1506 y las GPU 1510-1513 pueden estar físicamente acoplados a una memoria particular 1501-1502, 1520-1523, respectivamente, puede implementarse una arquitectura de memoria unificada en la que el mismo espacio de direcciones de sistema virtual (también denominado espacio "de direcciones eficaces") está distribuido entre todas las diversas memorias físicas. Por ejemplo, las memorias de procesador 1501-1502 pueden comprender cada una 64 GB del espacio de direcciones de la memoria del sistema y las memorias de GPU 1520-1523 pueden comprender cada una 32 GB del espacio de direcciones de la memoria del sistema (dando como resultado un total de 256 GB de memoria direccionable en este ejemplo).

La **Figura 15B** ilustra detalles adicionales para una interconexión entre un procesador multinúcleo 1507 y un módulo de aceleración de gráficos 1546 de acuerdo con una realización. El módulo de aceleración de gráficos 1546 puede incluir uno o más chips de GPU integrados en una tarjeta de línea que está acoplada al procesador 1507 por medio del enlace de alta velocidad 1540. Como alternativa, el módulo de aceleración de gráficos 1546 puede integrarse en el mismo paquete o chip que el procesador 1507.

El procesador 1507 ilustrado incluye una pluralidad de núcleos 1560A-1560D, cada uno con una memoria intermedia de búsqueda de traducción 1561A-1561D y una o más caché 1562A-1562D. Los núcleos pueden incluir diversos otros componentes para ejecutar instrucciones y procesar datos que no se ilustran para evitar complicar los principios subyacentes de la invención (por ejemplo, unidades de búsqueda de instrucciones, unidades de predicción de rama, decodificadores, unidades de ejecución, memoria intermedia de reordenamiento, etc.). Las cachés 1562A-1562D pueden comprender cachés de nivel 1 (L1) y de nivel 2 (L2). Además, una o más memorias caché compartidas 1526 pueden incluirse en la jerarquía de almacenamiento en memoria caché y ser compartidas por conjuntos de núcleos 1560A-1560D. Por ejemplo, una realización del procesador 1507 incluye 24 núcleos, cada uno con su propia caché L1, doce cachés L2 compartidas y doce cachés L3 compartidas. En esta realización, una de las caché L2 y L3 es compartida por dos núcleos adyacentes. El procesador 1507 y el módulo de aceleración de gráficos 1546 se conectan con la memoria del sistema 1541, que puede incluir memorias de procesador 1501-1502.

La coherencia se mantiene para los datos e instrucciones almacenados en las diversas caché 1562A-1562D, 1556 y la memoria del sistema 1541 mediante comunicación entre núcleos a través de un bus de coherencia 1564. Por ejemplo, cada caché puede tener una lógica/circuitería de coherencia de caché asociada con la misma con la que comunicarse a través del bus de coherencia 1564 en respuesta a lecturas o escrituras detectadas en líneas de caché particulares. En una implementación, se implementa un protocolo de fisgoneo de caché a través del bus de coherencia 1564 para fisgar accesos de caché. Las técnicas de fisgoneo/coherencia de caché son bien conocidas por los expertos en la materia y no se describirán en detalle en el presente caso para evitar complicar los principios subyacentes de la invención.

En una realización, un circuito intermediario 1525 acopla de manera comunicativa el módulo de aceleración de gráficos 1546 al bus de coherencia 1564, permitiendo que el módulo de aceleración de gráficos 1546 participe en el protocolo de coherencia de caché como un homólogo de los núcleos. En particular, una interfaz 1535 proporciona conectividad al circuito intermediario 1525 a través del enlace de alta velocidad 1540 (por ejemplo, un bus PCIe, NVLink, etc.) y una interfaz 1537 conecta el módulo de aceleración de gráficos 1546 al enlace de alta velocidad 1540.

En una implementación, un circuito de integración de acelerador 1536 proporciona servicios de gestión de memoria caché, de acceso a memoria, de gestión de contexto y de gestión de interrupciones en beneficio de una pluralidad de motores de procesamiento de gráficos 1531, 1532, N del módulo de aceleración de gráficos 1546. Cada uno de los motores de procesamiento de gráficos 1531, 1532, N puede comprender una unidad de procesamiento de gráficos (GPU) separada. Como alternativa, los motores de procesamiento de gráficos 1531, 1532, N pueden comprender diferentes tipos de motor de procesamiento de gráficos dentro de una GPU, tal como las unidades de ejecución de gráficos, los motores de procesamiento de medios (por ejemplo, codificadores/decodificadores de vídeo), muestreadores y motores blit. En otras palabras, el módulo de aceleración de gráficos puede ser una GPU con una pluralidad de motores de procesamiento de gráficos 1531-1532, N, o los motores de procesamiento de gráficos 1531-1532, N, pueden ser GPU individuales integradas en un paquete, tarjeta de línea o chip común.

En una realización, el circuito de integración de acelerador 1536 incluye una unidad de gestión de memoria (MMU) 1539 para realizar diversas funciones de gestión de memoria tales como traducciones de memoria virtual a física (también denominadas traducciones de memoria efectiva a real) y protocolos de acceso de memoria para acceder a la memoria de sistema 1541. La MMU 1539 puede incluir también una memoria intermedia de traducción anticipada (TLB) (no mostrada) para almacenar en caché las traducciones de dirección virtual/efectiva a física/real. En una implementación, una caché 1538 almacena órdenes y datos para un acceso efectivo por los motores de procesamiento gráfico 1531-1532, N. En una realización, los datos almacenados en la caché 1538 y en las memorias de gráficos

1533-1534, N se mantienen coherentes con las caché de núcleo 1562A-1562D, 1556 y la memoria de sistema 1511. Según se mencionó, esto se puede conseguir mediante el circuito de intermediario 1525 que toma parte en el mecanismo de coherencia de caché en nombre de la caché 1538 y las memorias 1533-1534, N (por ejemplo, enviando actualizaciones a la caché 1538 relacionadas con las modificaciones/accesos de líneas de caché en las caché del procesador 1562A-1562D, 1556 y recibiendo actualizaciones de la caché 1538).

Un conjunto de registros 1545 almacenan datos de contexto para subprocesos ejecutados por los motores de procesamiento gráfico 1531-1532, N y un circuito de gestión de contexto 1548 gestiona los contextos de subproceso. Por ejemplo, el circuito de gestión de contexto 1548 puede llevar a cabo operaciones de grabado y restaurado para grabar y restaurar contextos de los diversos subprocesos durante cambios de contexto (por ejemplo, donde un primer subproceso se graba y un segundo subproceso se almacena de modo que el segundo subproceso pueda ser ejecutado por un motor de procesamiento gráfico). Por ejemplo, en un cambio de contexto, el circuito de gestión de contexto 1548 puede almacenar valores de registro actuales en una región designada en la memoria (por ejemplo, identificada por un puntero de contexto). A continuación, puede restablecer los valores de registro cuando se vuelve al contexto. En una realización, un circuito de gestión de interrupciones 1547 recibe y procesa interrupciones recibidas desde los dispositivos de sistema.

En una implementación, las direcciones virtuales/efectivas de un motor de procesamiento gráfico 1531 se traducen a direcciones reales/físicas en memoria de sistema 1511 por la MMU 1539. Una realización del circuito de integración de acelerador 1536 soporta múltiples (por ejemplo, 4, 8, 16) módulos de aceleración de gráficos 1546 y/u otros dispositivos aceleradores. El módulo de aceleración de gráficos 1546 puede estar dedicado a una única aplicación ejecutada en el procesador 1507 o puede compartirse entre múltiples aplicaciones. En una realización, se presenta un entorno de ejecución de gráficos virtualizados en el que los recursos de los motores de procesamiento de gráficos 1531-1532, N se comparten con múltiples aplicaciones o máquinas virtuales (VM). Los recursos pueden subdividirse en "cortes" que se asignan a diferentes VM y/o aplicaciones basándose en los requisitos de procesamiento y las propiedades asociadas con las VM y/o las aplicaciones.

Por tanto, el circuito de integración de acelerador actúa como un puente al sistema para el módulo de aceleración de gráficos 1546 y proporciona servicios de traducción de direcciones y de caché de sistema. Además, el circuito de integración de acelerador 1536 puede proporcionar funciones de virtualización para que el procesador anfitrión gestione la virtualización de los motores de procesamiento de gráficos, las interrupciones y la gestión de memoria.

Debido a que los recursos de hardware de los motores de procesamiento de gráficos 1531-1532, N se mapean explícitamente al espacio de direcciones real visto por el procesador de anfitrión 1507, cualquier procesador de anfitrión puede direccionar estos recursos directamente usando un valor de dirección efectivo. Una función del circuito de integración de acelerador 1536, en una realización, es la separación física de los motores de procesamiento de gráficos 1531-1532, N para que aparezcan ante el sistema como unidades independientes.

Como se menciona, en la realización ilustrada, una o más memorias de gráficos 1533-1534, M están acopladas a cada uno de los motores de procesamiento de gráficos 1531-1532, N, respectivamente. Las memorias de gráficos 1533-1534, M almacenan instrucciones y datos que son procesados por cada uno de los motores de procesamiento de gráficos 1531-1532, N. Las memorias de gráficos 1533-1534, M pueden ser memorias volátiles, tales como DRAM (incluyendo DRAM apiladas), memoria GDDR (por ejemplo, GDDR5, GDDR6) o HBM y/o pueden ser memorias no volátiles, tales como 3D XPoint o Nano-Ram.

En una realización, para reducir el tráfico de datos a través del enlace 1540, se utilizan técnicas de desvío para garantizar que los datos almacenados en las memorias gráficas 1533-1534, M son datos que serán utilizados con mayor frecuencia por los motores de procesamiento gráfico 1531-1532, N y preferiblemente no utilizados por los núcleos 1560A-1560D (al menos no con frecuencia). Del mismo modo, el mecanismo de desvío intenta mantener los datos que necesitan los núcleos (y, preferiblemente, no los motores de procesamiento gráfico 1531-1532, N) dentro de las caché 1562A-1562D, 1556 de los núcleos y la memoria de sistema 1511.

La **Figura 15C** ilustra otra realización en la que el circuito de integración del acelerador 1536 está integrado dentro del procesador 1507. En esta realización, los motores de procesamiento de gráficos 1531-1532, N se comunican directamente a través del enlace de alta velocidad 1540 con el circuito de integración del acelerador 1536 a través de la interfaz 1537 y la interfaz 1535 (que, nuevamente, se puede utilizar cualquier forma de bus o protocolo de interfaz). El circuito de integración del acelerador 1536 puede realizar las mismas operaciones que las descritas con respecto a la **Figura 15B**, pero potencialmente con un rendimiento mayor dada su proximidad al bus de coherencia 1562 y las memorias caché 1562A-1562D, 1526.

Una realización admite diferentes modelos de programación, incluido un modelo de programación de proceso dedicado (sin virtualización del módulo de aceleración de gráficos) y modelos de programación compartidos (con virtualización). Este último puede incluir modelos de programación que son controlados por el circuito de integración de acelerador 1536 y modelos de programación que son controlados por el módulo de aceleración de gráficos 1546.

En una realización del modelo de proceso dedicado, los motores de procesamiento de gráficos 1531-1532, N están dedicados a una única aplicación o proceso bajo un único sistema operativo. La aplicación única puede canalizar otras solicitudes de aplicación a los motores de procesamiento de gráficos 1531-1532, N, proporcionando virtualización dentro de una VM/partición.

En los modelos de programación de procesos dedicados, los motores de procesamiento de gráficos 1531-1532, N, pueden ser compartidos por múltiples particiones de VM/aplicación. Los modelos compartidos requieren que un hipervisor de sistema virtualice los motores de procesamiento de gráficos 1531-1532, N para permitir el acceso por cada sistema operativo. Para sistemas de subdivisión única sin un hipervisor, los motores de procesamiento de gráficos 1531-1532, N son propiedad del sistema operativo. En ambos casos, el sistema operativo puede virtualizar los motores de procesamiento de gráficos 1531-1532, N para proporcionar acceso a cada proceso o aplicación.

Para el modelo de programación compartida, el módulo de aceleración de gráficos 1546 o un motor de procesamiento de gráficos 1531-1532, N individual selecciona un elemento de proceso usando un manejador de proceso. En una realización, los elementos de proceso se almacenan en la memoria de sistema 1511 y se pueden direccionar utilizando las técnicas de traducción de dirección eficaz a dirección real descritas en el presente documento. El manejador de proceso puede ser un valor específico de la implementación proporcionado al proceso de anfitrión cuando se registra su contexto con el motor de procesamiento de gráficos 1531-1532, N (es decir, llamando al software de sistema para añadir el elemento de proceso a la lista vinculada de elementos de proceso). Los 16 bits inferiores del manejador de proceso pueden ser el desplazamiento del elemento de proceso dentro de la lista vinculada de elementos de proceso.

La **Figura 15D** ilustra un corte de integración de acelerador ilustrativa 1590. Como se usa en el presente documento, un "segmento" comprende una porción específica de los recursos de procesamiento del circuito de integración de acelerador 1536. El espacio de direcciones efectivo de la aplicación 1582 dentro de la memoria de sistema 1511 almacena elementos de proceso 1583. En una realización, los elementos de proceso 1583 se almacenan en respuesta a las invocaciones de GPU 1581 desde las aplicaciones 1580 ejecutadas en el procesador 1507. Un elemento de proceso 1583 contiene el estado de proceso para la aplicación 1580 correspondiente. Un descriptor de trabajo (WD) 1584 contenido en el elemento de proceso 1583 puede ser un único trabajo solicitado por una aplicación o puede contener un puntero a una cola de trabajos. En este último caso, el WD 1584 es un puntero a la cola de solicitudes de trabajos en el espacio de direcciones 1582 de la aplicación.

El módulo de aceleración de gráficos 1546 y/o los motores de procesamiento de gráficos 1531-1532, N individuales pueden ser compartidos por todos, o por un subconjunto de, los procesos en el sistema. Las realizaciones de la invención incluyen una infraestructura para configurar el estado de proceso y enviar un WD 1584 a un módulo de aceleración de gráficos 1546 para empezar un trabajo en un entorno virtualizado.

En una implementación, el modelo de programación de proceso dedicado es específico para la implementación. En este modelo, un único proceso posee el módulo de aceleración de gráficos 1546 o un motor de procesamiento de gráficos 1531 individual. Debido a que el módulo de aceleración de gráficos 1546 es propiedad de un único proceso, el hipervisor inicializa el circuito de integración de acelerador 1536 para la subdivisión propietaria y el sistema operativo inicializa el circuito de integración de acelerador 1536 para el proceso propietario en el momento en el que se asigna el módulo de aceleración de gráficos 1546.

En la operación, una unidad de extracción de WD 1591 en el corte de integración de acelerador 1590 extrae el siguiente WD 1584 que incluye una indicación del trabajo que va a hacerse por uno de los motores de procesamiento de gráficos del módulo de aceleración de gráficos 1546. Los datos del WD 1584 pueden almacenarse en los registros 1545 y usarse por la MMU 1539, el circuito de gestión de interrupciones 1547 y/o el circuito de gestión de contexto 1546 como se ilustra. Por ejemplo, una realización de la MMU 1539 incluye circuitería de recorrido de páginas/segmentos para acceder a las tablas de segmentos/páginas 1586 dentro del espacio de direcciones virtuales del SO 1585. El circuito de gestión de interrupciones 1547 puede procesar eventos de interrupción 1592 recibidos desde el módulo de aceleración de gráficos 1546. Cuando se realizan operaciones gráficas, una dirección efectiva 1593 generada por un motor de procesamiento de gráficos 1531-1532, N se convierte en una dirección real por medio de la MMU 1539.

En una realización, se duplica el mismo conjunto de registros 1545 para cada motor de procesamiento de gráficos 1531-1532, N y/o módulo de aceleración de gráficos 1546 y puede inicializarse por el hipervisor o el sistema operativo. Cada uno de estos registros duplicados puede incluirse en un corte de integración de acelerador 1590. Se muestran los registros ilustrativos que pueden inicializarse por el hipervisor en la **Tabla 1**.

Tabla 1 - Registros inicializados por el hipervisor

1	Registro de control de corte
2	Puntero de área de procesos planificados de dirección real (RA)
3	Registro de anulación de máscara de autoridad

4	Desplazamiento de entrada de tabla de vectores de interrupción
5	Límite de entrada de tabla de vectores de interrupción
6	Registro de estado
7	ID de subdivisión lógica
8	Puntero de registro de utilización del acelerador del hipervisor de dirección real (RA)
9	Registro de descripción de almacenamiento

En la **Tabla 2** se muestran los registros ilustrativos que pueden inicializarse por el sistema operativo.

Tabla 2 - Registros inicializados por sistema operativo

1	Identificación de procesos e hilos
2	Puntero de guardar/restaurar contexto de dirección efectiva (EA)
3	Puntero de registro de utilización de acelerador de dirección virtual (VA)
4	Puntero de tabla de segmentos de almacenamiento de dirección virtual (VA)
5	Máscara de autoridad
6	Descriptor de trabajo

En una realización, cada WD 1584 es específico a un módulo de aceleración de gráficos particular 1546 y/o al motor de procesamiento de gráficos 1531-1532, N. Contiene toda la información que requiere un motor de procesamiento de gráficos 1531-1532, N para hacer su trabajo o puede ser un puntero a una ubicación de memoria donde la aplicación ha establecido una cola de comandos de trabajo para que se complete.

La **Figura 15E** ilustra detalles adicionales para una realización de un modelo compartido. Esta realización incluye un espacio de direcciones reales de hipervisor 1598 en el que se almacena una lista de elementos de proceso 1599. El espacio de direcciones real de hipervisor 1598 es accesible mediante un hipervisor 1596 que virtualiza los motores de módulo de aceleración de gráficos para el sistema operativo 1595.

Los modelos de programación compartida permiten que todos o un subconjunto de procesos de todas o un subconjunto de subdivisiones en el sistema usen un módulo de aceleración de gráficos 1546. Hay dos modelos de programación donde el módulo de aceleración de gráficos 1546 se comparte por múltiples procesos y subdivisiones: compartido en cortes de tiempo y compartido dirigido a gráficos.

En este modelo, el hipervisor de sistema 1596 tiene propiedad del módulo de aceleración de gráficos 1546 y hace que su función esté disponible para todos los sistemas operativos 1595. Para que un módulo de aceleración de gráficos 1546 soporte la virtualización por el hipervisor de sistema 1596, el módulo de aceleración de gráficos 1546 puede adherirse a los siguientes requisitos: 1) La solicitud de trabajo de una aplicación debe ser autónoma (es decir, no es necesario mantener el estado entre trabajos) o el módulo de aceleración de gráficos 1546 debe proporcionar un mecanismo de almacenamiento y restauración de contexto. 2) El módulo de aceleración de gráficos 1546 garantiza que la solicitud de trabajo de una aplicación se completa en una cantidad especificada de tiempo, incluyendo cualquier fallo de conversión, o el módulo de aceleración de gráficos 1546 proporciona la capacidad de dar prioridad al procesamiento del trabajo. 3) Se ha de garantizar al módulo de aceleración de gráficos 1546 la equidad entre procesos cuando se opera en el modelo de programación compartido dirigido.

En una realización, para el modelo compartido, se requiere que la aplicación 1580 haga una llamada de sistema al sistema operativo 1595 con un tipo de módulo de aceleración de gráficos 1546, un descriptor de trabajo (WD), un valor de registro de máscara de autoridad (AMR) y un puntero de área de grabación/restauración de contexto (CSRP). El tipo del módulo de aceleración de gráficos 1546 describe la función de aceleración dirigida como objetivo para la llamada de sistema. El tipo del módulo de aceleración de gráficos 1546 puede ser un valor específico de sistema. El WD se formatea específicamente para el módulo de aceleración de gráficos 1546 y puede estar en forma de un comando de módulo de aceleración de gráficos 1546, un puntero de dirección efectiva a una estructura definida por el usuario, un puntero de dirección efectiva a una cola de comandos o cualquier otra estructura de datos para describir el trabajo que va a hacerse por el módulo de aceleración de gráficos 1546. En una realización, el valor de AMR es el estado de AMR a utilizar para el proceso actual. El valor pasado al sistema operativo es similar a una aplicación que configura el AMR. Si las implementaciones del circuito de integración de acelerador 1536 y del módulo de aceleración de gráficos 1546 no soportan un registro de anulación de máscara de autoridad de usuario (UAMOR), el sistema operativo puede aplicar el valor de UAMOR actual al valor de AMR antes de pasar el AMR en la llamada de hipervisor. El hipervisor 1596 puede aplicar opcionalmente el valor actual del registro de anulación de máscara de autoridad (AMOR) antes de colocar el AMR en el elemento de proceso 1583. En una realización, el CSRP es uno de los registros

1545 que contienen la dirección efectiva de un área en el espacio de direcciones de la aplicación 1582 para que el módulo de aceleración de gráficos 1546 grabe y restablezca el estado de contexto. Este puntero es opcional si no es necesario guardar un estado entre trabajos o cuando se adelanta un trabajo. El área de guardado/restauración de contexto puede estar fijada en la memoria de sistema.

5 Tras recibir la llamada de sistema, el sistema operativo 1595 puede verificar que se ha registrado la aplicación 1580 y que se le ha dado la autoridad para usar el módulo de aceleración de gráficos 1546. El sistema operativo 1595, a continuación, llama al hipervisor 1596 con la información mostrada en la **Tabla 3**.

10 **Tabla 3** - Parámetros de llamada de SO al hipervisor

1	Un descriptor de trabajo (WD)
2	Un valor de registro de máscara de autoridad (AMR) (potencialmente enmascarado).
3	Un puntero de área de grabación/restauración de contexto (CSRP) de dirección efectiva (EA)
4	Un ID de proceso (PID) e ID de hilo (TID) opcional
5	Un puntero de registro de utilización de acelerador (AURP) de dirección virtual (VA)
6	La dirección virtual del puntero de tabla de segmento de almacenamiento (SSTP)
7	Un número de servicio de interrupción lógica (LISN)

15 Tras recibir la llamada de hipervisor, el hipervisor 1596 verifica que se ha registrado el sistema operativo 1595 y se le ha dado la autoridad para usar el módulo de aceleración de gráficos 1546. El hipervisor 1596 pone, a continuación, el elemento de proceso 1583 en la lista vinculada de elementos de proceso para el tipo de módulo de aceleración de gráficos 1546 correspondiente. El elemento de proceso puede incluir la información mostrada en la **Tabla 4**.

Tabla 4 - Información de elemento de proceso

1	Un descriptor de trabajo (WD)
2	Un valor de registro de máscara de autoridad (AMR) (potencialmente enmascarado).
3	Un puntero de área de grabación/restauración de contexto (CSRP) de dirección efectiva (EA)
4	Un ID de proceso (PID) e ID de hilo (TID) opcional
5	Un puntero de registro de utilización de acelerador (AURP) de dirección virtual (VA)
6	La dirección virtual del puntero de tabla de segmento de almacenamiento (SSTP)
7	Un número de servicio de interrupción lógica (LISN)
8	Tabla de vectores de interrupción, derivada de los parámetros de llamada de hipervisor.
9	Un valor de registro de estado (SR)
10	Un ID de subdivisión lógica (LPID)
11	Un puntero de registro de utilización de acelerador de hipervisor de dirección real (RA)
12	El registro de descriptor de almacenamiento (SDR)

20 En una realización, el hipervisor inicializa una pluralidad de registros 1545 de corte de integración de acelerador 1590.

25 Como se ilustra en la **Figura 15F**, una realización de la invención emplea una memoria unificada direccionable por medio de un espacio de direcciones de memoria virtual común usado para acceder a las memorias físicas de procesador 1501-1502 y las memorias de GPU 1520-1523. En esta implementación, las operaciones ejecutadas en las GPU 1510-1513 utilizan el mismo espacio de direcciones de memoria virtuales/efectivas para acceder a las memorias de procesador 1501-1502 y viceversa, simplificando así la programabilidad. En una realización, una primera parte del espacio de direcciones virtuales/eficaces está asignada a la memoria de procesador 1501, una segunda parte a la segunda memoria de procesador 1502, una tercera parte a la memoria de GPU 1520, y así sucesivamente.

30 El espacio de memoria virtual/efectivo total (denominado, en ocasiones, el espacio de direcciones efectivo) está distribuido, de esta manera, a lo largo de cada una de las memorias de procesador 1501-1502 y de las memorias de GPU 1520-1523, permitiendo que cualquier procesador o GPU acceda a cualquier memoria física con una dirección virtual asignada a esa memoria.

En una realización, la circuitería de gestión de desvío/coherencia 1594A-1594E dentro de una o más de las MMU 1539A-1539E garantiza la coherencia de caché entre las caché de los procesadores de anfitrión (p. ej., 1505) y las GPU 1510-1513 e implementa técnicas de desvío que indican las memorias físicas en las que deberían almacenarse ciertos tipos de datos. Si bien se ilustran múltiples instancias de circuitos de gestión de polarización/coherencia 1594A-1594E en **Figura 15F**, el circuito de polarización/coherencia puede implementarse dentro de la MMU de uno o más procesadores principales 1505 y/o dentro del circuito de integración del acelerador 1536.

Una realización permite que la memoria conectada a GPU 1520-1523 se asigne como parte de la memoria del sistema y se acceda a ella utilizando tecnología de memoria virtual compartida (SVM), pero sin sufrir los típicos inconvenientes de rendimiento asociados con la coherencia total de la caché del sistema. La capacidad de acceder a la memoria conectada a GPU 1520-1523 como memoria del sistema sin una onerosa sobrecarga de coherencia de caché proporciona un entorno operativo beneficioso para la descarga de la GPU. Esta disposición permite que el software del procesador de anfitrión 1505 establezca operandos y acceda a resultados de cálculo, sin la sobrecarga de copias de datos de DMA de E/S tradicionales. Tales copias tradicionales implican llamadas de controlador, interrupciones y accesos de E/S asignados con memoria (MMIO) que son, todos ellos, ineficientes en relación con los accesos de memoria sencillos. Al mismo tiempo, la capacidad de acceder a la memoria adjunta a la GPU 1520-1523 sin sobrecargas de coherencia de caché puede ser crítica para el tiempo de ejecución de un cálculo descargado. En casos con tráfico de memoria de escritura de transmisión por flujo continuo sustancial, por ejemplo, la sobrecarga de coherencia de caché puede reducir significativamente el ancho de banda de escritura eficaz observado por una GPU 1510-1513. La eficiencia del establecimiento de operandos, la eficiencia del acceso a resultados y la eficiencia del cálculo de GPU desempeñan, todas ellas, un papel en la determinación de la eficacia de la descarga de GPU.

En una implementación, la selección entre el desvío de GPU y el desvío de procesador de anfitrión se controla por una estructura de datos de rastreador de desvío. Se puede usar una tabla de desvío, por ejemplo, que puede ser una estructura granular de página (es decir, controlada en la granularidad de una página de memoria) que incluye 1 o 2 bits por página de memoria adjunta a la GPU. La tabla de desvío puede implementarse en un intervalo de memoria robado de una o más memorias anexadas a GPU 1520-1523, con o sin una caché de desvío en la GPU 1510-1513 (por ejemplo, para almacenar en caché entradas usadas de manera frecuente/reciente de la tabla de desvío). Como alternativa, toda la tabla de desvío puede mantenerse dentro de la GPU.

En una implementación, se accede a la entrada de tabla de desvío asociada con cada acceso a la memoria adjunta a la GPU 1520-1523 antes del acceso real a la memoria de GPU, provocando las siguientes operaciones. En primer lugar, las solicitudes locales de la GPU 1510-1513 que encuentran su página en el desvío de GPU se reenvían directamente a una correspondiente memoria de GPU 1520-1523. Las solicitudes locales desde la GPU que encuentran su página en el desvío de anfitrión se reenvían al procesador 1505 (por ejemplo, a través de un enlace de alta velocidad como se ha analizado anteriormente). En una realización, las solicitudes del procesador 1505 que encuentran la página solicitada en el desvío de procesador de anfitrión completan la solicitud como una lectura de memoria normal. Como alternativa, solicitudes dirigidas a una página con desvío de GPU pueden redirigirse a la GPU 1510-1513. A continuación, la GPU puede hacer que la página pase a un desvío de procesador anfitrión si no está usando actualmente la página.

El estado de desvío de una página se puede cambiar mediante un mecanismo basado en software, un mecanismo basado en software asistido por hardware o, para un conjunto limitado de casos, un mecanismo puramente basado en hardware.

Un mecanismo para cambiar el estado de desvío emplea una llamada API (por ejemplo, OpenCL), que, a su vez, llama al controlador de dispositivo de la GPU que, a su vez, envía un mensaje (o pone en cola un descriptor de comando) a la GPU indicándole que cambie el estado de desvío y, para algunas transiciones, realice una operación de vaciado de caché en el anfitrión. Se requiere la operación de vaciado de caché para una transición desde el procesador de anfitrión 1505 a un desvío de GPU, pero no se requiere para la transición opuesta.

En una realización, la coherencia de caché se mantiene haciendo temporalmente que las páginas con desvío de GPU no puedan ser almacenadas en caché por el procesador de anfitrión 1505. Para acceder a estas páginas, el procesador 1505 puede solicitar acceso desde la GPU 1510 que puede conceder, o no, acceso de manera inmediata, dependiendo de la implementación. Por lo tanto, para reducir la comunicación entre el procesador 1505 y la GPU 1510, es beneficioso garantizar que las páginas con desvío de GPU son aquellas que son requeridas por la GPU, pero no por el procesador de anfitrión 1505, y viceversa.

Canalización de procesamiento de gráficos

La **Figura 16** ilustra una canalización de procesamiento de gráficos 1600, de acuerdo con una realización. En una realización, un procesador de gráficos puede implementar el canal de procesamiento de gráficos 1600 ilustrado. El procesador de gráficos puede incluirse dentro de los subsistemas de procesamiento paralelo como se describe en el presente documento, tal como el procesador paralelo 1300 de la **Figura 13**, que, en una realización, es una variante del/de los procesador(es) paralelo(s) 1212 de la **Figura 12**. Los diversos sistemas de procesamiento paralelo pueden implementar la canalización de procesamiento de gráficos 1600 mediante una o más instancias de la unidad de

procesamiento paralelo (por ejemplo, la unidad de procesamiento paralelo 1302 de la **Figura 13**) como se describe en el presente documento. Por ejemplo, una unidad sombreadora (por ejemplo, el multiprocesador de gráficos 1334 de la **Figura 13**) puede estar configurada para realizar las funciones de una o más de una unidad de procesamiento de vértices 1604, una unidad de procesamiento de control de teselación 1608, una unidad de procesamiento de evaluación de teselación 1612, una unidad de procesamiento de geometría 1616 y una unidad de procesamiento de fragmentos/píxeles 1624. Las funciones del ensamblador de datos 1602, los ensambladores de primitivas 1606, 1614, 1618, la unidad de teselación 1610, el rasterizador 1622, y la unidad de operaciones de rasterización 1626 pueden realizarse también por otros motores de procesamiento dentro de una agrupación de procesamiento (por ejemplo, la agrupación de procesamiento 1314 de la **Figura 13**) y una unidad de subdivisión correspondiente (por ejemplo, la unidad de subdivisión 1320A-1320N de la **Figura 13**). La canalización de procesamiento de gráficos 1600 puede implementarse también usando unidades de procesamiento especializadas para una o más funciones. En una realización, una o más porciones de la canalización de procesamiento de gráficos 1600 pueden realizarse mediante una lógica de procesamiento paralelo dentro de un procesador de propósito general (por ejemplo, una CPU). En una realización, una o más porciones de la canalización de procesamiento de gráficos 1600 pueden acceder a una memoria en chip (por ejemplo, la memoria de procesador paralelo 1322 como en la **Figura 13**) mediante una interfaz de memoria 1628, que puede ser una instancia de la interfaz de memoria 1318 de la **Figura 13**.

En una realización, el ensamblador de datos 1602 es una unidad de procesamiento que recopila datos de vértice para superficies y primitivas. El ensamblador de datos 1602 emite entonces los datos de vértice, incluyendo los atributos de vértice, a la unidad de procesamiento de vértices 1604. La unidad de procesamiento de vértices 1604 es una unidad de ejecución programable que ejecuta programas de sombreado de vértices, iluminando y transformando los datos de vértices según lo especificado por los programas de sombreado de vértices. La unidad de procesamiento de vértices 1604 lee datos que están almacenados en la memoria caché, local o del sistema para su uso en el procesamiento de los datos de vértices y se puede programar para transformar los datos de vértices desde una representación de coordenadas basada en objetos a un espacio de coordenadas del espacio global o un espacio de coordenadas de dispositivo normalizado.

Una primera instancia de un ensamblador de primitivas 1606 recibe atributos de vértices desde la unidad de procesamiento de vértices 1604. El ensamblador de primitivas 1606 lee atributos de vértices almacenados según sea necesario y construye primitivas de gráficos para su procesamiento por la unidad de procesamiento de control de teselación 1608. Las primitivas de gráficos incluyen triángulos, segmentos de línea, puntos, parches y así sucesivamente de acuerdo con son soportados por varias interfaces de programación de aplicaciones (API) de procesamiento de gráficos.

La unidad de procesamiento de control de teselación 1608 trata los vértices de entrada como puntos de control para un parche geométrico. Los puntos de control se transforman de una representación de entrada a partir del parche (por ejemplo, las bases del parche) a una representación que es adecuada para su uso en una evaluación superficial por la unidad de procesamiento de evaluación de teselación 1612. La unidad de procesamiento de control de teselación 1608 también puede calcular factores de teselación para bordes de parches geométricos. Un factor de teselación es aplicable a un único borde y cuantifica un nivel de detalle, dependiente de la vista, asociado con el borde. Una unidad de teselación 1610 está configurada para recibir los factores de teselación para bordes de un parche y para teselar el parche en múltiples primitivas geométricas, tales como primitivas de línea, de triángulo o cuadriláteros, que se transmiten a una unidad de procesamiento de evaluación de teselación 1612. La unidad de procesamiento de evaluación de teselación 1612 opera sobre coordenadas parametrizadas del parche subdividido para generar una representación superficial y atributos de vértice para cada vértice asociado con las primitivas geométricas.

Una segunda instancia de un ensamblador de primitivas 1614 recibe atributos de vértices desde la unidad de procesamiento de evaluación de teselación 1612, que lee los atributos de vértices almacenados de acuerdo con sea necesario y construye primitivas de gráficos para su procesamiento por la unidad de procesamiento de geometría 1616. La unidad de procesamiento de geometría 1616 es una unidad de ejecución programable que ejecuta programas sombreadores de geometría para transformar primitivas de gráficos recibidas desde el ensamblador de primitivas 1614 según se especifica por los programas sombreadores de geometría. En una realización, la unidad de procesamiento de geometría 1616 está programada para subdividir las primitivas de gráficos en una o más primitivas de gráficos nuevas y calcular parámetros usados para rasterizar las nuevas primitivas de gráficos.

En algunas realizaciones, la unidad de procesamiento de geometría 1616 puede añadir o borrar elementos en el flujo de geometría. La unidad de procesamiento de geometría 1616 emite los parámetros y vértices que especifican primitivas de gráficos nuevas al ensamblador de primitivas 1618. El ensamblador de primitivas 1618 recibe los parámetros y vértices desde la unidad de procesamiento de geometría 1616 y construye primitivas de gráficos para su procesamiento por una unidad de escala, selección y recorte de ventana gráfica 1620. La unidad de procesamiento de geometría 1616 lee datos que están almacenados en la memoria del procesador en paralelo o en la memoria de sistema para su utilización en el procesamiento de los datos de geometría. La unidad de escalado, selección y recorte de la ventana gráfica 1620 lleva a cabo el recorte, la selección y el escalado de la ventana y envía las primitivas de gráficos procesadas a un rasterizador 1622.

El rasterizador 1622 puede llevar a cabo una selección de profundidad y otras optimizaciones basadas en la profundidad. El rasterizador 1622 también lleva a cabo la conversión de exploración en las nuevas primitivas de gráficos para generar fragmentos y emitir aquellos fragmentos y datos de cobertura asociados a la unidad de procesamiento de fragmentos/píxeles 1624. La unidad de procesamiento de fragmentos/píxeles 1624 es una unidad de ejecución programable que se configura para ejecutar programas de sombreado de fragmentos o programas de sombreado de píxeles. La unidad de procesamiento de fragmentos/píxeles 1624 transforma fragmentos o píxeles recibidos del rasterizador 1622, según lo especificado por los programas de sombreado de píxeles o fragmentos. Por ejemplo, la unidad de procesamiento de fragmentos/píxeles 1624 se puede programar para llevar a cabo operaciones que incluyen, entre otros, asignación de textura, sombreado, mezcla, corrección de textura y corrección de perspectiva para producir fragmentos o píxeles sombreados que se emiten a una unidad de operaciones de trama 1626. La unidad de procesamiento de fragmentos/píxeles 1624 puede leer datos que se almacenan en cualquiera de la memoria del procesador en paralelo o la memoria de sistema para su uso cuando se procesan los datos de fragmento. Los programas de sombreado de fragmentos o de píxeles se pueden configurar para sombrear a granularidad de muestra, de píxel, de tesela u otras dependiendo de las tasas de muestreo configuradas para las unidades de procesamiento.

La unidad de operaciones de rasterización 1626 es una unidad de procesamiento que realiza operaciones de rasterización que incluyen, pero sin limitación, estarcido, prueba z, mezcla y similares, y emite datos de píxeles como datos de gráficos procesados para que se almacenen en la memoria de gráficos (por ejemplo, la memoria de procesador paralelo 1322 como en la **Figura 13**, y/o la memoria de sistema 1204 como en la Figura 12, para que se visualicen en el uno o más dispositivo o dispositivos de visualización 1210 o para su procesamiento adicional por uno del uno o más procesador o procesadores 1202 o procesador o procesadores paralelos 1212. En algunas realizaciones, la unidad de operaciones de trama 1626 está configurada para comprimir datos z o de color que se escriben en la memoria y descomprimir datos z o de color que se leen de la memoria.

Ejecución de operaciones de reducción sobre elementos vectoriales

Un vector puede contener múltiples elementos de datos que comparten las mismas características y pueden combinarse a través de operaciones de reducción. Por ejemplo, el vector puede contener elementos de datos con los mismos valores (duplicados). Para ciertos cálculos, un elemento de datos dentro de un conjunto de duplicados tiene una dependencia de datos que implica otro elemento de datos dentro del mismo conjunto de duplicados. El conjunto de duplicados necesita identificarse para permitir que el cálculo se vectorice cuando un procesador opera sobre el vector. Los duplicados pueden detectarse usando una instrucción de detección de conflicto tal como VPCONFLICT.

En particular, la instrucción puede probar la igualdad de cada elemento de su entrada de registro vectorial con todos los elementos anteriores de esa entrada (por ejemplo, todos los elementos más cercanos al bit menos significativo (LSB)), y emite los resultados de estas comparaciones como un conjunto de vectores de bits. La instrucción de detección de conflicto proporciona una manera de determinar si un elemento tiene una dependencia de datos que implica a otros elementos dentro del mismo registro (por ejemplo, un registro de SIMD). La capacidad de identificar valores de duplicados dentro de elementos separados del registro permite que se vectorice código escalar en los casos en los que las posibles dependencias de datos a través de los elementos de registro podrían impedir de otro modo la vectorización.

Las arquitecturas de conjunto de instrucciones (ISA) anteriores usan una pluralidad de instrucciones para realizar operaciones de reducción en múltiples etapas. Por ejemplo, una instrucción puede detectar los duplicados dentro de un vector (por ejemplo, determinando un subconjunto de elementos con índices singulares), y la detección puede realizarse en forma de SIMD. A continuación, otra instrucción puede realizar operaciones de reducción sobre los duplicados con el mismo índice.

Sin embargo, es deseable que ciertas operaciones sean capaces de implementar una instrucción única para operaciones de reducción sobre uno o más vectores almacenados en uno o más registros (u otras unidades de almacenamiento), de modo que una vez que se decodifica la instrucción única, hace que un procesador realice las operaciones de reducción requeridas sin instrucciones adicionales. La instrucción única para realizar las operaciones de reducción da como resultado una mejor compilación y eficiencia computacional.

Ejecución de operaciones de reducción sobre elementos vectoriales en un registro

Las realizaciones de la invención incluyen técnicas para realizar operaciones de reducción sobre elementos vectoriales en un registro usando una instrucción única. Tras ejecutar la instrucción única, las operaciones de reducción pueden realizarse usando múltiples reducciones de árbol en paralelo, una reducción por valor de índice singular, sobre el elemento dentro de un registro (por ejemplo, un registro de SIMD). La instrucción única puede combinar las operaciones realizadas por múltiples operaciones anteriores. Por ejemplo, la instrucción única puede ordenar cada uno de los elementos que comparten un índice (previamente realizado por instrucciones tales como VPOPCNT). La instrucción única también puede identificar y combinar iterativamente resultados parciales de pares de elementos de datos dentro del mismo índice (realizado anteriormente por instrucciones tales como VPCONFLICT).

Dentro de un registro, puede haber múltiples valores que es necesario combinar a través de uno o más patrones de reducción. Por ejemplo, una aplicación puede tener un conjunto de valores {a0, b0, a1, a2, b1, a3, a4, b2} dentro de un único registro que es necesario combinar para que se sumen todos los valores 'a' y se sumen todos los valores de 'b', dando solo dos valores {a0+a1+a2+a3+a4, b0+b1+b2}. Dada una operación de reducción con solo dos entradas (por ejemplo, a través de un 'añadir' en un procesador), una forma eficiente de sumar todos los valores de 'a' y 'b' es realizar múltiples reducciones de árbol binario en paralelo a través de los elementos del registro.

Algunas realizaciones de la invención generan (1) una primera salida que identifica las reducciones independientes y (2) una segunda salida para identificar el orden (por ejemplo, hijos izquierdos frente a derechos) de los árboles de reducción binarios. El primero puede usarse para paralelizar las operaciones de reducción para las reducciones independientes y el último para contar el número de instancias de cada valor de índice singular dentro de un vector de índices asociado.

La **Figura 17** muestra operaciones de reducción de árbol sobre una pluralidad de elementos vectoriales almacenados en un registro de entradas tras ejecutar una instrucción única según una realización de la invención. La instrucción única toma el formato de "vmatchindx zmm0{k0}, zmm1, zmm2" como se muestra en la referencia 1700, aunque pueden implementarse muchas variedades de formatos de instrucción, donde uno o más operandos y/o valores inmediatos, junto con código o códigos de operación como se ha analizado anteriormente en el presente documento, pueden especificarse para las operaciones de reducción según realizaciones de la invención. En algunas realizaciones, la instrucción única puede emitirse como una instrucción de única instrucción de múltiples datos (SIMD) y, adicionalmente/alternativamente, la instrucción única puede emitirse como una instrucción de instrucción única de múltiples hilos (SMT) como se ha analizado en el presente documento anteriormente en relación con la **Figura 13**.

Las operaciones de reducción toman valores de un registro de entradas (zmm2) 1710. El registro de entradas 1710 almacena valores de un conjunto de elementos de datos d_0 a d_{15} , y el conjunto de elementos de datos puede pertenecer a un vector/disposición ordenada/matriz para el que pueden aplicarse las operaciones de reducción. El registro de entradas puede ser uno de los registros vectoriales 514 en la **Figura 5** en una realización. Cada valor dentro del registro de entradas 1710 puede considerarse que está dentro de un carril para ejecutarse usando técnicas de SIMD. El registro de entradas 1710 también puede estar dentro del archivo de registro 1358 de la **Figura 13D**, donde los elementos de datos en el registro de entradas pueden vectorizarse para su ejecución por los grupos de hilos (por ejemplo, las urdimbres).

Las operaciones de reducción también toman valores de un registro de índices (zmm1) 1720. Los valores de índice indican qué valores dentro del registro de entradas están asociados entre sí. Si dos elementos de datos dentro del registro de entradas corresponden al mismo valor de índice, estos están implicados en la misma reducción de árbol; de lo contrario, los mismos están implicados en diferentes reducciones de árbol. Por ejemplo, d_0 , d_3 , d_8 , d_{14} , y d_{15} tienen el mismo valor de índice 'A'; d_1 , de d_4 a d_6 , de d_9 a d_{11} y d_{13} tienen el mismo valor de índice 'B'; y d_2 y d_7 tienen el mismo valor de índice 'D'. Estos elementos de datos con el mismo valor de índice están implicados en reducciones de árbol respectivas.

Las operaciones de reducción se muestran en la referencia 1730, y pueden ser una o más de obtener la suma (a través de "sumar"), el mínimo, el máximo, la media, la mediana, el producto, la diferencia (a través de "restar"), el recuento de ocurrencias de elementos de datos asociados basándose en los valores de índice. Las operaciones de reducción pueden ser una o más de: acumulación de los valores de elementos de datos asociados (mediante suma, resta o multiplicación), la selección de un valor máximo o un valor mínimo de los valores de elementos de datos asociados, y cálculo de un valor medio o mediano de los valores de elementos de datos asociados. La acumulación (específicamente, obtener la suma) se usa como un ejemplo para el siguiente análisis.

Las operaciones de reducción empiezan con la condición inicial 1732 de los valores de registro de entradas. Puede realizarse un conjunto de iteraciones sobre los elementos de datos asociados basándose en los valores de índice. Por ejemplo, d_0 , d_3 , d_8 , d_{14} y d_{15} tienen el mismo valor de índice 'A' y se muestra que estos valores se reducen a través de tres iteraciones. Para la primera iteración en la referencia 1734, los pares de elementos de datos con el valor 'A' (contando desde la posición de bit menos significativa en esta realización) se suman entre sí para llegar a los resultados intermedios 1734. El valor en el elemento de datos con número impar seguirá siendo el mismo (por ejemplo, el valor de d_{15} en esta realización). Los otros elementos de datos con los mismos valores de índice 'B' y 'D' irán a la misma iteración respectivamente. En la segunda iteración, los pares de resultados intermedios 1734 se suman entre sí de nuevo para llegar a los resultados intermedios 1736. El proceso continúa para alcanzar un único valor para cada índice singular como el resultado en los resultados finales 1738.

Los resultados de las operaciones de reducción se almacenan en la referencia 1760 dentro del registro de salidas (zmm0) 1750. El registro de salidas 1750 puede almacenar los resultados finales 1738, cada uno para un valor de índice singular en las posiciones de elemento de datos asociadas de zmm0. Por ejemplo, el resultado final para el valor de índice 'A', la suma (para 'A') = $d_0 + d_3 + d_8 + d_{14} + d_{15}$, puede almacenarse en todas las posiciones de bit asociadas en zmm0 en R_0 , R_3 , R_8 , R_{14} y R_{15} . Como alternativa, el registro de salidas puede almacenar los resultados finales para los valores de índice singulares solo en la posición de elemento de datos más a la izquierda (o más a la derecha, u otra predefinida) de zmm0 asociada con los valores de índice singulares, por ejemplo, R_{15} para el valor de

índice 'A'. Además, el registro de salidas 1750 puede almacenar como alternativa los resultados que incluyen los resultados intermedios en las posiciones de elemento de datos asociadas de zmm0. Por ejemplo, R_0 puede almacenar d_0 , R_3 almacena $d_0 + d_3$, R_8 almacena d_8 , R_{14} almacena $d_0 + d_3 + d_8 + d_{14}$, and R_{15} almacena $d_0 + d_3 + d_8 + d_{14} + d_{15}$.

5 Obsérvese que, aunque las operaciones de reducción de árbol usan árboles binarios como ejemplo, también pueden implementarse otros tipos de reducción de árbol que incluyen más elementos de datos por iteración. Adicionalmente, aunque el emparejamiento dentro de la reducción de árbol es desde el emparejamiento más a la izquierda hacia la derecha, puede implementarse lo contrario en realizaciones alternativas.

10 Las operaciones de reducción pueden tomar adicionalmente valores de un vector de máscaras (k0) en algunas realizaciones. Los valores pueden tomarse de un registro de máscara u otra ubicación de almacenamiento (por ejemplo, caché o cualquiera de los niveles de jerarquía de memoria). Puede establecerse un valor de máscara para los elementos de datos de entrada o los de salida del registro de entradas 1710 o el registro de salidas 1750. Cuando el vector de máscaras es para el registro de entradas y se borra un valor de máscara (por ejemplo, ajuste de valor para que sea '0'), el elemento de datos correspondiente en el registro de entradas se ignora en la operación de reducción. Como alternativa, el vector de máscaras puede establecerse para el registro de salidas y cuando se borra un valor de máscara, el elemento de datos correspondiente en el registro de salidas puede establecerse a cero ("enmascaramiento de puesta a cero") o al valor original antes de las operaciones de reducción ("enmascaramiento de fusión"). Como alternativa, el valor de máscara puede establecerse para que sea '1' para ignorar el elemento de datos correspondiente en el registro de entradas y/o el registro de salidas en una realización.

25 Usando una reducción de árbol, la reducción itera, como máximo, $\log_2 N$ veces (siendo N el número de elementos de datos, la longitud del vector). Por ejemplo, cuando todos los 16 elementos de datos del registro de entradas 1710 tienen el mismo valor de índice, la operación de reducción itera $\log_2 N = 4$ veces. Esto es ventajoso frente a enfoques anteriores, donde la operación de reducción iteraría $N-1$ veces, una para cada reducción serializada.

Adicionalmente, usando una instrucción única para realizar las operaciones de reducción, el conjunto de instrucciones puede ser significativamente más eficiente. La instrucción en sí puede especificar una operación de reducción particular que va a realizarse (por ejemplo, una o más de obtener la suma, el mínimo, el máximo, la media, la mediana, el producto, el recuento de ocurrencias de elementos de datos asociados), con valores de máscara específicos aplicados usando el vector de máscaras, y los resultados intermedios pueden almacenarse en el registro de salidas para operaciones futuras. Basándose en los valores en el registro de índices, las operaciones de reducción pueden aplicarse a cualquier elemento de datos de un vector de entradas cargado en el registro de entradas, y la reducción puede realizarse en paralelo usando técnicas de SIMD y/o SIMT, mejorando de ese modo significativamente la eficiencia de cálculo de los procesadores (por ejemplo, CPU, GPU u otras unidades de procesador).

Ejecución de operaciones de reducción sobre elementos vectoriales en múltiples registros

40 Las operaciones de reducción pueden extenderse a valores de vector almacenados en múltiples registros. La **Figura 18** muestra operaciones de reducción sobre una pluralidad de elementos vectoriales almacenados en múltiples registros de entradas tras ejecutar una instrucción única según una realización de la invención. La **Figura 18** es similar a la **Figura 17** con algunos detalles omitidos para centrarse en los detalles adicionales para la implementación de múltiples registros de entradas, y las mismas referencias o similares indican elementos o componentes que tienen las mismas funcionalidades o similares.

45 En una realización, la instrucción única toma el formato de "vmatchindx2 zmm0{k0}, zmm1{k1}, zmm2, zmm3, zmm4, zmm5" como se muestra en la referencia 1800. Las operaciones de reducción toman valores de un registro de entradas (zmm4) 1810 y un registro de entradas (zmm5) 1812, que almacena valores de un conjunto de elementos de datos que pertenecen a uno o más vectores para los que pueden aplicarse las operaciones de reducción. Cada registro de entradas 1810/1812 puede operar de manera similar al registro de entradas 1710 en una realización. Las operaciones de reducción también toman valores de un registro de índices (zmm2) 1820 y un registro de índices (zmm3) 1822. Cada registro de índices puede operar de manera similar al registro de índices 1720, y si dos elementos de datos dentro de los registros de entradas corresponden a los mismos valores de índice, estos están implicados en la misma reducción de árbol. Las operaciones de reducción se muestran en la referencia 1830, las operaciones de la cual son similares a la analizada en el presente documento anteriormente en relación con la referencia 1730.

50 Los resultados de las operaciones de reducción se almacenan en la referencia 1860 dentro de un registro de salidas (zmm0) 1850 y un registro de salidas (zmm1) 1852. El almacenamiento de los resultados es similar a las operaciones en la referencia 1760 como se ha analizado anteriormente en el presente documento.

60 Obsérvese que pueden implementarse muchas variedades de formatos de instrucción para las operaciones de reducción sobre valores de vector almacenados en múltiples registros. En una realización alternativa, la instrucción única toma el formato de "vmatchindx2 zmm0{k0}, registro 1, registro 2", donde el registro 1 corresponde a zmm2 y el registro 2 corresponde a zmm4 en la **Figura 18**. De esta manera, la instrucción única especifica solo una ubicación de registro para cada tipo de registro, aunque el procesador operará sobre dos registros para cada tipo de registro. Por ejemplo, mientras que zmm0 puede especificarse como el registro de salidas (por ejemplo, con el número de registro

"10"), el procesador también operará sobre el registro de salidas inmediatamente siguiente, zmm1 (por ejemplo, con el número de registro "11"). Por lo tanto, zmm1 es un registro de salidas que no se indica explícitamente en la instrucción única como se muestra en la referencia 1882. El registro indicado explícitamente y el registro indicado implícitamente pueden emparejarse para ser pares en número par/impar, de modo que, si uno de un par de registros se especifica en la instrucción, se operará sobre los valores del otro registro en el par.

De manera similar, zmm4 puede especificarse como el registro de entradas, zmm2 como el registro de índices, y k0 como el vector de máscaras en la instrucción única; sin embargo, el procesador también tomará valores de entrada de zmm5, valores de índice de zmm3 y valores de máscara del vector de máscaras k1, respectivamente. Como se muestra en la referencia 1880, estos valores no se indican explícitamente en la instrucción única.

Las operaciones de reducción sobre elementos vectoriales no se limitan a valores dentro de dos registros de entradas, y puede operarse sobre valores en registros adicionales mediante una instrucción única. Las operaciones de reducción para valores de vector sobre múltiples registros pueden realizarse de manera similar a aquellas sobre dos registros tales como los registros de entradas 1810 y 1812 analizados en el presente documento.

Ejecución de operaciones de reducción sobre elementos vectoriales en múltiples registros de diferentes unidades informáticas

Un archivo de registro de un multiprocesador puede dividirse entre diferentes unidades informáticas. Por ejemplo, el archivo de registro puede dividirse entre las diferentes unidades informáticas dentro de un multiprocesador para urdimbres o hilos (por ejemplo, el archivo de registro 1358 puede dividirse entre las diferentes urdimbres que son ejecutadas por el multiprocesador de gráficos 1334 como se ha analizado anteriormente en el presente documento). Para elementos vectoriales almacenados en múltiples registros de diferentes unidades informáticas, la instrucción única puede especificar explícitamente cada operando (por ejemplo, registro(s) de entradas, registro(s) de índices, vector(es) de máscaras y/o registro(s) de salidas) como se analiza en el presente documento. La instrucción puede identificar explícitamente todos los operandos (por ejemplo, registros y vector de máscaras) cuando el software/multiprocesador conoce el conjunto completo de registros disponibles para todos los hilos/urdimbres (por ejemplo, un multiprocesador de gráficos lo conoce habitualmente).

Cuando el software/multiprocesador no tiene el conocimiento, pueden implementarse realizaciones alternativas. Por ejemplo, la instrucción única puede tomar el formato de "vmatchindx2 zmmo{k0}, zmm1, zmm2, reg/imm", donde el operando "reg/imm" especifica un registro (u otra ubicación de almacenamiento tal como una caché/memoria en la jerarquía de memoria) o un valor inmediato que identifica valores para un segundo hilo o urdimbre. El software/multiprocesador tiene conocimiento de los operandos anteriores (zmm0, zmm1, zmm2, k0) para un primer hilo o urdimbre, y "reg/imm" puede contener el identificador del segundo hilo o urdimbre, para el que los otros registros de entradas/índices/salidas y el vector de máscaras mantienen valores para las operaciones de reducción. Como alternativa, "reg/imm" puede especificar un desplazamiento con respecto al primer hilo o urdimbre. Por ejemplo, si un hilo X ejecuta la instrucción única, la instrucción única puede ejecutarse con la información combinada para el hilo X + reg/imm.

Además, los registros para la instrucción única pueden especificarse usando números de registro virtual para los registros, y el software/multiprocesador traduce los números de registro virtual para cada hilo o urdimbre a números de registro físicos. Los multiprocesadores hoy en día tienen una noción tanto de los registros lógicos/arquitectónicos como de los registros físicos. Si la instrucción única especifica un número de registro de salidas "10", un multiprocesador puede identificar el registro físico "10" para cada hilo/urdimbre. Tal traducción es sencilla para algunos multiprocesadores (por ejemplo, CPU). Para otros multiprocesadores (por ejemplo, GPU), un multiprocesador sabe que el número total de registros que necesita un hilo/urdimbre y asigna un fragmento contiguo de registros para cada hilo/urdimbre en consecuencia. Por ejemplo, el multiprocesador puede asignar los registros 0 a N-1 al hilo/urdimbre 0, N a 2N-1 a la urdimbre 1, y así sucesivamente. Cada hilo/urdimbre tiene un punto de inicio en el archivo de registro. El multiprocesador puede encontrar el registro físico que corresponde a un registro virtual especificado para una urdimbre añadiendo el desplazamiento para el hilo/urdimbre (por ejemplo, número de hilo/urdimbre * desplazamiento + registro virtual especificado). Por lo tanto, la instrucción única puede usarse para operaciones de reducción a través de valores de entrada en múltiples registros de entradas, incluso cuando los múltiples registros de entradas no se identifican explícitamente en la instrucción única.

Cuando las operaciones de reducción se realizan usando múltiples unidades informáticas dentro de un multiprocesador para urdimbres o hilos, puede ser necesaria la sincronización entre las múltiples unidades informáticas. La sincronización puede lograrse usando barreras para garantizar que las urdimbres e hilos tengan su información lista para combinarse. Por ejemplo, puede usarse una instrucción tal como DMB (barrera de memoria de datos), DSB (barrera de sincronización de datos) o ISB (barrera de sincronización de instrucciones) en algunas realizaciones.

Como alternativa, la instrucción única puede implementar sincronización sin usar ninguna instrucción adicional. Por ejemplo, la instrucción única puede tomar el formato de "vmatchindx2 zmmo{k0}, zmm1, zmm2, sync_reg/imm", donde el operando "sync_reg/imm" especifica una ubicación (un registro u otra ubicación de almacenamiento tal como una

caché/memoria en la jerarquía de memoria) que las urdimbres/hilos usarán para sincronizarse. Suponiendo que dos urdimbres/hilos están implicados en las operaciones de reducción. Los dos hilos/urdimbres ejecutarán la instrucción única, y tendrán el mismo valor para "sync_reg/imm". La ubicación especificada se inicializará a un valor inicial, antes de que las urdimbres/hilos implicados ejecuten la instrucción. Cuando el primer hilo/urdimbre ejecuta la instrucción, puede comprobar la ubicación especificada. Cuando ve que la ubicación especificada contiene el valor inicial, sobrescribe la ubicación con otro valor, indicando que ha llegado, y que está esperando la segunda urdimbre/hilo. A continuación, cuando el segundo hilo/urdimbre ejecuta la instrucción, este comprueba de nuevo la ubicación especificada. Cuando determina que la ubicación especificada no contiene el valor inicial, la determinación desencadena la ejecución de la operación de combinación a través de las diferentes unidades informáticas. Cuando se completa la operación de combinación, se permite que procedan ambos hilos/urdimbres.

Cuando más de dos urdimbres/hilos están implicados en las operaciones de reducción, un valor en la ubicación especificada puede indicar a una urdimbre/hilo que la urdimbre/hilo es la última en llegar, y el valor desencadena la ejecución de la operación de combinación a través de las diferentes unidades informáticas, y a todos los hilos/urdimbres se les permite proceder cuando se completa la operación de combinación.

La **Figura 19** muestra operaciones de reducción sobre una pluralidad de elementos vectoriales almacenados en múltiples registros de entradas tras usar sincronización ejecutando una instrucción única según una realización de la invención. La **Figura 19** es similar a la **Figura 18** con algunos detalles omitidos para centrarse en los detalles adicionales para la implementación de múltiples registros de entradas, y las mismas referencias o similares indican elementos o componentes que tienen las mismas funcionalidades o similares.

Una porción de los elementos vectoriales es ejecutada por un primer conjunto de unidades informáticas en la referencia 1970, mientras que otra porción de los elementos vectoriales es ejecutada por otros conjuntos de unidades informáticas como se muestra en la referencia 1972. Diferentes conjuntos de unidades informáticas pueden usar diferentes partes de un archivo de registro de un multiprocesador (por ejemplo, una CPU/GPU). Adicionalmente, los vectores de máscaras 1940 y 1942 opcionales usados para ejecutar la instrucción única para operaciones de reducción pueden almacenarse usando también la parte diferente del archivo de registro.

La instrucción única (por ejemplo, "vmatchindx2 zmm0{k0}, zmm1, zmm2, sync_reg/imm" como se ha analizado anteriormente) puede especificar una ubicación de sincronización 1900, que indica a las unidades informáticas que están implicadas en las operaciones de reducción si se logra la sincronización entre las unidades informáticas implicadas. Una vez que se logra la sincronización, la operación de combinación puede realizarse en la referencia 1934, donde pueden completarse los resultados combinados de las unidades informáticas. Una vez que se completa la combinación, los resultados pueden almacenarse en los registros de salidas 1950 y 1952 como se muestra en la referencia 1960.

A través del uso de la indicación de sincronización en la instrucción única, las realizaciones de la invención no requieren instrucciones adicionales para lograr una reducción sincronizada alrededor de elementos vectoriales almacenados en múltiples registros en diferentes porciones de un archivo de registro. La instrucción única es, por lo tanto, más eficiente que múltiples instrucciones para lograr la sincronización.

Algunas realizaciones de la invención

La **Figura 20** es un diagrama de flujo que muestra operaciones de reducción sobre una pluralidad de elementos vectoriales tras ejecutar una instrucción única según una realización de la invención. El método 2000 puede realizarse en un procesador, que contiene circuitería de decodificación y circuitería de ejecución. En la referencia 2002, la circuitería de decodificación decodifica una instrucción, donde la instrucción especifica un primer registro de entradas que contiene una pluralidad de valores de elementos de datos, un primer registro de índices que contiene una pluralidad de índices, y un registro de salida. Cada índice de la pluralidad de índices se mapea a una posición de elemento de datos singular del primer registro de entradas. Aunque en el presente documento se dan varios ejemplos de instrucción en relación con las **Figuras 17-19**, pueden implementarse muchas variedades de formatos de instrucción, donde uno o más operandos y/o valores inmediatos, junto con código o códigos de operación como se ha analizado anteriormente en el presente documento, pueden especificarse para las operaciones de reducción según realizaciones de la invención.

La instrucción es para realizar operaciones de reducción sobre la pluralidad de elementos vectoriales, que pueden pertenecer a una disposición ordenada/matriz. El código de operación (código de op.) de la instrucción especifica una o más operaciones de reducción. Las operaciones de reducción pueden ser una o más de: acumulación de los valores de elementos de datos asociados, la selección de un valor máximo o un valor mínimo de los valores de elementos de datos asociados, y cálculo de un valor medio o mediano de los valores de elementos de datos asociados. La acumulación (obtener la suma) se usa como un ejemplo para el siguiente análisis.

En algunas realizaciones, la instrucción especifica adicionalmente un vector de máscaras que contiene una pluralidad de máscaras, donde cada máscara indica que una posición de elemento de datos del registro de salidas está activa o inactiva, y donde los resultados no se escriben en la posición de elemento de datos que está inactiva. En una

realización alternativa, cada máscara indica que una posición de elemento de datos del registro de entradas está activa o inactiva, y los elementos de datos inactivos no están implicados en la operación de reducción provocada por la instrucción.

En la referencia 2004, el circuito de ejecución ejecuta la instrucción decodificada. La ejecución incluye identificar valores de elementos de datos que están asociados entre sí basándose en los índices en la referencia 2012. A continuación, en la referencia 2014, la circuitería de ejecución realiza una o más operaciones de reducción sobre los valores de elementos de datos asociados basándose en la identificación; y en la referencia 2016, los resultados de las una o más operaciones de reducción se almacenan en el registro de salidas.

En una realización, un grupo de valores de elementos de datos se asocia entre sí cuando el grupo de valores de elementos de datos tiene un mismo valor de índice. En una realización, realizar las una o más operaciones de reducción es, para el grupo de valores de elementos de datos que comparten el mismo valor de índice, combinar el grupo de valores de elementos de datos para generar una combinación aritmética como resultado. La combinación aritmética puede incluir obtener la suma (a través de "sumar"), la diferencia (a través de "restar"), el mínimo, el máximo, la media, la mediana, el producto, el recuento de ocurrencias de elementos de datos asociados basándose en los valores de índice.

En una realización, los resultados se almacenan en una pluralidad de posiciones de elemento de datos del registro de salidas, correspondiendo cada posición de elemento de datos a uno de valores de elementos de datos asociados correspondientes. En una realización, las una o más operaciones de reducción se realizan en una pluralidad de iteraciones sobre un grupo de valores de elementos de datos asociados, y los resultados intermedios de la pluralidad de iteraciones se almacenan en posiciones de elementos de datos correspondientes a unos de valores de elementos de datos asociados correspondientes implicados en la obtención de los resultados intermedios, como se explica en el presente documento en relación con la **Figura 17**.

En una realización, la ejecución de la instrucción decodificada se realiza adicionalmente sobre un segundo registro de entradas, y se usan otra pluralidad de valores de elementos de datos dentro del segundo registro de entradas y otra pluralidad de índices dentro de un segundo registro de índices en la ejecución de las una o más reducción operaciones. En una realización, la instrucción especifica adicionalmente un segundo registro de entradas que contiene otra pluralidad de valores de elementos de datos, y un segundo registro de índices que contiene otra pluralidad de índices, cada índice de la otra pluralidad de índices se mapea a una posición de elemento de datos singular del segundo registro de entradas, y las una o más operaciones de reducción se realizan sobre los valores de elementos de datos del primer y el segundo registros de entradas basándose en índices del primer y el segundo registros de índices. Estas realizaciones se analizan con más detalle en el presente documento en relación con la **Figura 18**.

En una realización, la instrucción es ejecutada por dos o más unidades informáticas. En una realización, cada una de las dos o más unidades informáticas es una urdimbre o un hilo. En una realización, las dos o más unidades informáticas están sincronizadas en la ejecución de las una o más operaciones de reducción. En una realización, la instrucción especifica adicionalmente una ubicación que almacena un valor que indica si las dos o más unidades informáticas están sincronizadas. En una realización, el procesador es una unidad de procesamiento de gráficos (GPU). Estas realizaciones se analizan con más detalle en el presente documento en relación con la **Figura 19**.

En la memoria descriptiva anterior, las realizaciones de la invención se han descrito con referencia a realizaciones ejemplares específicas de la misma. Sin embargo, será evidente que se pueden realizar diversas modificaciones y cambios a la misma sin apartarse del alcance más amplio de la invención como se expone en las reivindicaciones adjuntas. En consecuencia, la memoria descriptiva y los dibujos han de considerarse en un sentido ilustrativo más que restrictivo.

Las realizaciones de la invención pueden incluir diversas etapas, que se han descrito anteriormente. Los etapas se pueden materializar en instrucciones ejecutables por máquina que se pueden utilizar para que un procesador de propósito general o especial lleve a cabo las etapas. Como alternativa, estas etapas se pueden llevar a cabo por componentes de hardware específicos que contienen lógica cableada para llevar a cabo las etapas, o por cualquier combinación de componentes informáticos programados y componentes de hardware personalizados.

Como se describe en la presente memoria, las instrucciones pueden hacer referencia a configuraciones específicas de hardware, como por ejemplo circuitos integrados de aplicación específica (ASIC) configurados para llevar a cabo algunas operaciones o que tienen una funcionalidad predeterminada, o instrucciones de software almacenadas en la memoria incorporada en un medio legible por ordenador no transitorio. Por lo tanto, las técnicas mostradas en las figuras se pueden implementar utilizando código y datos almacenados y ejecutados en uno o más dispositivos electrónicos (por ejemplo, una estación final, un elemento de red, etc.). Dichos dispositivos electrónicos almacenan y comunican (internamente y/o con otros dispositivos electrónicos a través de una red) código y datos utilizando medios informáticos legibles por máquina, como por ejemplo medios de almacenamiento informáticos legibles por máquina no transitorios (por ejemplo, discos magnéticos; discos ópticos; memoria de acceso aleatorio; memoria de sólo lectura; dispositivos de memoria flash; memoria de cambio de fase) y medios de comunicaciones informáticos legibles por máquina transitorios (por ejemplo, señales eléctricas, ópticas, acústicas u otras formas de señales propagadas - como

5 por ejemplo ondas portadoras, señales infrarrojas, señales digitales, etc.). Además, dichos dispositivos electrónicos normalmente incluyen un conjunto de uno o más procesadores acoplados a uno o más componentes, como por ejemplo uno o más dispositivos de almacenamiento (medios de almacenamiento no transitorios legibles por máquina), dispositivos de entrada/salida de usuario (por ejemplo, un teclado, una pantalla táctil y/o una pantalla) y conexiones de red. El acoplamiento del conjunto de procesadores y otros componentes se realiza normalmente a través de uno o más buses y puentes (también denominados controladores de bus). El dispositivo de almacenamiento y las señales que transportan el tráfico de red representan, respectivamente, uno o más medios de almacenamiento legibles por máquina y medios de comunicaciones legibles por máquina. Por lo tanto, el dispositivo de almacenamiento de un dispositivo electrónico dado almacena normalmente código y/o datos para su ejecución en el conjunto de uno o más procesadores de dicho dispositivo electrónico. Por supuesto, una o más partes de una realización de la invención se pueden implementar utilizando diferentes combinaciones de software, firmware y/o hardware. A lo largo de esta descripción detallada, a efectos explicativos, se han expuesto numerosos detalles específicos con el fin de proporcionar una comprensión completa de la presente invención. Sin embargo, será evidente para un experto en la técnica que la invención se puede llevar a la práctica sin algunos de estos detalles específicos. En ciertas instancias, estructuras y funciones bien conocidas no se describieron con todo lujo de detalles para evitar complicar la materia objeto de la presente invención.

REIVINDICACIONES

1. Un procesador (600) para realizar operaciones correspondientes a una instrucción de única instrucción de múltiples hilos, SIMT, siendo la instrucción SIMT una instrucción única con un formato que ha de incluir un código de operación para indicar una operación de reducción y una pluralidad de operandos para indicar una máscara para la operación de reducción en un grupo de hilos y una pluralidad de registros que van a ser usados por la operación de reducción, incluyendo el procesador (600):
la pluralidad de registros; y
una pluralidad de unidades de computación para realizar respectivamente la instrucción SIMT para hilos correspondientes del grupo de hilos, para:
realizar la operación de reducción indicada sobre una pluralidad de elementos de datos, que van a almacenarse en la pluralidad de registros, que son indicados por la instrucción SIMT y corresponden respectivamente a la pluralidad de unidades de computación, para generar un elemento de datos de resultado que va a almacenarse en un registro de destino indicado por la instrucción SIMT, en donde la máscara almacenada en un registro de máscara ha de tener una pluralidad de primeros valores cada uno de los cuales es para indicar que un elemento de datos correspondiente de la pluralidad de elementos de datos ha de incluirse en la operación de reducción, y la máscara ha de tener cero o más segundos valores cada uno de los cuales es para indicar que un elemento de datos adicional correspondiente no ha de incluirse en la operación de reducción;
realizar una sincronización indicada por la instrucción SIMT para sincronizar los hilos del grupo de hilos haciendo que los hilos del grupo de hilos esperen hasta que todos los hilos del grupo de hilos hayan realizado la instrucción SIMT; y almacenar el elemento de datos de resultado en el registro de destino.
2. El procesador (600) de la reivindicación 1, en donde la instrucción SIMT ha de indicar la operación de reducción como una cualquiera de una pluralidad de operaciones de reducción soportadas por la instrucción SIMT, incluyendo la pluralidad de operaciones de reducción al menos una operación de reducción de adición, una operación de reducción máxima y una operación de reducción mínima.
3. El procesador (600) de la reivindicación 2, en donde:
 - la operación de reducción indicada es la operación de reducción de adición, y realizar la operación de reducción de adición incluye sumar la pluralidad de elementos de datos;
 - la operación de reducción indicada es la operación de reducción máxima, y realizar la operación de reducción máxima incluye seleccionar uno máximo de la pluralidad de elementos de datos; o
 - la operación de reducción indicada es la operación de reducción mínima, y realizar la operación de reducción mínima incluye seleccionar uno mínimo de la pluralidad de elementos de datos.
4. El procesador (600) de cualquiera de las reivindicaciones 1 a 3, en donde al menos algunas de la pluralidad de unidades de computación, para realizar la operación de reducción, han de realizar la instrucción SIMT en momentos diferentes.
5. El procesador (600) de cualquiera de las reivindicaciones 1 a 4, en donde la pluralidad de elementos de datos han de ser parte de una matriz.
6. El procesador (600) de cualquiera de las reivindicaciones 1 a 5, en donde los hilos del grupo de hilos comprenden hilos de una urdimbre.
7. El procesador (600) de cualquiera de las reivindicaciones 1 a 6, que comprende además un archivo de registro que incluye la pluralidad de registros.
8. El procesador de cualquiera de las reivindicaciones 1 a 7, que comprende uno de:
 - una pluralidad de núcleos de gráficos (602A-602N);
 - una pluralidad de núcleos heterogéneos (602A-602N);
 - una pluralidad de núcleos de gráficos heterogéneos (602A-602N).
9. El procesador (600) de cualquiera de las reivindicaciones 1 a 8, en donde el procesador (600) es parte de una unidad de procesamiento de gráficos, GPU.
10. El procesador (600) de cualquiera de las reivindicaciones 1 o 9, que comprende además un convertidor de instrucciones para convertir la instrucción SIMT en una o más instrucciones de un conjunto de instrucciones diferente ejecutable por el procesador.
11. Un aparato que comprende:
 - una pluralidad de controladores de memoria (614);
 - una memoria caché de nivel dos, L2, (606) acoplada a la pluralidad de controladores de memoria (614);
 - el procesador (600) de cualquiera de las reivindicaciones 1 a 10, el procesador (600) acoplado a la pluralidad de controladores de memoria (614) y acoplado a la memoria caché de L2 (606); y
 - una interfaz de interconexión (612) acoplada al procesador (600); y

una interfaz de controlador de bus (616) acoplada al procesador (600).

12. Un método para realizar operaciones correspondientes a una instrucción de única instrucción de múltiples hilos, SIMT, siendo la instrucción SIMT una instrucción única con un formato que ha de incluir un código de operación para indicar una operación de reducción y una pluralidad de operandos para indicar una máscara para la operación de reducción en un grupo de hilos y una pluralidad de registros que van a ser usados por la operación de reducción, incluyendo el método:

realizar respectivamente, por una pluralidad de unidades de computación de un procesador, la instrucción SIMT para hilos correspondientes de un grupo de hilos, para:

realizar la operación de reducción indicada sobre una pluralidad de elementos de datos, que van a almacenarse en la pluralidad de registros, que son indicados por la instrucción SIMT y corresponden respectivamente a la pluralidad de unidades de computación, para generar un elemento de datos de resultado que va a almacenarse en un registro de destino indicado por la instrucción SIMT, en donde la máscara almacenada en un registro de máscara ha de tener una pluralidad de primeros valores cada uno de los cuales es para indicar que un elemento de datos correspondiente de la pluralidad de elementos de datos ha de incluirse en la operación de reducción, y la máscara ha de tener cero o más segundos valores cada uno de los cuales es para indicar que un elemento de datos adicional correspondiente no ha de incluirse en la operación de reducción;

realizar una sincronización indicada por la instrucción SIMT para sincronizar los hilos del grupo de hilos haciendo que los hilos del grupo de hilos esperen hasta que todos los hilos del grupo de hilos hayan realizado la instrucción SIMT; y almacenar el elemento de datos de resultado en el registro de destino.

13. Uno o más medios de almacenamiento legibles por máquina que tienen almacenadas en los mismos instrucciones de programa informático ejecutables que, cuando son ejecutadas por uno o más procesadores, hacen que el uno o más procesadores realicen el método de la reivindicación 12.

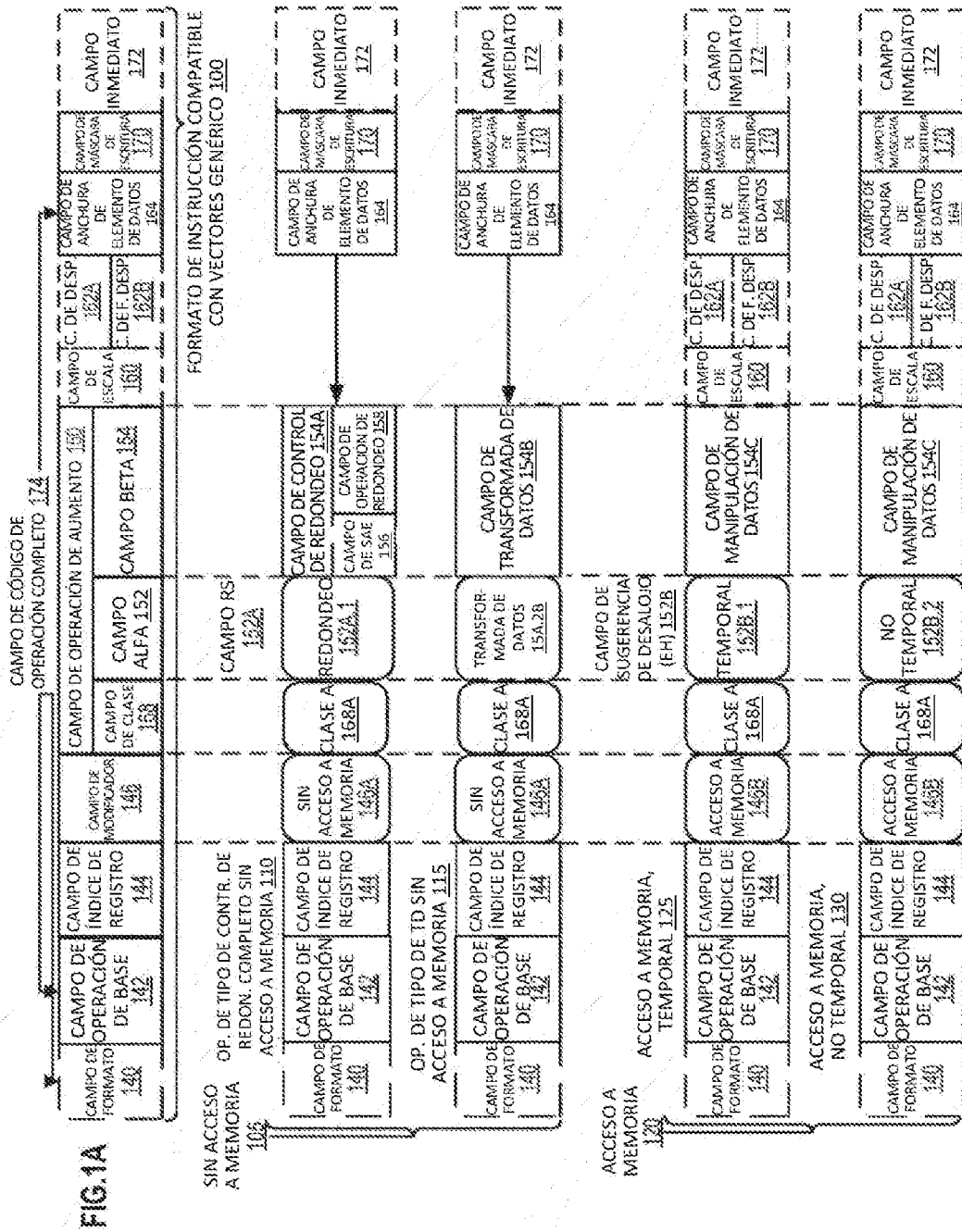
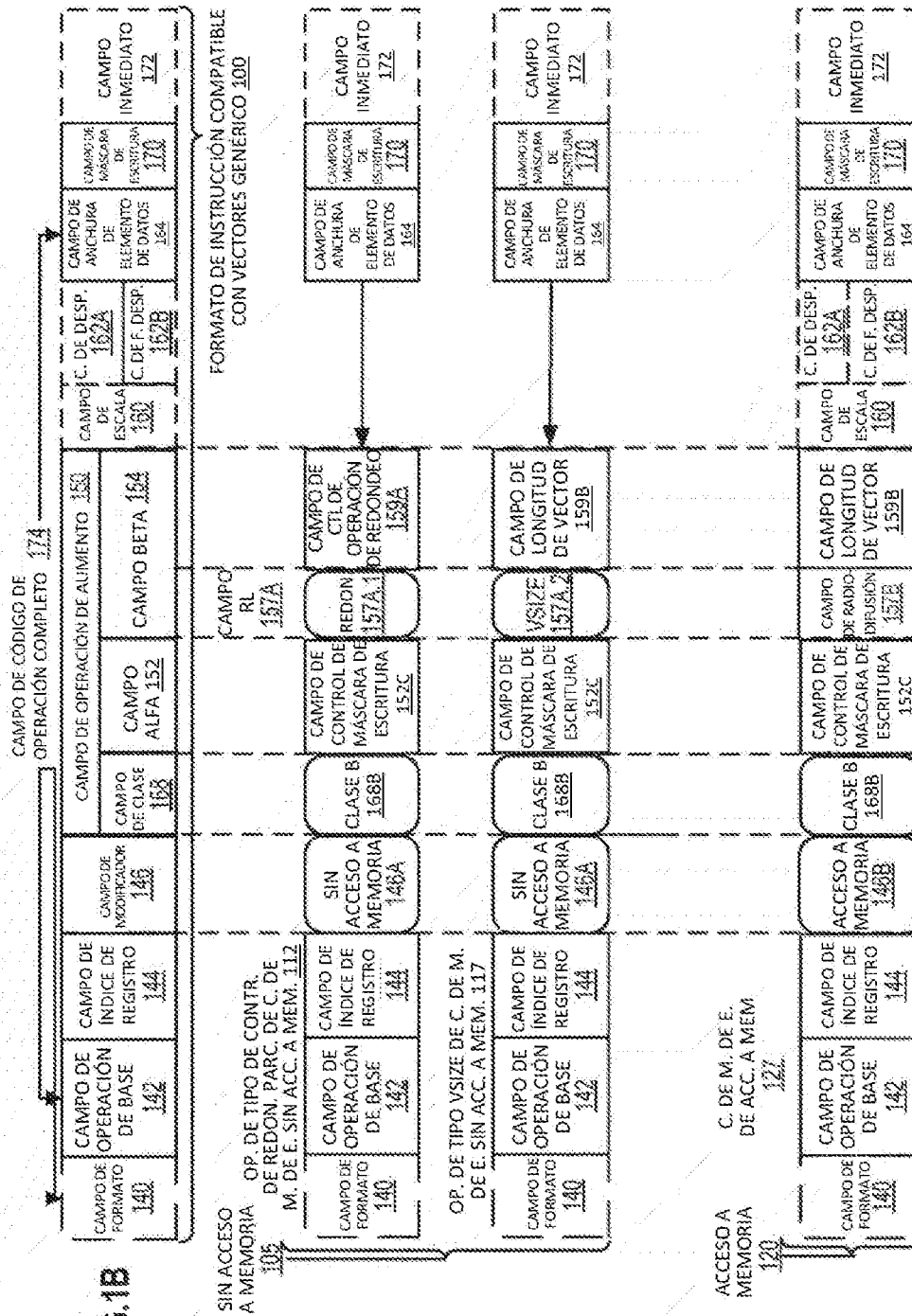


FIG.1B



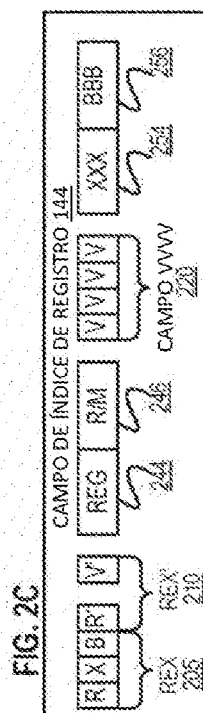
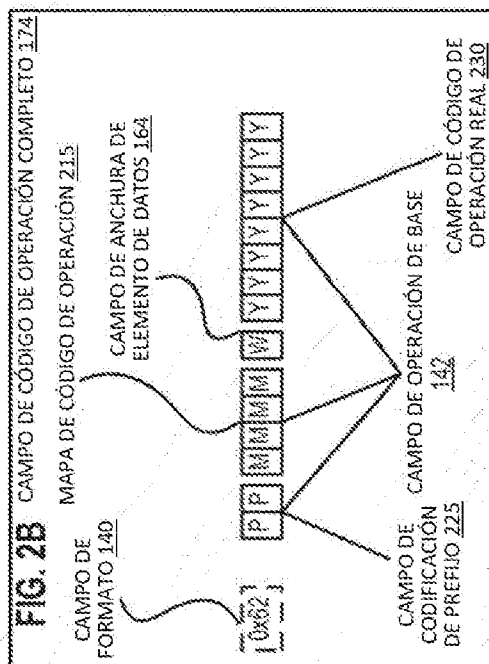
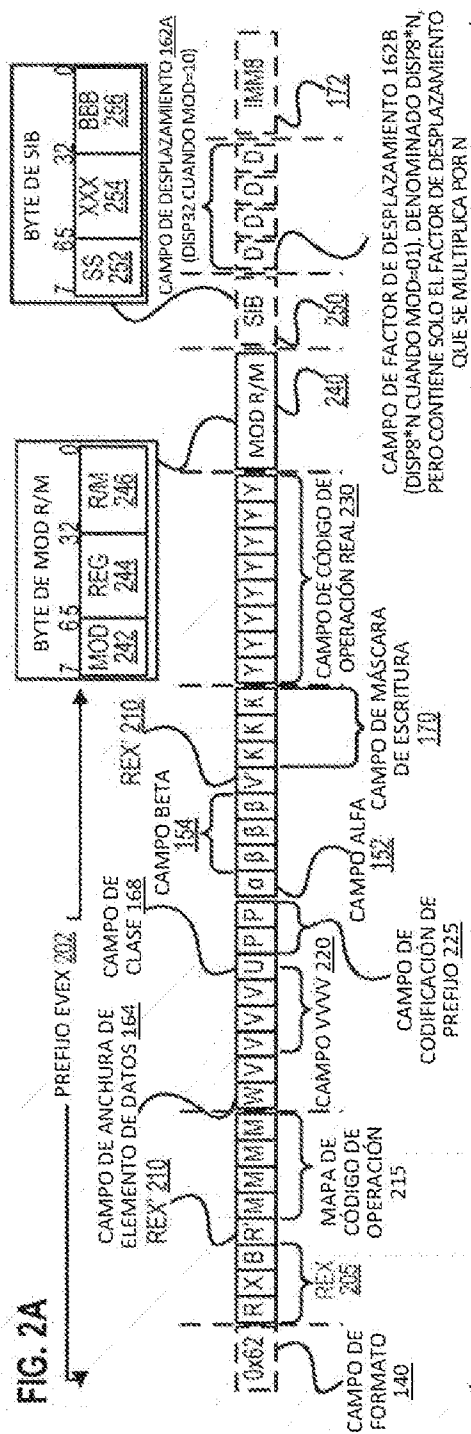
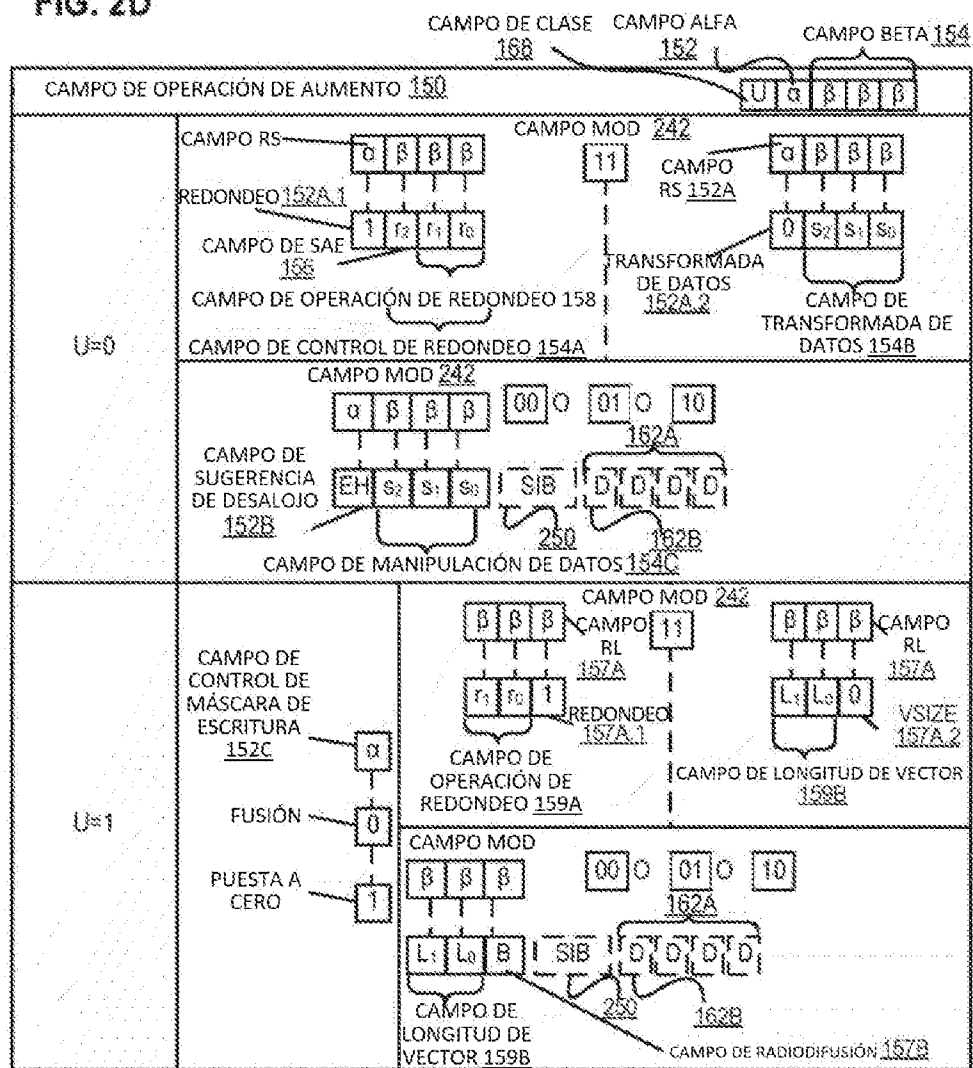
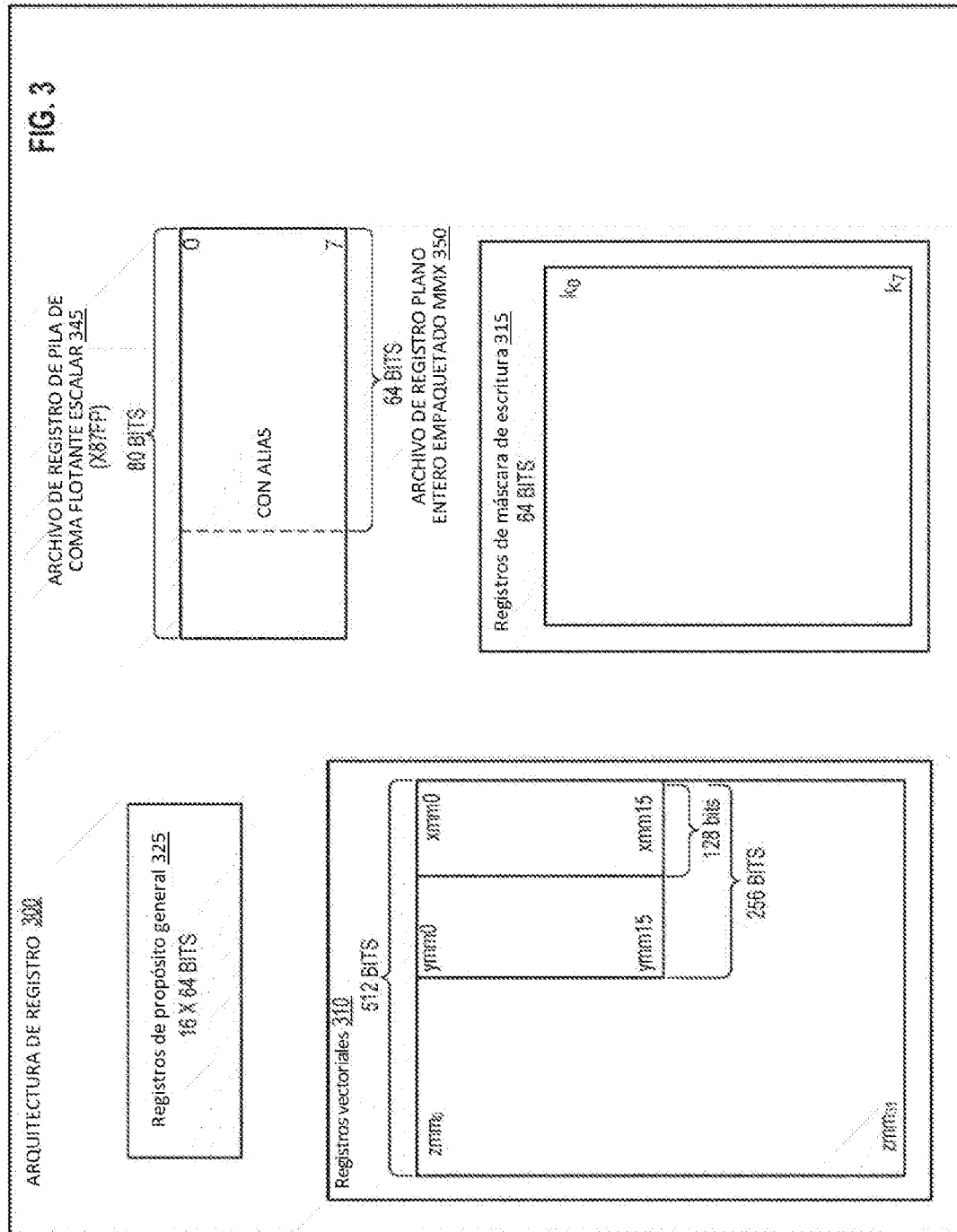


FIG. 2D





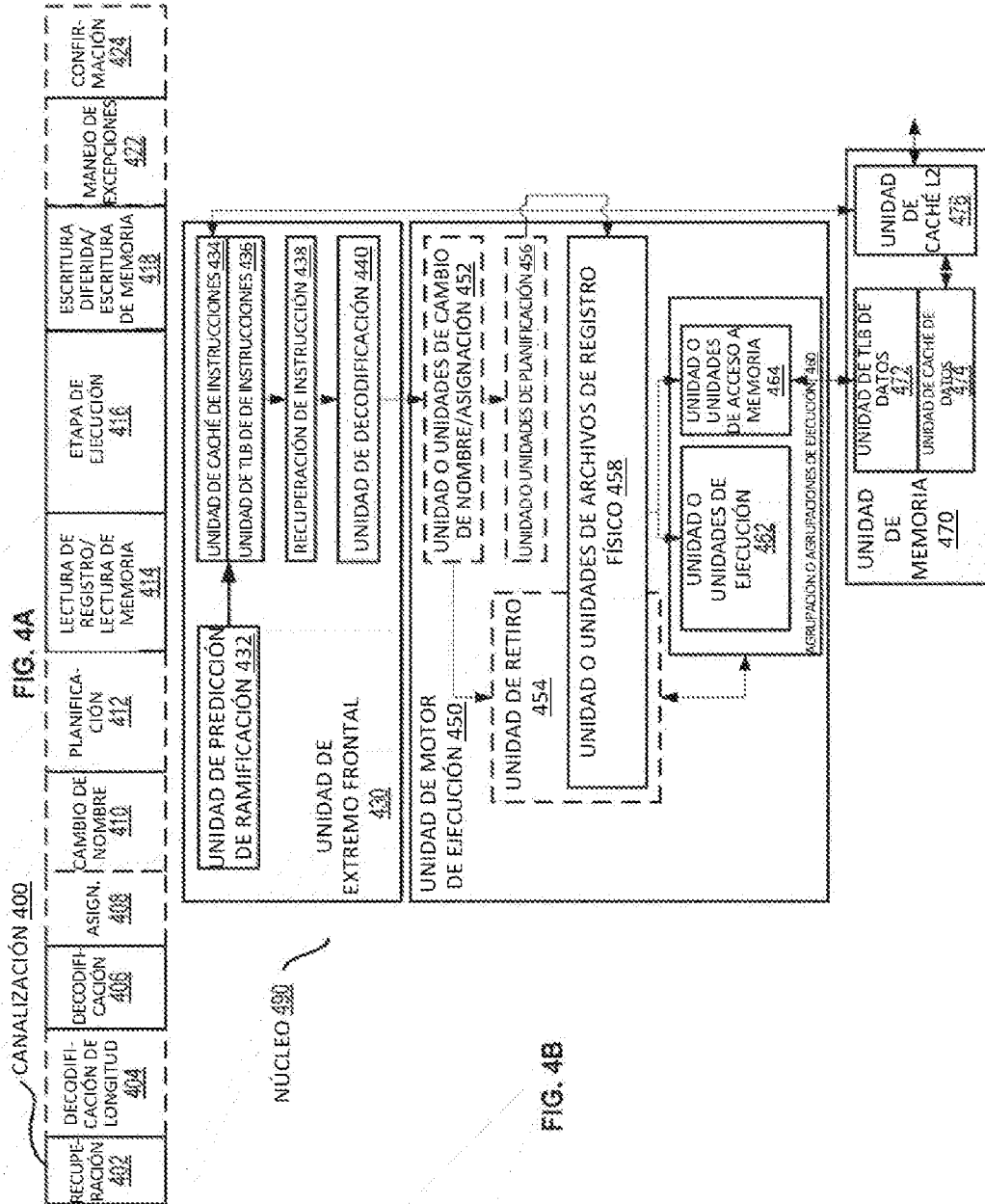


FIG. 5A

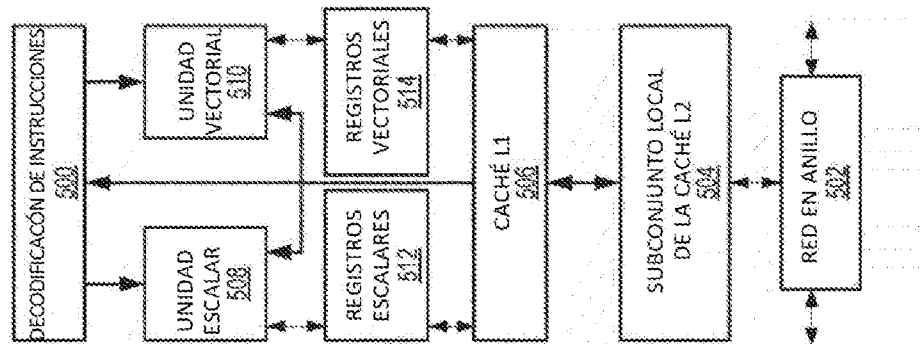
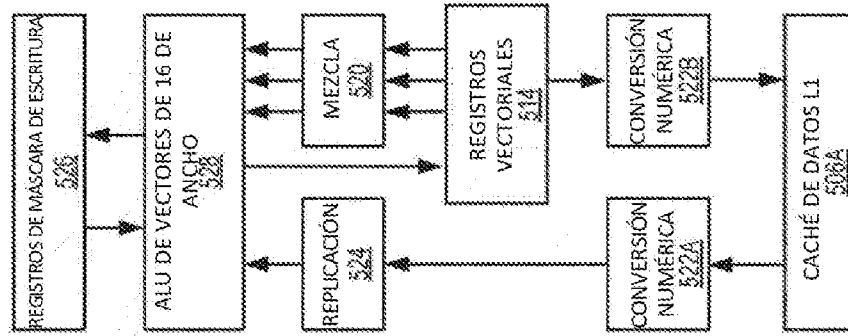


FIG. 5B



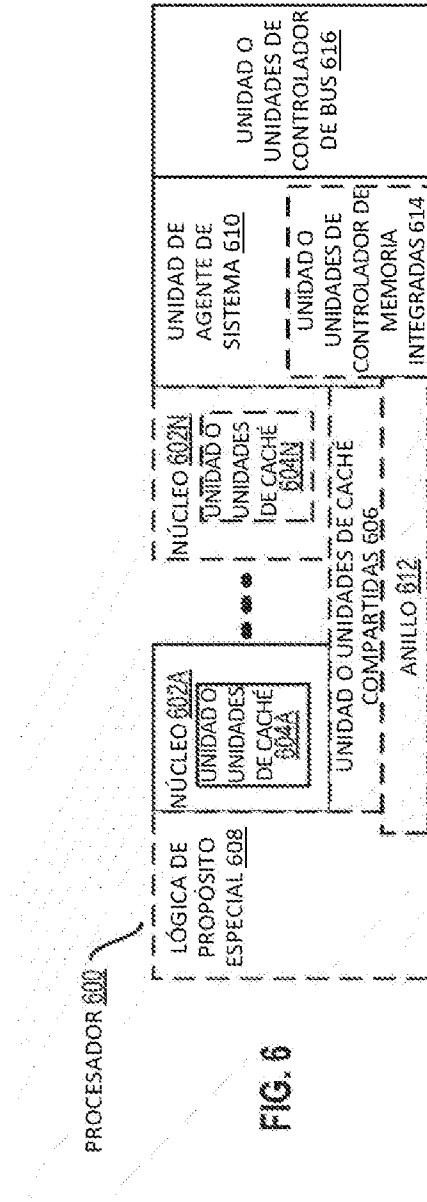


FIG. 6

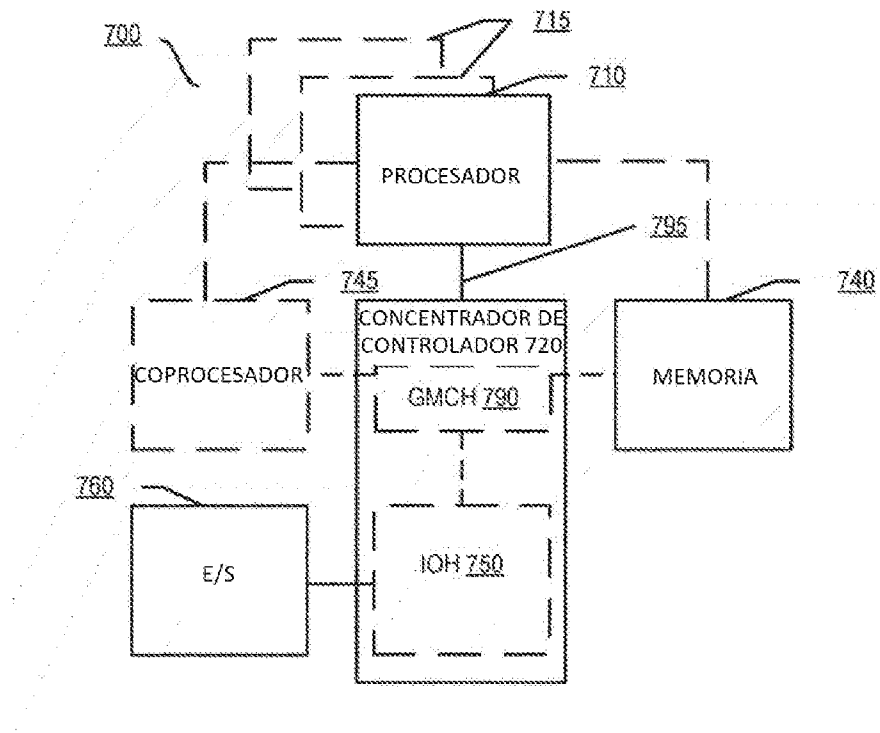


FIG. 7

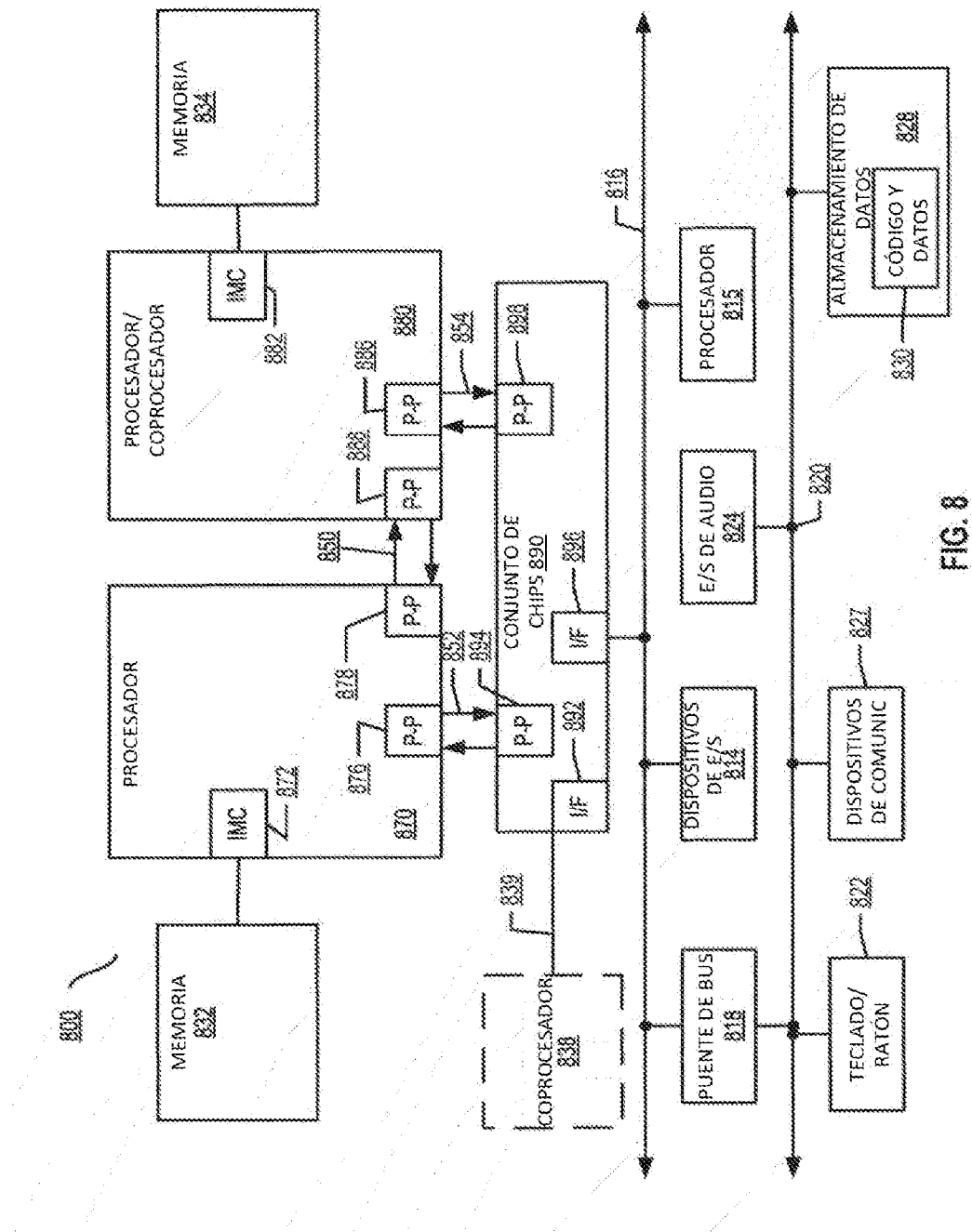


FIG. 8

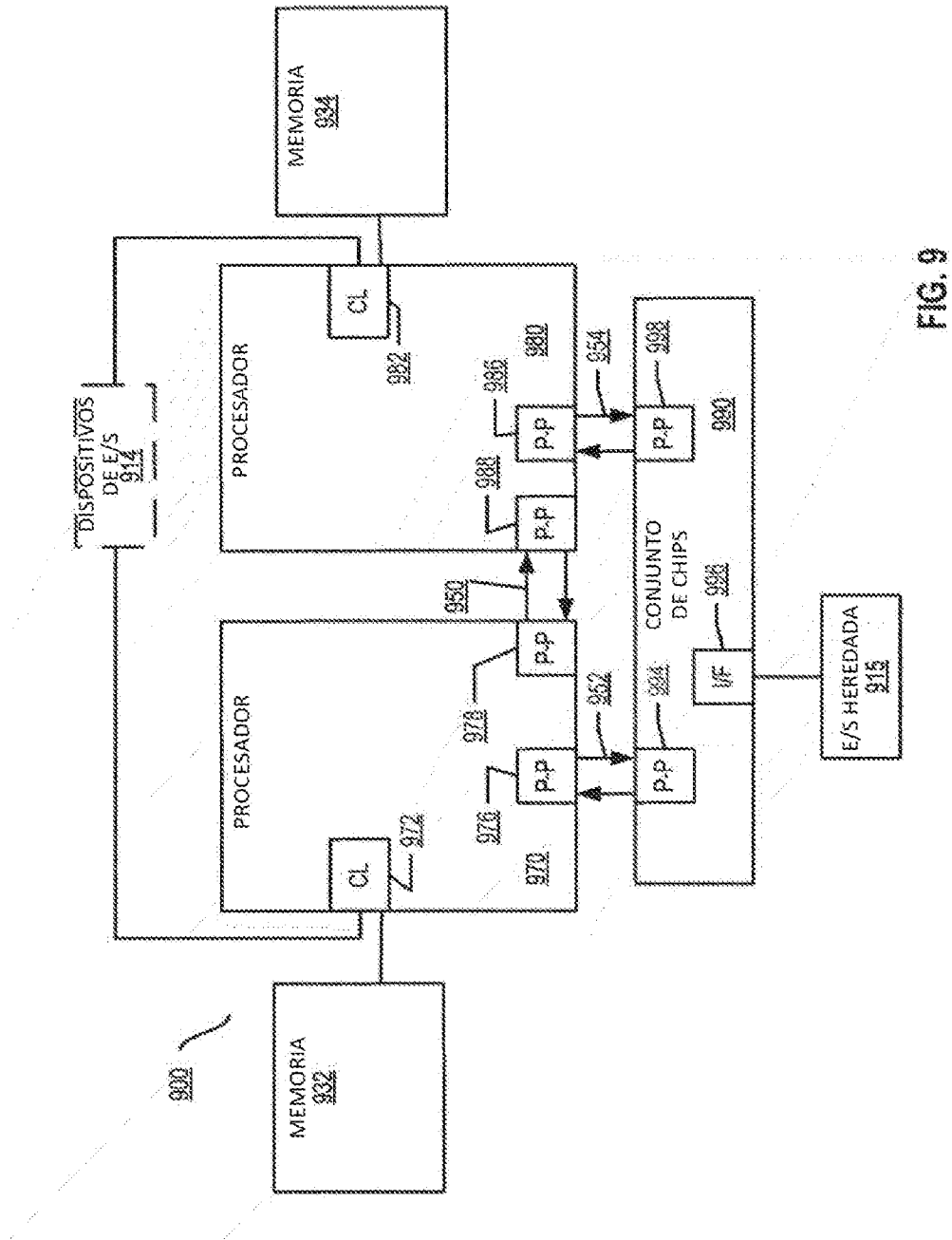
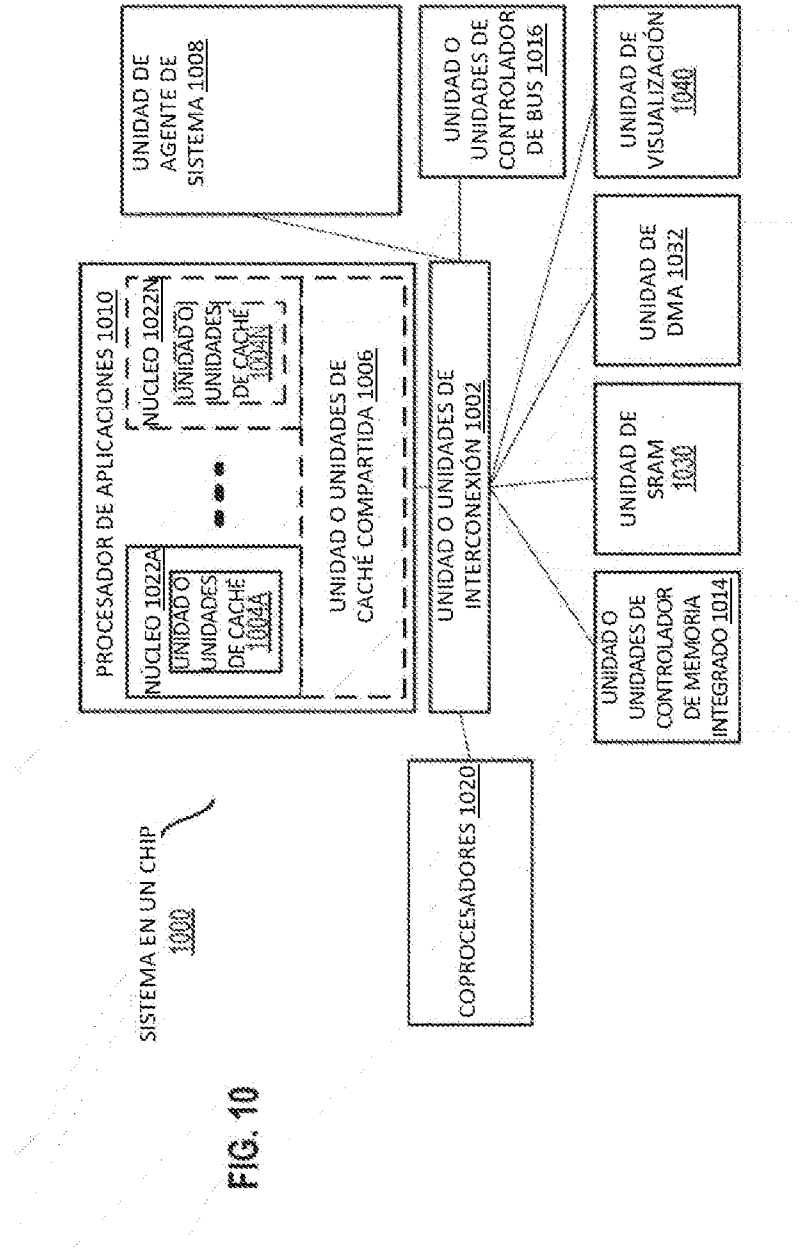
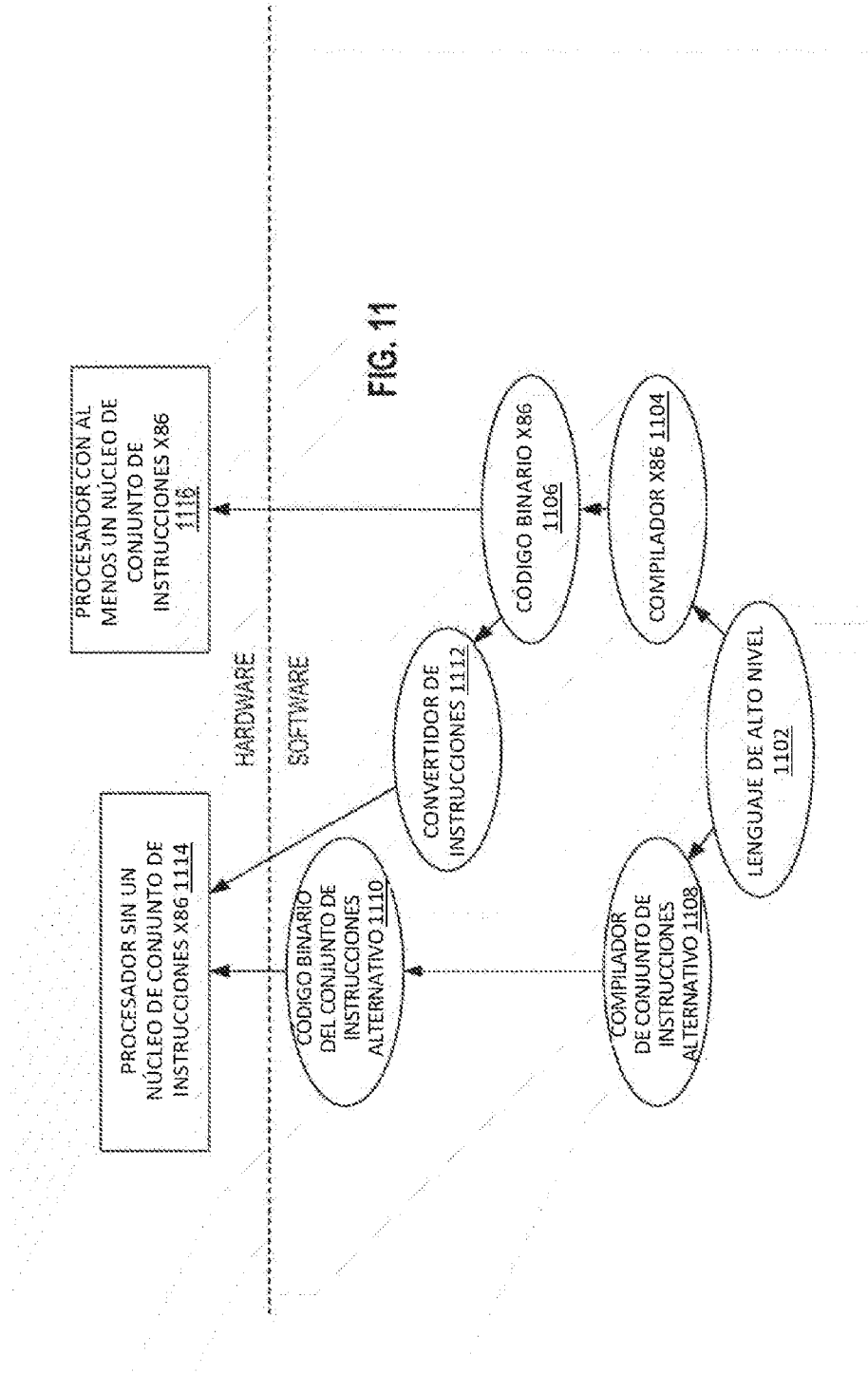


FIG. 9





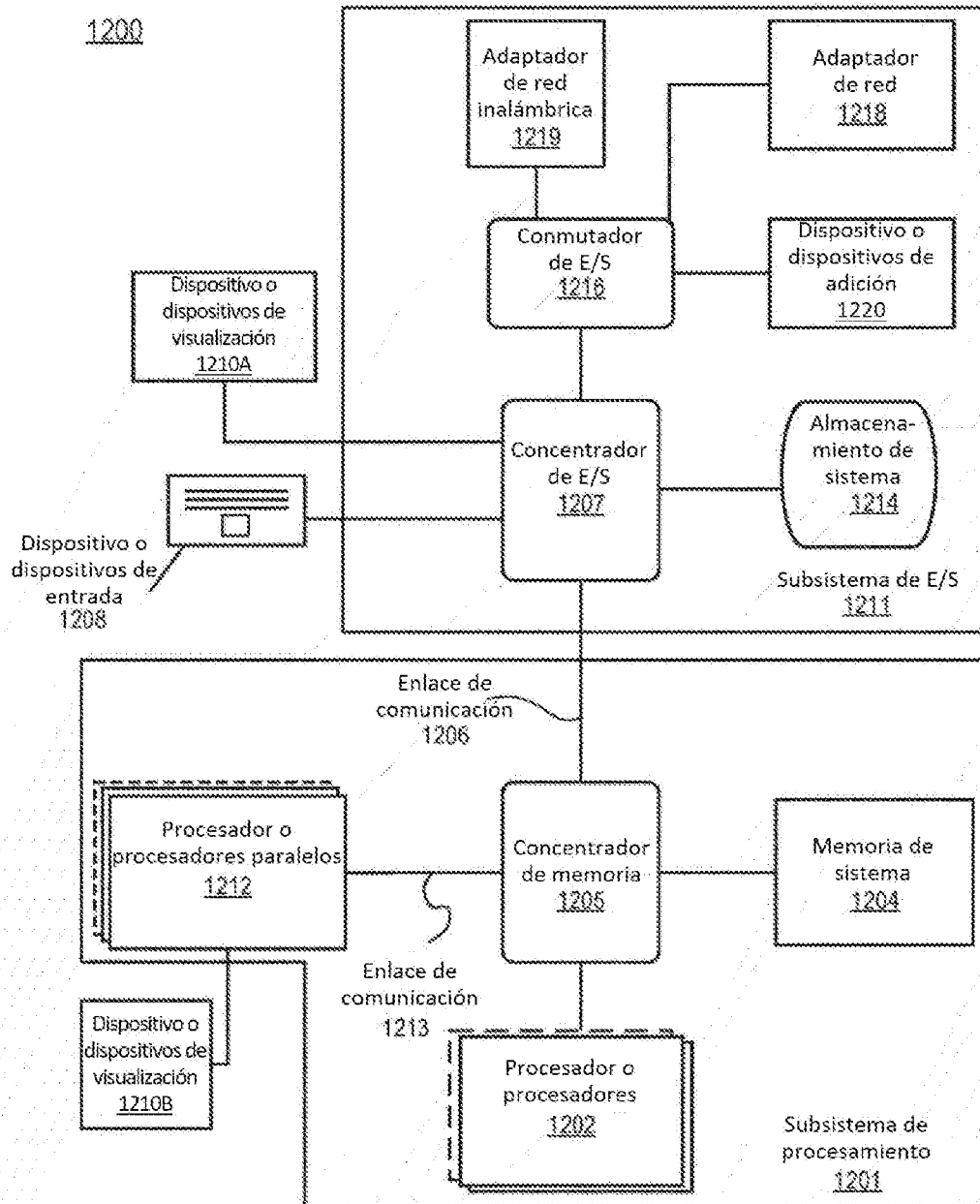


FIG. 12

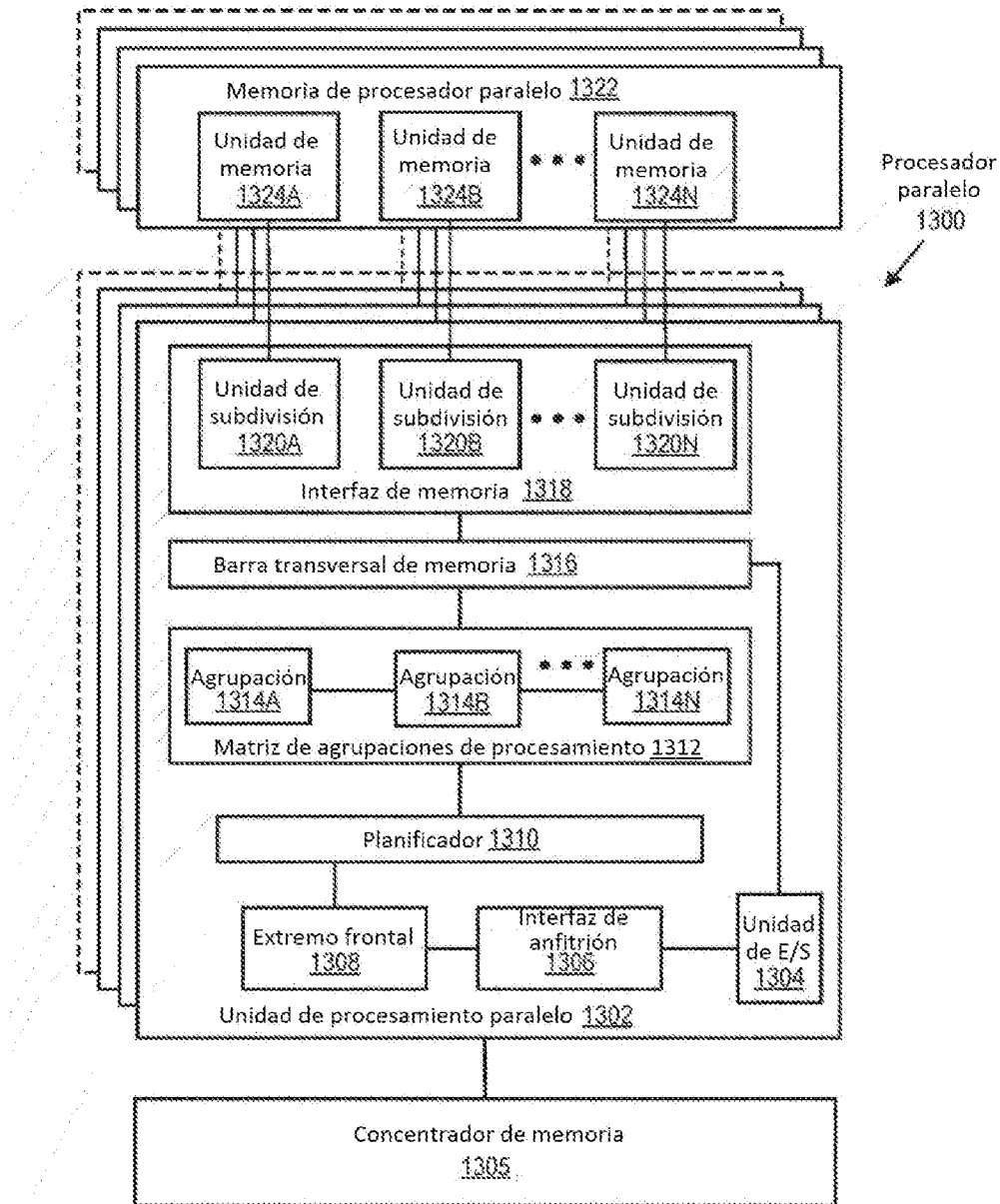


FIG. 13A

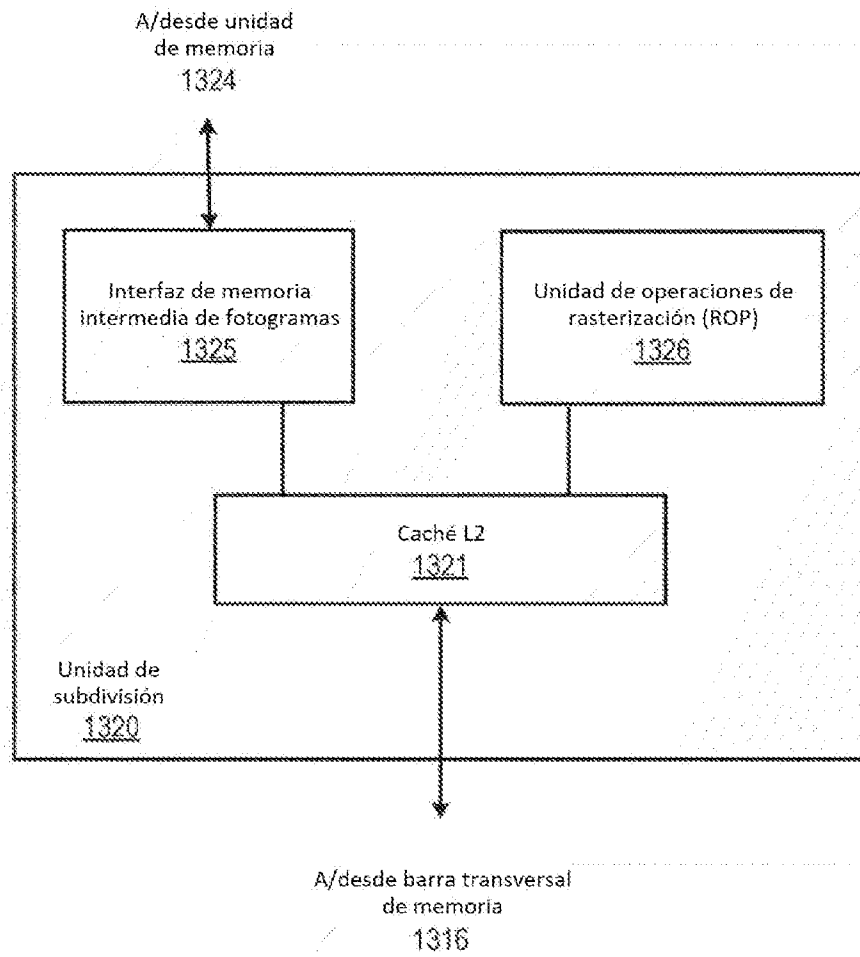


FIG. 13B

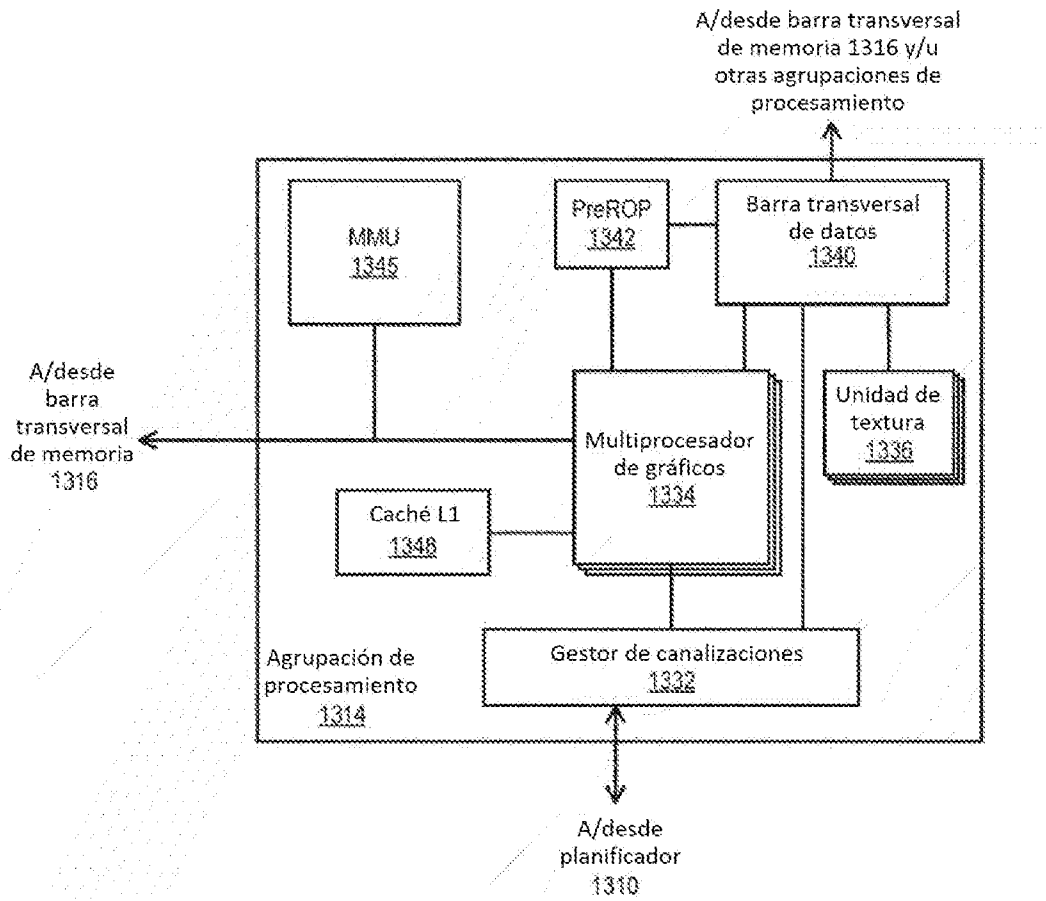


FIG. 13C

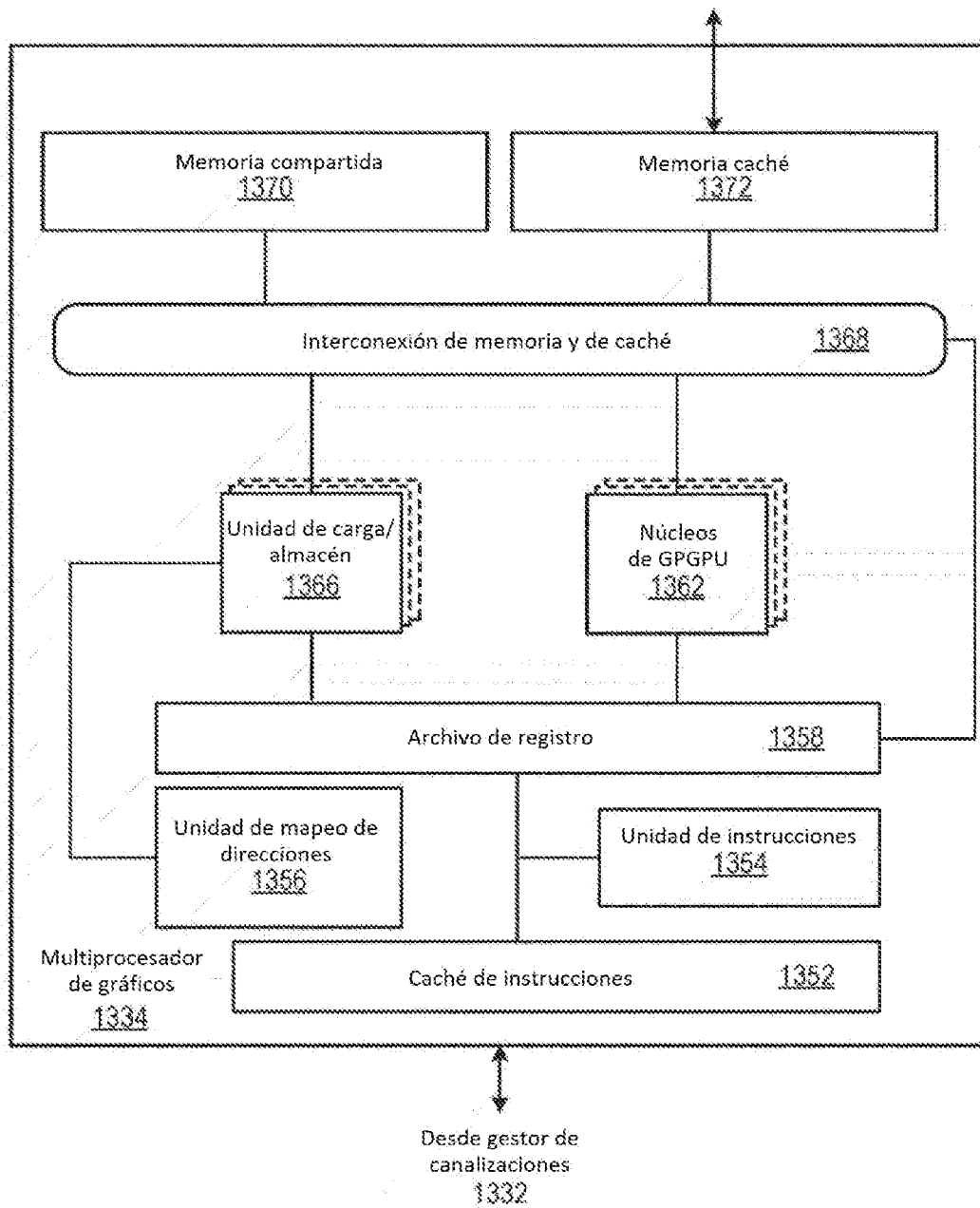


FIG. 13D

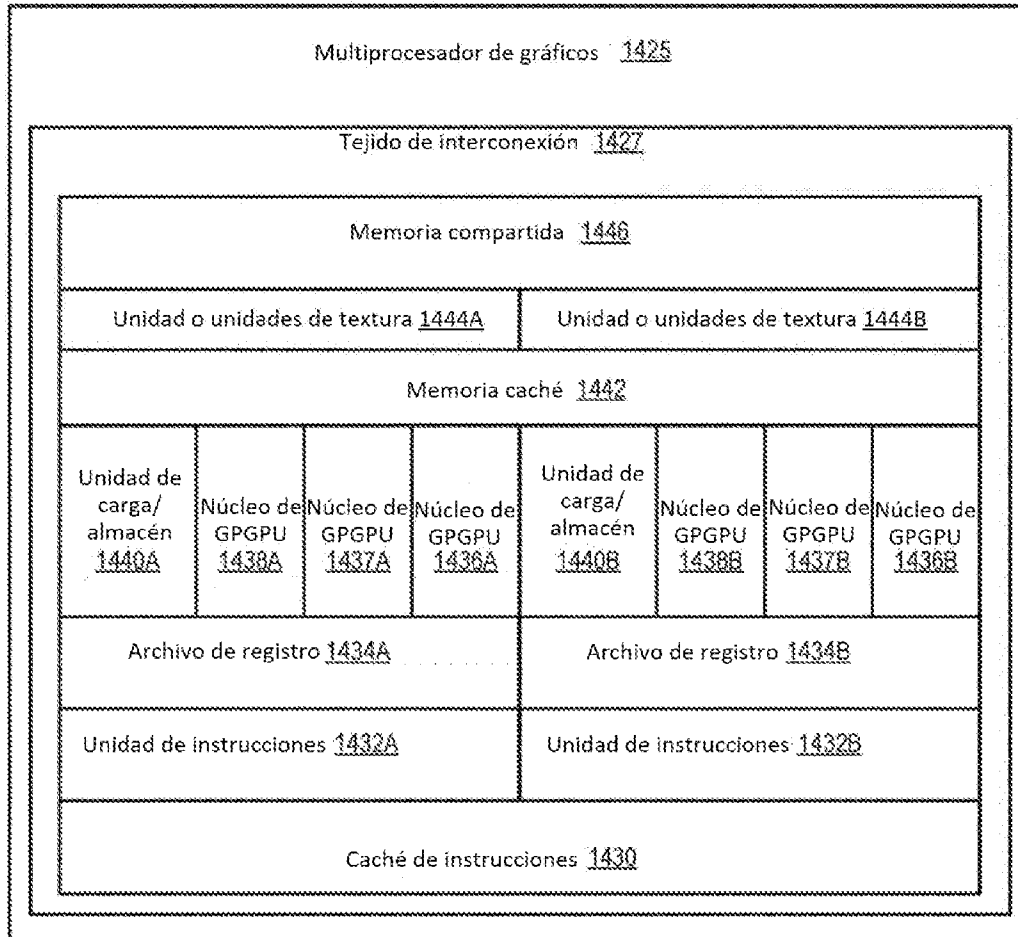


FIG. 14A

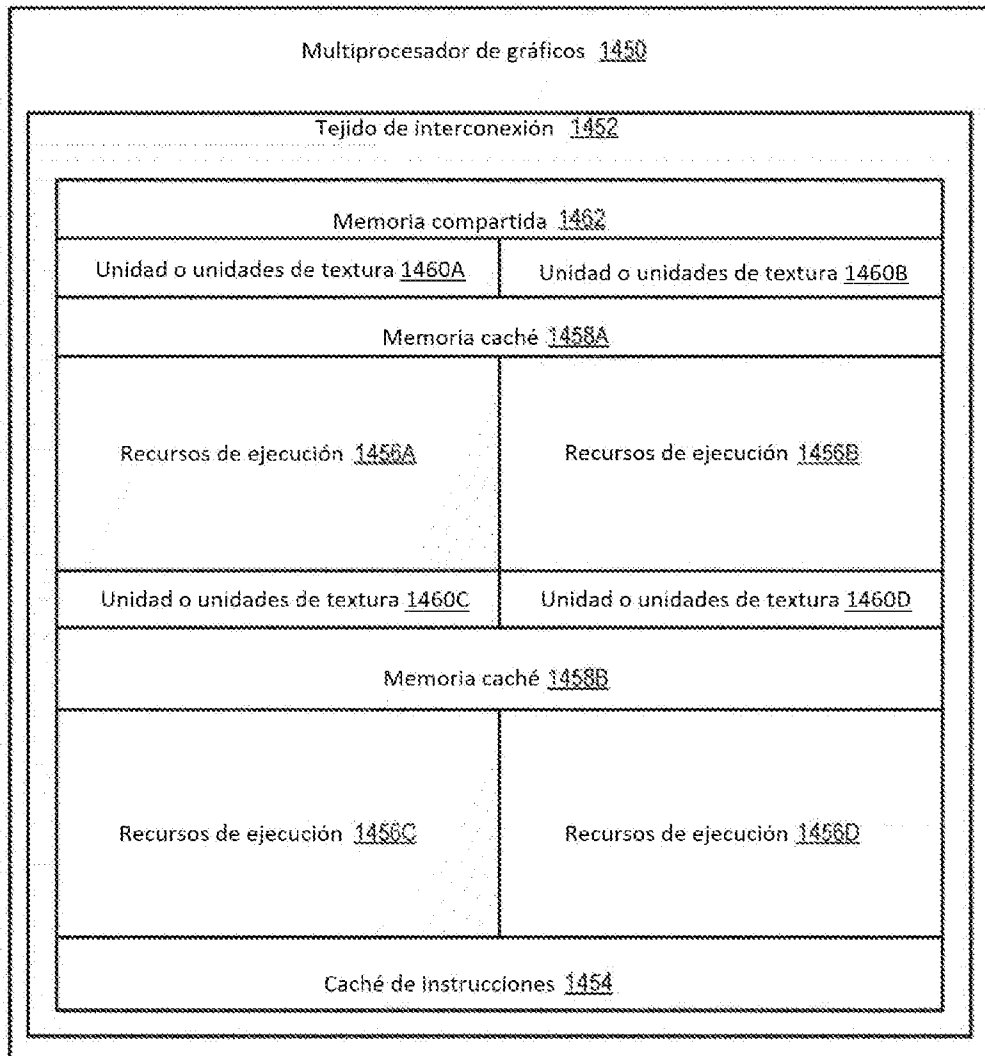


FIG. 14B

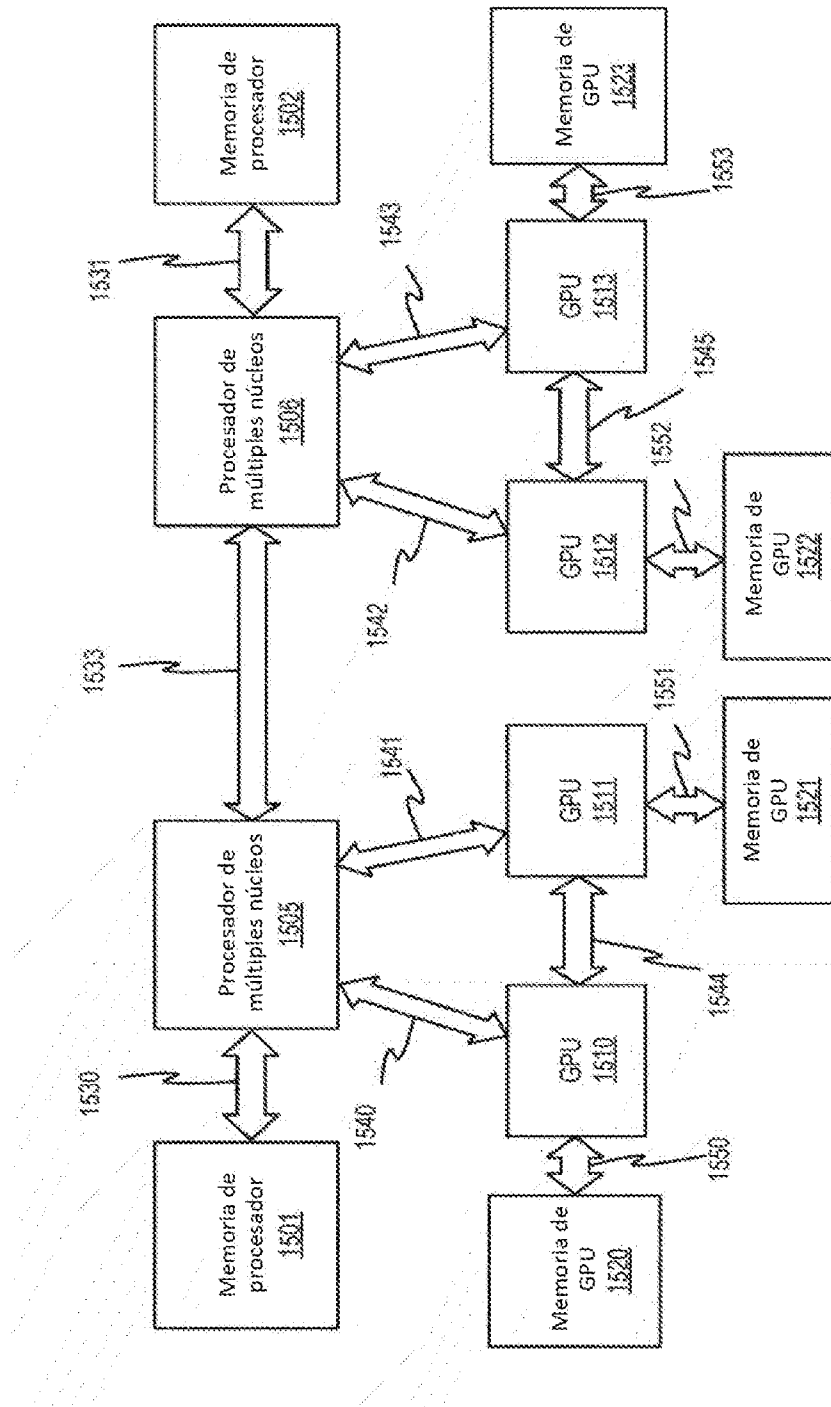


FIG. 15A

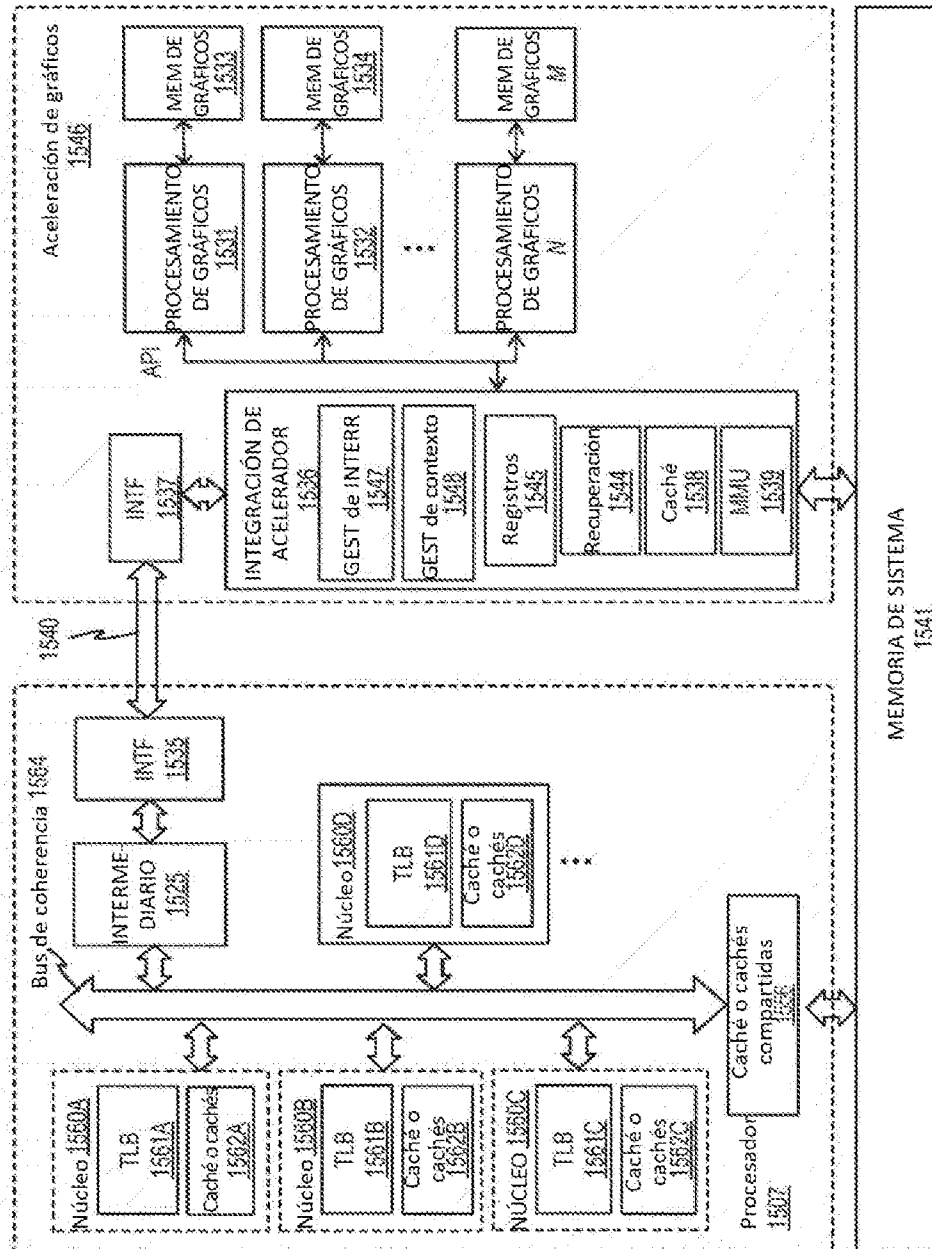


FIG. 15B

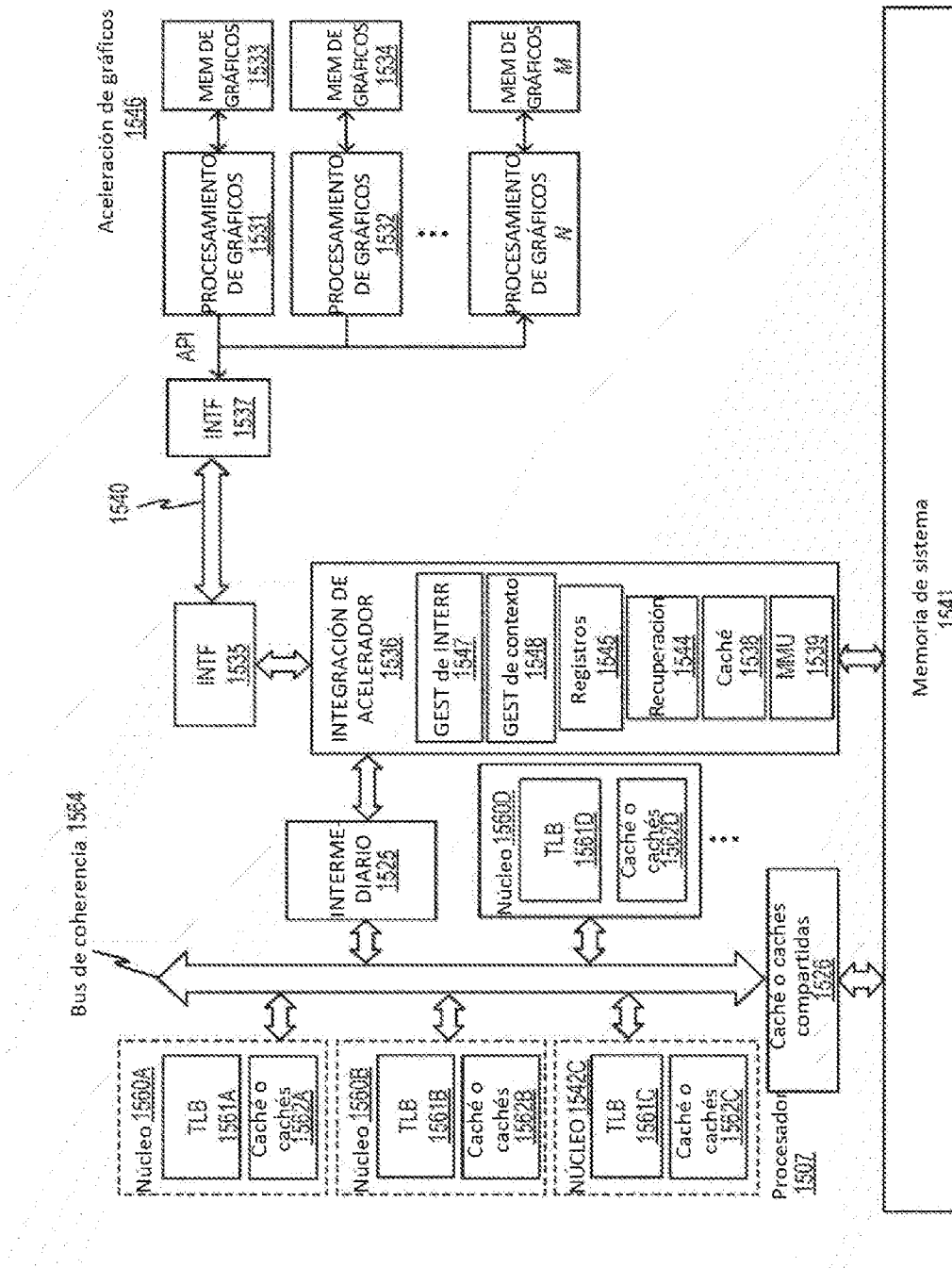


FIG. 15C

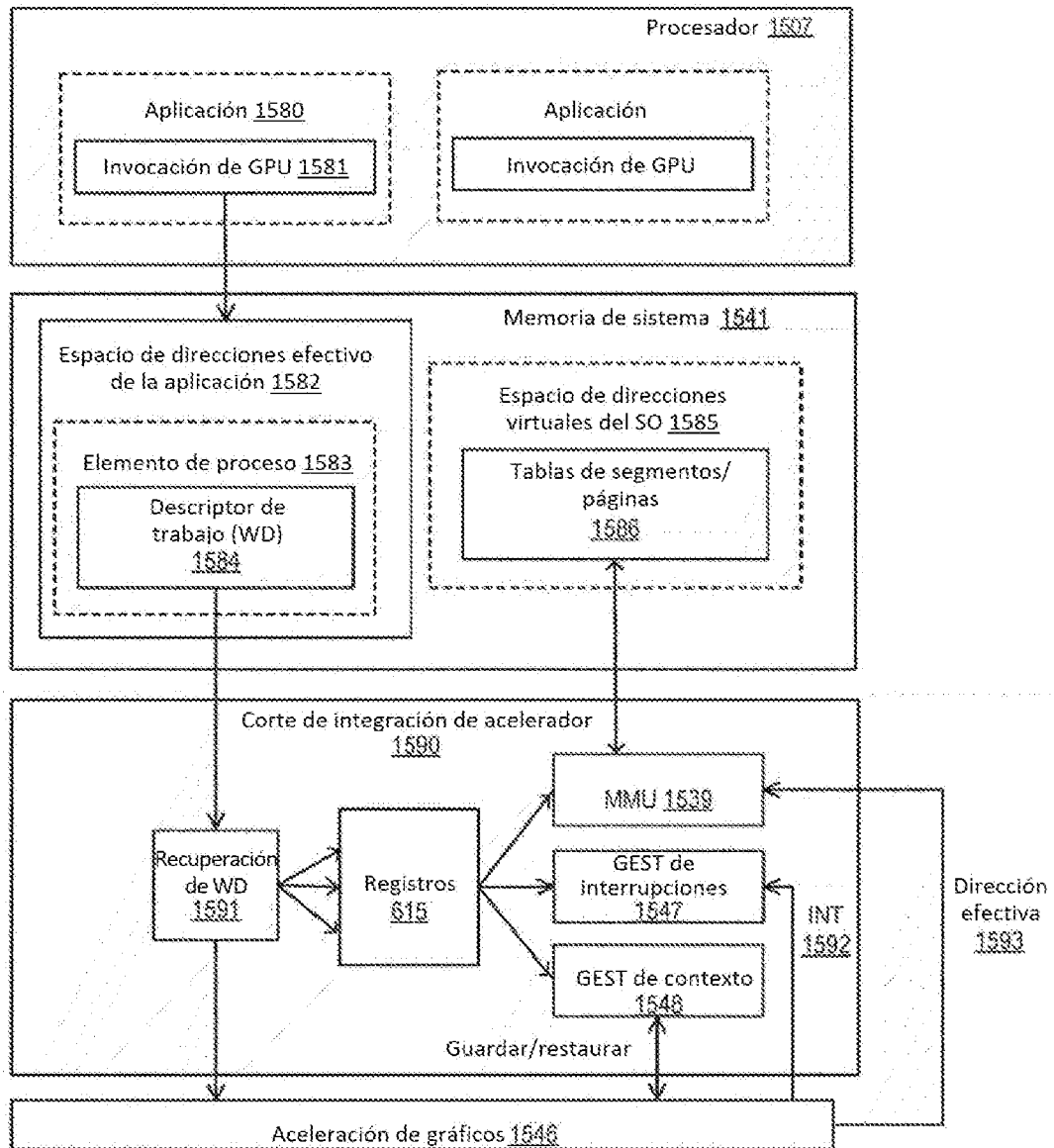


FIG. 15D

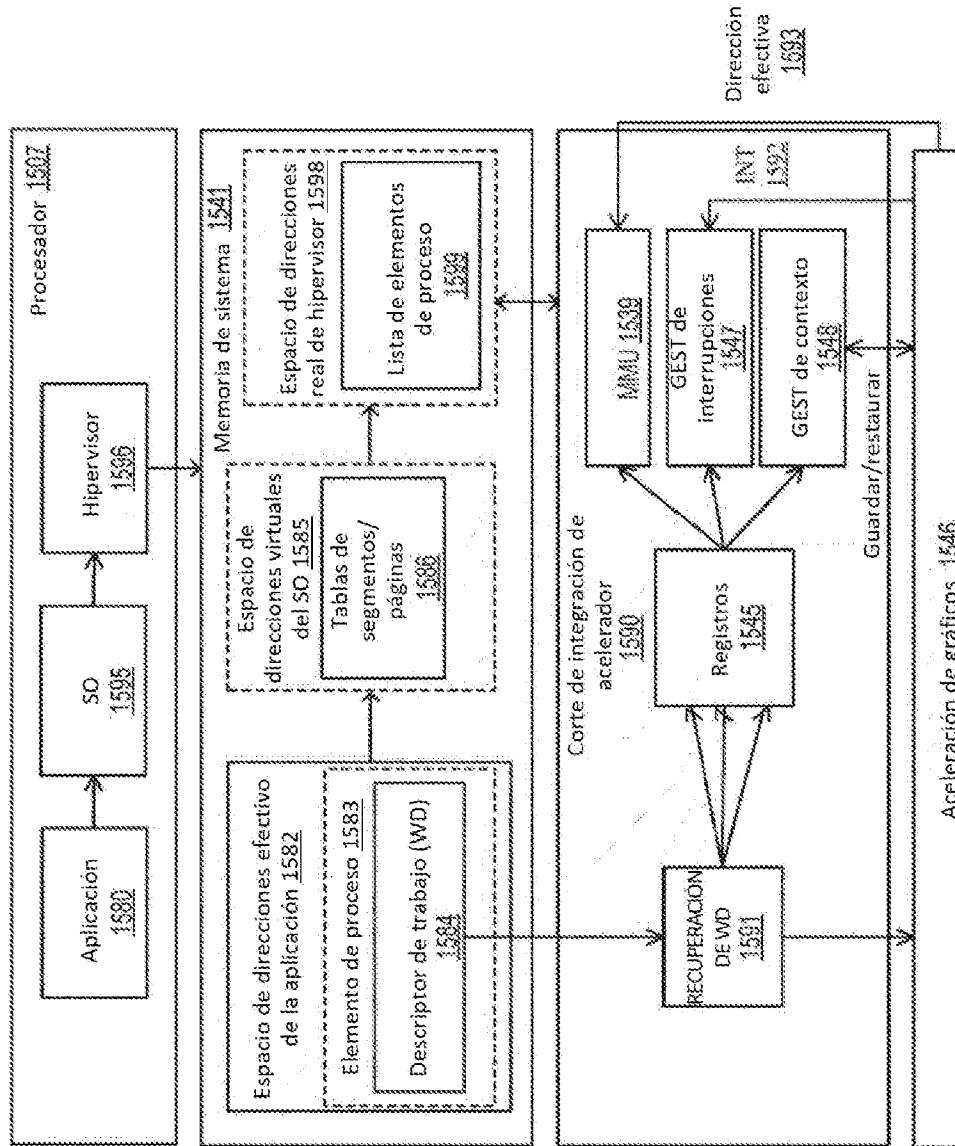


FIG. 15E

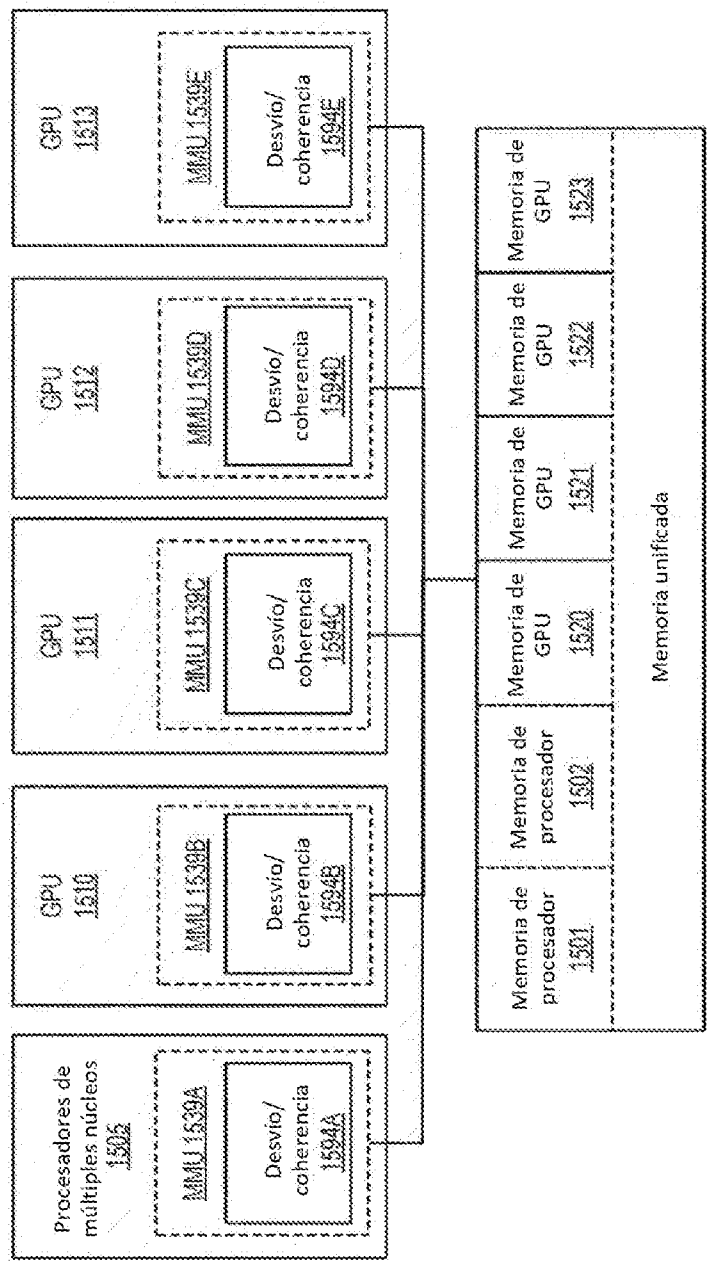


FIG. 15F

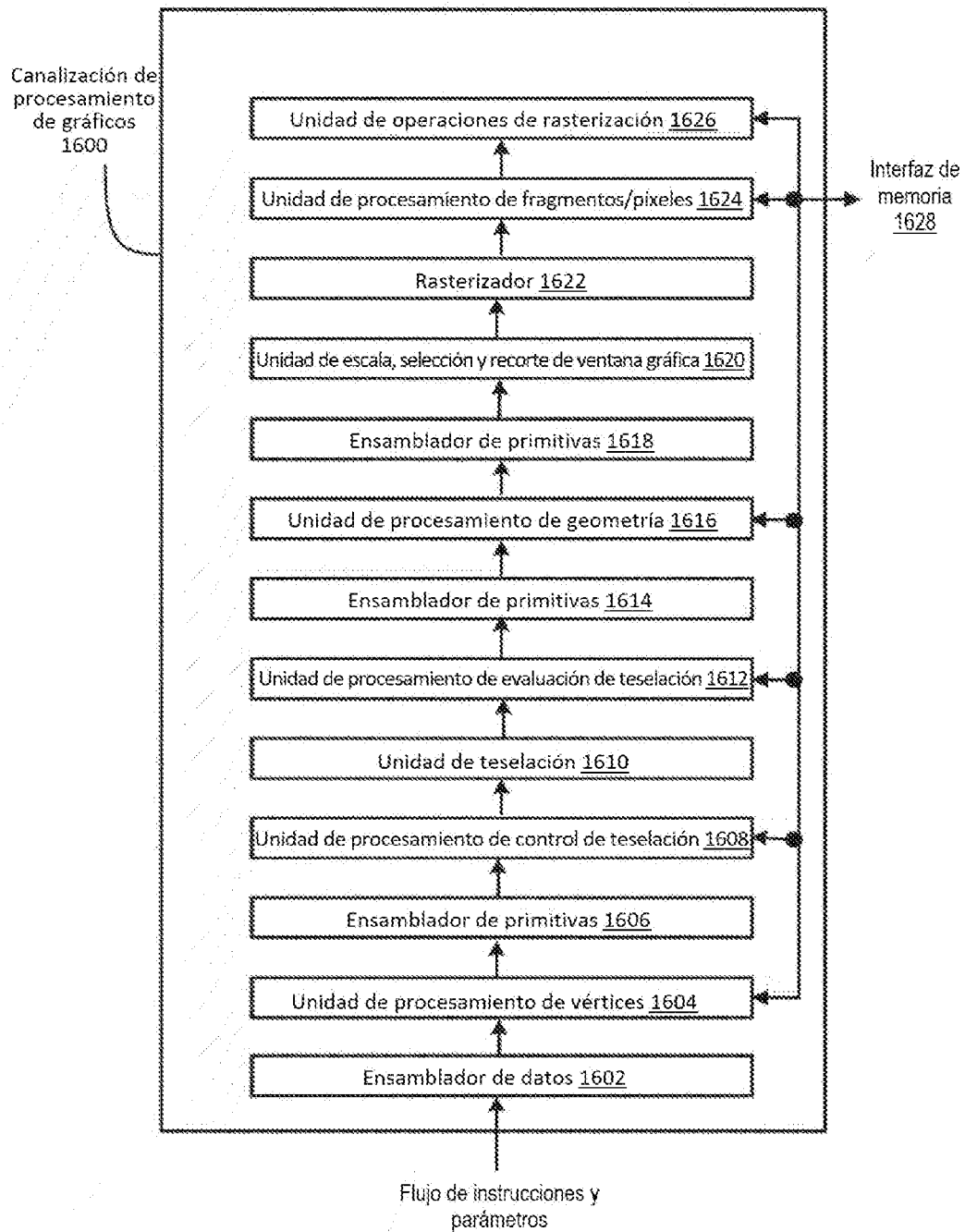


FIG. 16

FIG. 17

Ejecución de `vmatchindx zmm0[k0], zmm1, zmm2 1700`

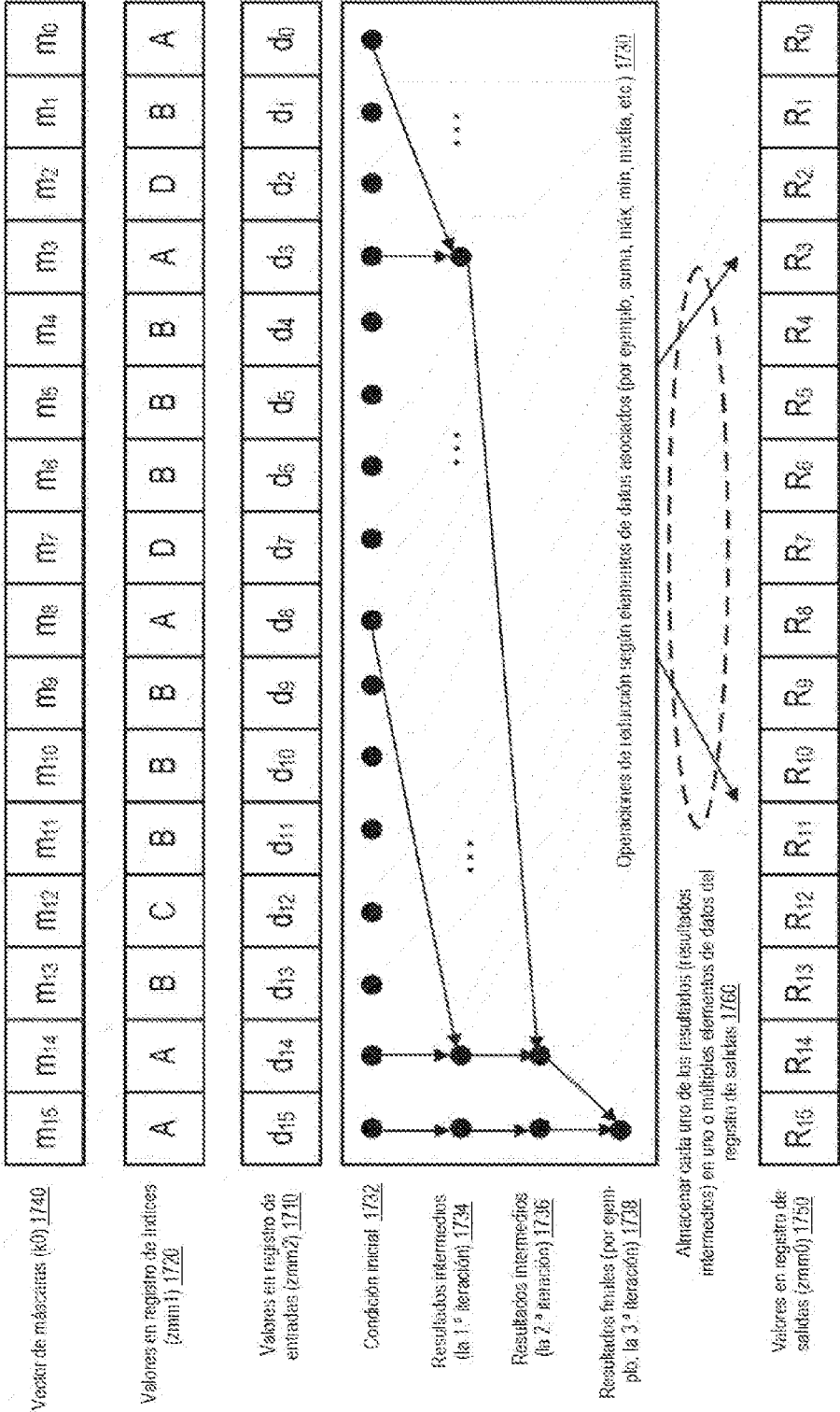


FIG. 18

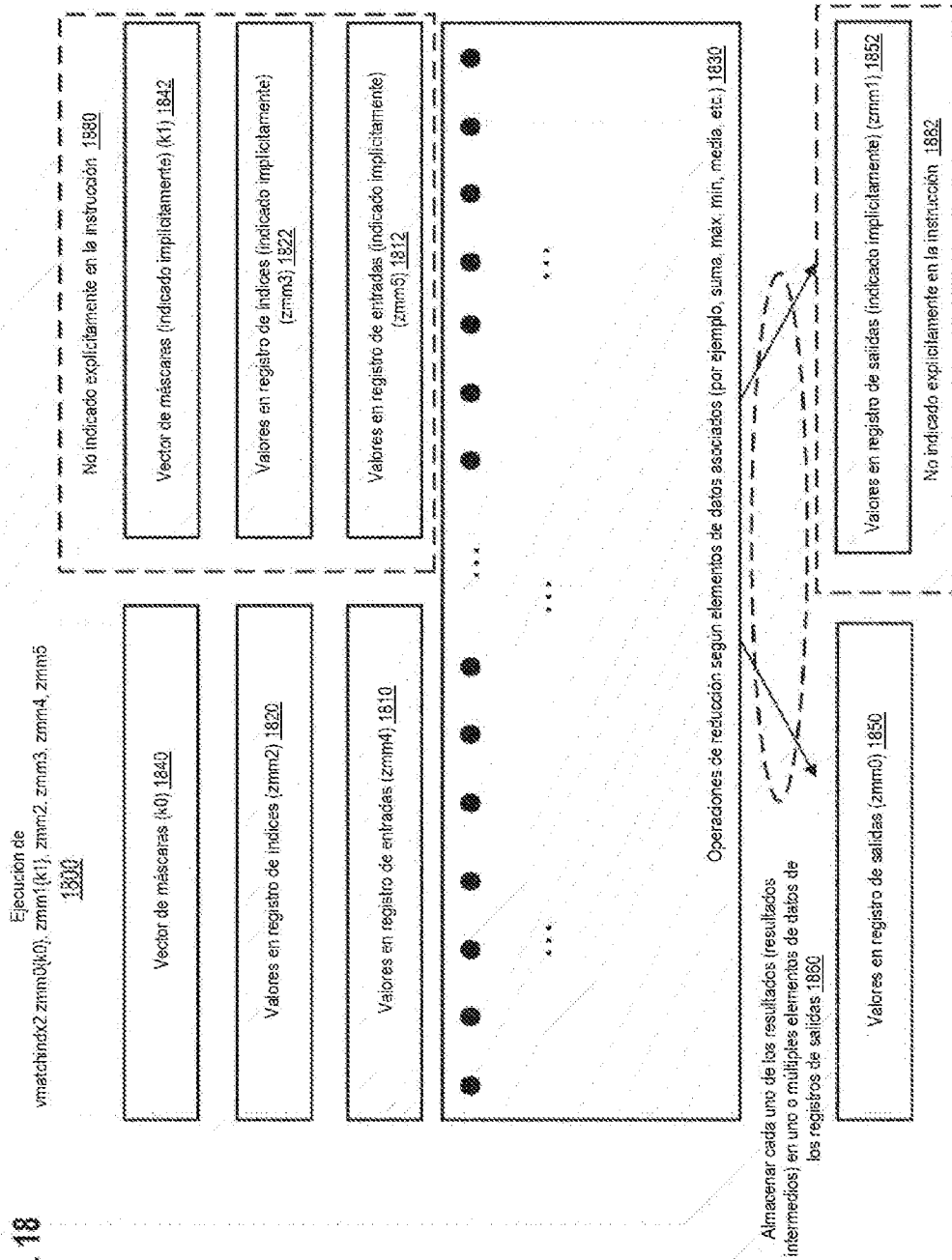
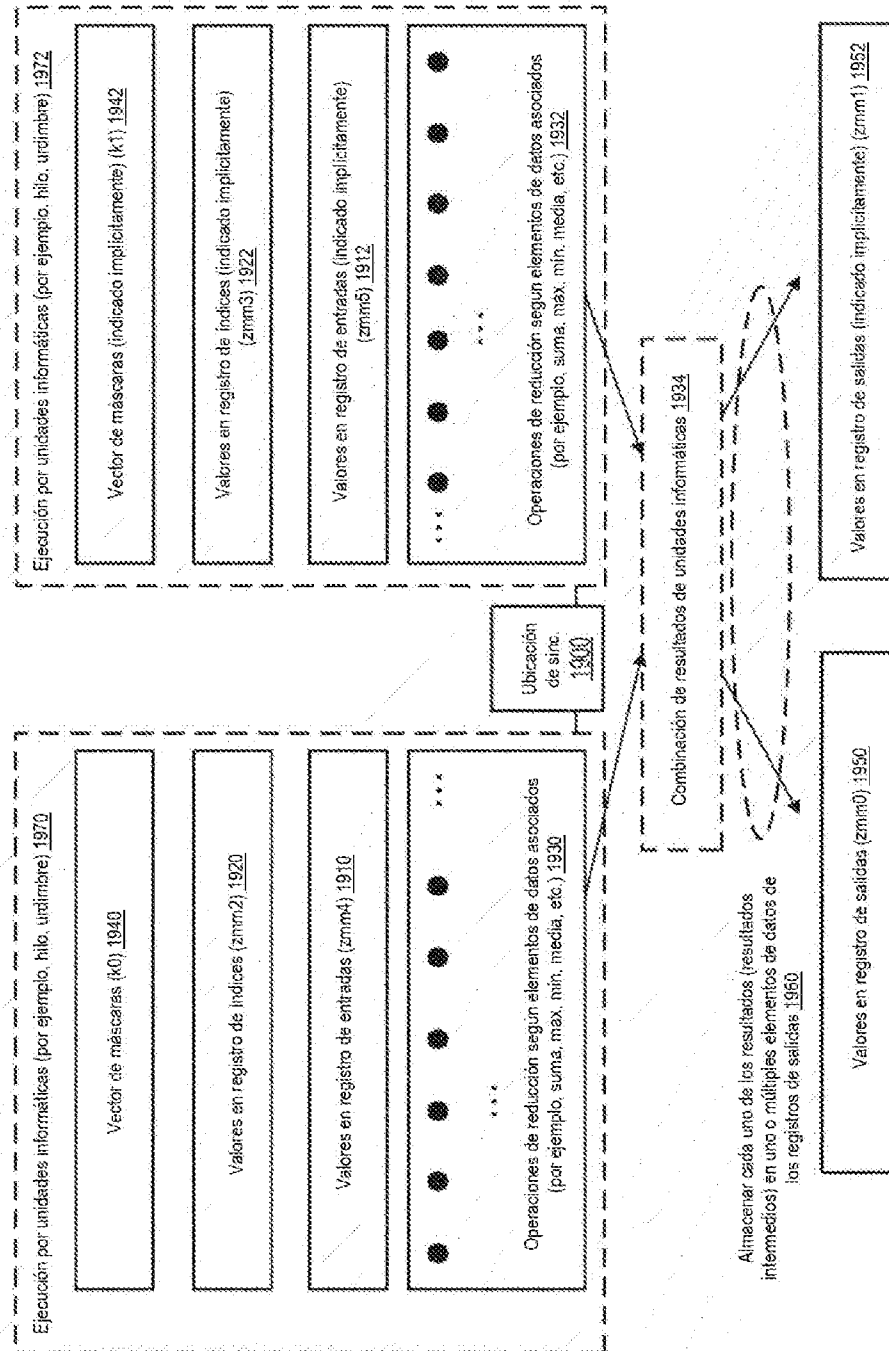


FIG. 19



2000

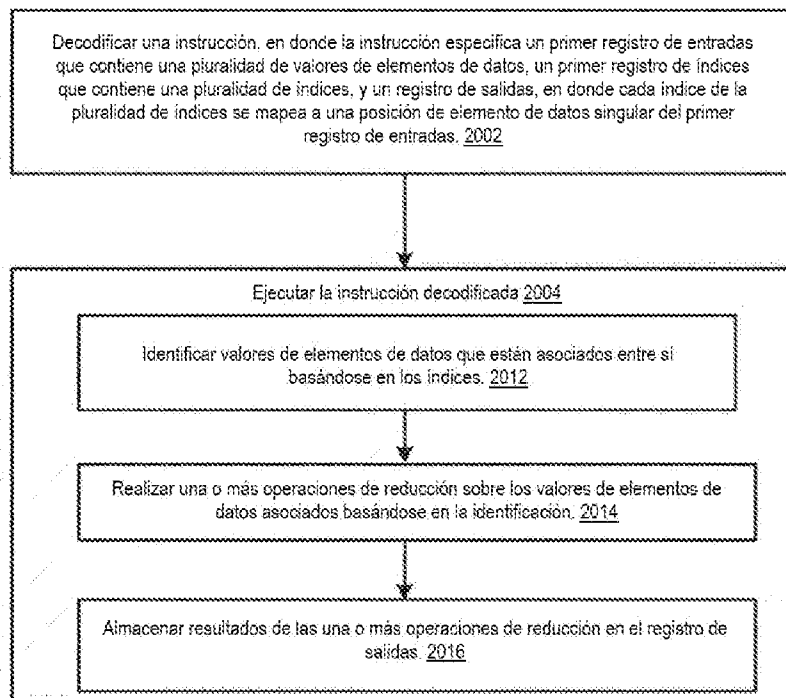


FIG. 20