(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 688 808**

(51) Int. CI.:

G06F 1/32 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **31.08.2012 PCT/US2012/053341**

(87) Fecha y número de publicación internacional: **28.03.2013 WO13043350**

(96) Fecha de presentación y número de la solicitud europea: **31.08.2012 E 12762460 (9)**

(97) Fecha y número de publicación de la concesión europea: **04.07.2018 EP 2758853**

(54) Título: **Optimización dinámica de energía para dispositivos informáticos**

(30) Prioridad:

20.09.2011 US 201161536684 P
23.11.2011 US 201113303871

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
07.11.2018

(73) Titular/es:

QUALCOMM INCORPORATED (100.0%)
5775 Morehouse Drive
San Diego, CA 92121, US

(72) Inventor/es:

VICK, CHRISTOPHER A. y
WRIGHT, GREGORY M.

(74) Agente/Representante:

FORTEA LAGUNA, Juan José

DESCRIPCIÓN

Optimización dinámica de energía para dispositivos informáticos

SOLICITUDES RELACIONADAS

[1] Esta solicitud reivindica el beneficio de la prioridad de la solicitud provisional de Estados Unidos n.º 61/536.684, titulada «Dynamic Power Optimization For Computing Devices» [«Optimización dinámica de energía para dispositivos informáticos»] presentada el 20 de septiembre de 2011.

ANTECEDENTES

[2] Las tecnologías de comunicación celular e inalámbrica han experimentado un crecimiento fulminante en los últimos años. Este crecimiento se ha visto impulsado por mejores comunicaciones, hardware, redes más grandes y protocolos más fiables. Los proveedores de servicios inalámbricos actualmente pueden ofrecer a sus clientes una variedad cada vez mayor de funciones y servicios, y proporcionar a los usuarios niveles inauditos de acceso a información, recursos y comunicaciones. Para adaptarse al ritmo de estas mejoras en el servicio, los dispositivos electrónicos móviles (por ejemplo, teléfonos celulares, tabletas, ordenadores portátiles, etc.) son más potentes que nunca. Los usuarios de dispositivos móviles en la actualidad ejecutan rutinariamente múltiples aplicaciones y servicios de software de gran complejidad y consumo de energía en sus dispositivos móviles, sin ninguna conexión cableada a una fuente de energía. Como resultado, la vida útil de la batería de un dispositivo móvil y las características de consumo de energía se están convirtiendo en cuestiones cada vez más importantes para los consumidores de dispositivos móviles. Por ejemplo, el documento US 2005/0229149 A1 describe un compilador con optimización de energía.

[3] La mayor duración de la batería aumenta la satisfacción del usuario al permitir a este realizar más acciones con un dispositivo inalámbrico durante períodos de tiempo más largos. Para aumentar la duración de la batería, los dispositivos móviles típicamente intentan optimizar el consumo de energía de los dispositivos móviles utilizando técnicas dinámicas de escalado de voltaje y frecuencia. Estas técnicas permiten que unos canales de dispositivos programables se ejecuten en un modo de menor energía y/o menor rendimiento cuando se detectan aplicaciones no esenciales o condiciones de baja carga. Por ejemplo, un dispositivo móvil puede estar configurado para poner uno o más procesadores y/o recursos en un estado de baja energía cuando están inactivos. Si bien estas técnicas pueden mejorar el rendimiento general de la batería, requieren que los procesadores y/o recursos del dispositivo se pongan en un estado inactivo y no pueden mejorar las características de consumo de energía de aplicaciones o procesos individuales que se ejecutan en el dispositivo. Por lo tanto, las técnicas existentes intentan adaptar el comportamiento del dispositivo móvil a las aplicaciones de software que se ejecutan en el dispositivo, en lugar de adaptar las aplicaciones para que consuman menos energía en el dispositivo. Dado que muchas aplicaciones de software modernas requieren un procesamiento de gran consumo de energía, la reducción del consumo de energía de los procesos que se ejecutan en el dispositivo, sin alterar el rendimiento de los procesos, mejorará en gran medida la satisfacción del usuario.

SUMARIO

[4] Los diversos aspectos incluyen procedimientos de optimización de código objeto para ahorrar energía durante la ejecución en un dispositivo informático, que incluyen recibir un código objeto binario compilado en software de sistema, analizar el código objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que puede optimizarse para ahorrar energía, realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada, y ejecutar el código objeto de energía optimizada en un procesador del dispositivo informático. En un aspecto, el software de sistema que recibe el código objeto binario compilado es uno de una máquina virtual de sistema o un hipervisor. En un aspecto, el software de sistema que recibe el código objeto binario compilado es un sistema operativo. En un aspecto, realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada incluye traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones. En un aspecto, la primera arquitectura de conjunto de instrucciones es la misma que la segunda arquitectura de conjunto de instrucciones. En un aspecto, analizar el código objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que se pueden optimizar para ahorrar energía incluye determinar si hay operaciones alternativas que logran los mismos resultados que las operaciones de código objeto identificadas, y realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada incluye reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas. En un aspecto, el procedimiento incluye además detectar una conexión a una nueva fuente de energía. En un aspecto, realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada se realiza cuando se detecta la

conexión a la nueva fuente de energía. En un aspecto, analizar el código objeto recibido incluye utilizar un modelo de consumo de energía para identificar segmentos de código objeto que pueden optimizarse para eficiencia energética. En un aspecto, el procedimiento incluye además medir una cantidad de energía consumida en la ejecución de segmentos de código objeto de energía optimizada, comparar la cantidad medida de energía consumida con unas predicciones del modelo de consumo de energía y modificar el modelo de consumo de energía basándose en un resultado de la comparación.

[5] Otros aspectos incluyen un dispositivo informático configurado para optimizar un código objeto durante una ejecución para un ahorro de energía mejorado, que incluye medios para recibir en un código objeto binario compilado en software de sistema, medios para analizar el código objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que pueden optimizarse para ahorrar energía, medios para realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada, y medios para ejecutar el código objeto de energía optimizada en un procesador del dispositivo informático. En un aspecto, unos medios para realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada incluyen medios para traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones. En un aspecto, unos medios para traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones incluyen medios para traducir la primera arquitectura de conjunto de instrucciones como una arquitectura de conjunto de instrucciones que es la misma que la segunda arquitectura de conjunto de instrucciones. En un aspecto, unos medios para analizar el código objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que se pueden optimizar para ahorrar energía incluyen medios para determinar si hay operaciones alternativas que logran los mismos resultados que las operaciones de código objeto identificadas. En un aspecto, unos medios para realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada incluyen medios para reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas. En un aspecto, el dispositivo informático incluye además medios para detectar una conexión a una nueva fuente de energía. En un aspecto, unos medios para realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada incluye medios para traducir el código recibido para generar un código objeto de energía optimizada cuando se detecta la conexión a la nueva fuente de energía. En un aspecto, unos medios para analizar el código objeto recibido incluyen medios para utilizar un modelo de consumo de energía para identificar segmentos de código objeto que pueden optimizarse para eficiencia energética. En un aspecto, el dispositivo informático incluye además medios para medir una cantidad de energía consumida en la ejecución de segmentos de código objeto de energía optimizada, medios para comparar la cantidad medida de energía consumida con predicciones del modelo de consumo de energía, y medios para modificar el modelo de consumo de energía basándose en un resultado de la comparación.

[6] Otros aspectos incluyen un dispositivo informático que incluye una memoria y un procesador acoplado a la memoria, en el que el procesador está configurado con instrucciones ejecutables por procesador para realizar unas operaciones que incluyen recibir el código objeto binario compilado en software de sistema, analizar el código objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que pueden optimizarse para ahorrar energía, realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada, y ejecutar el código objeto de energía optimizada en un procesador del dispositivo informático. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones de tal forma que realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada incluye traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones de tal forma que la primera arquitectura de conjunto de instrucciones es la misma arquitectura de conjunto de instrucciones que la segunda arquitectura de conjunto de instrucciones.

[7] En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones de tal forma que analizar el código del objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que pueden optimizarse para ahorrar energía incluye determinar si hay operaciones alternativas que logran los mismos resultados que las operaciones de código objeto identificadas, y realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada incluye reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas. En un aspecto, las instrucciones de software ejecutable por procesador almacenadas están configuradas para hacer

que un procesador realice unas operaciones que incluyen la detección de una conexión a una nueva fuente de energía. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones de tal forma que la realización en el proceso de traductor binario dinámico de una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada tiene lugar cuando se detecta una conexión a la nueva fuente de energía. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones de tal forma que analizar el código objeto recibido incluye utilizar un modelo de consumo de energía para identificar segmentos de código objeto que se pueden optimizar para eficiencia energética. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones que incluyen además medir una cantidad de energía consumida en la ejecución de segmentos de código objeto de energía optimizada, comparar la cantidad medida de energía consumida con unas predicciones del modelo de consumo de energía, y modificar el modelo de consumo de energía basándose en un resultado de la comparación.

[8] Otros aspectos incluyen un medio de almacenamiento no transitorio legible por procesador que tiene almacenadas instrucciones de software ejecutables por procesador configuradas para hacer que un procesador realice unas operaciones para optimizar un código objeto para ahorrar energía durante la ejecución en un dispositivo informático, incluyendo las operaciones recibir un código objeto binario compilado en software de sistema, analizar el código objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que pueden optimizarse para ahorrar energía, realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada, y ejecutar el código objeto de energía optimizada en un procesador del dispositivo informático. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones de tal forma que realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada incluye traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones de tal forma que la primera arquitectura de conjunto de instrucciones es la misma arquitectura de conjunto de instrucciones que la segunda arquitectura de conjunto de instrucciones. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones de tal forma que analizar el código objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que se pueden optimizar para ahorrar energía incluye determinar si hay operaciones alternativas que logran los mismos resultados que las operaciones de código objeto identificadas. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas además para hacer que un procesador realice unas operaciones de tal forma que realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada incluye reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones que incluyen detectar una conexión a una nueva fuente de energía. En un aspecto, las instrucciones ejecutables por procesador almacenadas están configuradas además para hacer que un procesador realice unas operaciones de tal forma que realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada tiene lugar cuando se detecta una conexión a la nueva fuente de energía. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones de tal forma que analizar el código objeto recibido incluye utilizar un modelo de consumo de energía para identificar segmentos de código objeto que se pueden optimizar para eficiencia energética. En un aspecto, las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice unas operaciones que incluyen además medir una cantidad de energía consumida en la ejecución de segmentos de código objeto de energía optimizada, comparar la cantidad medida de energía consumida con unas predicciones del modelo de consumo de energía, y modificar el modelo de consumo de energía basándose en un resultado de la comparación.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

[9] Los dibujos adjuntos, que se incorporan al presente documento y forman parte de esta memoria descriptiva, ilustran modos de realización a modo de ejemplo de la presente invención, y junto con la descripción general dada anteriormente y la descripción detallada dada a continuación, sirven para explicar las características de la presente invención.

La figura 1 es un diagrama de arquitectura de ordenador en capas que ilustra unos componentes e interfaces lógicos en un sistema informático adecuado para implementar los diversos aspectos.

Las figuras 2A y 2B son diagramas de flujo de procesos que ilustran componentes lógicos y transformaciones de código para distribuir un código en un formato adecuado para implementar los diversos aspectos.

Las figuras 3A y 3B son diagramas de arquitectura de ordenador en capas que ilustran componentes lógicos en máquinas virtuales adecuadas para implementar los diversos aspectos.

La figura 4 es un diagrama de bloques de componentes que ilustra los componentes lógicos y los flujos de datos de una máquina virtual de sistema de acuerdo con un aspecto.

La figura 5 es un diagrama de flujo de procesos que ilustra un procedimiento de aspecto para generar un código objeto optimizado.

La figura 6 es un diagrama de flujo de componentes que ilustra componentes lógicos y flujos de datos para medir las características de consumo de energía de ejecución de un código para reoptimizar continuamente el código objeto generado de acuerdo con un procedimiento de aspecto.

La figura 7 es un diagrama de flujo de procesos que ilustra un procedimiento de aspecto para medir las características de consumo de energía de ejecución de un código y reoptimizar continuamente el código objeto.

La figura 8 es un diagrama de flujo de procesos que ilustra un procedimiento de aspecto para realizar optimizaciones de un código objeto tras haberse detectado una fuente de energía conectada.

La figura 9 es un diagrama de bloques de componentes que ilustra un dispositivo móvil adecuado para implementar los diversos aspectos.

La figura 10 es un diagrama de bloques de componentes que ilustra otro dispositivo móvil adecuado para implementar los diversos aspectos.

DESCRIPCIÓN DETALLADA

[10] Los diversos aspectos se describirán en detalle con referencia a los dibujos adjuntos. Siempre que sea posible, se utilizarán los mismos números de referencia en todos los dibujos para referirse a partes iguales o similares. Las referencias a ejemplos e implementaciones particulares se hacen con fines ilustrativos, y no pretenden limitar el alcance de la presente invención o de las reivindicaciones.

[11] Los términos «a modo de ejemplo» se utilizan en el presente documento para indicar que «sirve de ejemplo, caso o ilustración». No ha de interpretarse necesariamente que cualquier implementación, descrita en el presente documento como «a modo de ejemplo», es preferida o ventajosa con respecto a otras implementaciones.

[12] Los términos «dispositivo móvil» y «dispositivo informático» se utilizan indistintamente en el presente documento para referirse a uno cualquiera o todos los teléfonos celulares, asistentes de datos personales (PDA), ordenadores de bolsillo, receptores de correo electrónico inalámbricos (por ejemplo, los dispositivos Blackberry® y Treo®), teléfonos celulares con conexión a Internet multimedia (por ejemplo, Blackberry Storm®), receptores de sistema de posicionamiento global (GPS), controladores de juegos inalámbricos y dispositivos electrónicos personales similares que incluyen un procesador programable y funcionan con energía de batería de tal forma que los procedimientos de conservación de energía son beneficiosos.

[13] El término «recurso» se utiliza en el presente documento para referirse a cualquiera de una amplia diversidad de circuitos (por ejemplo, puertos, relojes, buses, osciladores, etc.), componentes (por ejemplo, memoria), señales (por ejemplo, señales de reloj), funciones y fuentes de tensión (por ejemplo, rieles de tensión), que se pueden utilizar para admitir procesadores y clientes que se ejecutan en un dispositivo informático.

[14] Como se ha analizado anteriormente, las técnicas existentes para incrementar la duración de la batería en general ponen uno o más procesadores y/o recursos en un estado de baja energía. Estas técnicas requieren que los procesadores/recursos del dispositivo se pongan en un estado inactivo o de baja frecuencia, y no cambian el código ejecutado por las aplicaciones o los procesos.

[15] Los diversos aspectos proporcionan procedimientos, sistemas y dispositivos que utilizan técnicas de virtualización que pueden implementarse dentro de una capa de hipervisor para reducir la cantidad de energía consumida por los procesadores/recursos activos. En un primer aspecto, una máquina virtual recibe un código objeto para su ejecución, analiza el código objeto para reconocer operaciones y parámetros que caracterizan las operaciones que los procesadores de dispositivo van a realizar, y realiza traducciones de binario a binario para transformar o traducir el código objeto en un nuevo código objeto que puede funcionar de manera más eficiente

en el hardware del dispositivo móvil específico. Este reconocimiento y transformación de código objeto se puede llevar a cabo de acuerdo con un modelo específico de dispositivo. Mediante un modelo que está asociado con la arquitectura de procesador de un dispositivo móvil dado, la máquina virtual puede determinar que la ejecución del código objeto en un dispositivo de hardware particular puede requerir mucha energía. La máquina virtual puede entonces traducir el código objeto binario como un segundo código objeto binario diferente que tiene diferentes operadores (por ejemplo, operaciones de cambio y adición frente a operaciones de multiplicación) para ahorrar energía. Por lo tanto, mediante una traducción directa de binario a binario, la información del código se puede preservar mientras se puede reducir la cantidad total de energía gastada para procesar el código objeto.

[16] En un segundo aspecto, el modelo de consumo de energía por código objeto y las traducciones realizadas para optimizar el código se actualizan basándose en unas mediciones de la energía real consumida por el código objeto optimizado previamente. De esta manera, el rendimiento real de los procesadores de dispositivo móvil puede utilizarse para optimizar el código objeto en lugar de depender de un modelo fijo que puede no reflejar la variabilidad de lote a lote en el rendimiento de procesador. En este aspecto, los diversos procesadores del dispositivo informático, tales como la unidad de procesador central, los procesadores de módem y un procesador de receptor GPS (por nombrar algunos), pueden instrumentarse para medir la energía consumida durante la ejecución de un código objeto. Para permitir el seguimiento del consumo de energía con unas transformaciones de optimización de código objeto particular, unos fragmentos o elementos relacionados de código objeto se etiquetan cuando se optimizan y transforman. Cuando un procesador ejecuta el código, el consumo de energía medido asociado con la etiqueta de código y la medición se comparan con un modelo de predicción de rendimiento como el mostrado en la siguiente figura. La comparación entre el consumo de energía real y el rendimiento previsto se retroalimenta al proceso de optimización para que puedan identificarse o utilizarse mejores procedimientos de optimización para subsiguientes optimizaciones de código objeto. A continuación, la máquina virtual puede reoptimizar un código objeto tal como se ha descrito anteriormente, de tal forma que la próxima vez la aplicación se ejecuta en el dispositivo móvil.

[17] En general, las técnicas de virtualización se implementan en una máquina virtual (VM), que es una aplicación de software que ejecuta programas de aplicación como una máquina de hardware físico. Específicamente, una máquina virtual proporciona una interfaz entre unos programas de aplicación y el hardware físico, lo que permite potencialmente que unos programas de aplicación vinculados a una arquitectura de conjunto de instrucciones (ISA) específica se ejecuten en hardware que implementa una arquitectura de conjunto de instrucciones diferente. La virtualización es beneficiosa en los diversos aspectos, porque los programas de aplicación se distribuyen típicamente como archivos binarios compilados que están vinculados a una arquitectura de conjunto de instrucciones específica y dependen de una interfaz de sistema operativo (OSI) específica. Sin ayuda de máquinas virtuales, unos archivos binarios compilados solo se pueden ejecutar en sistemas que admiten la arquitectura de conjunto de instrucciones específica (por ejemplo, Intel IA-32, etc.) y una interfaz de sistema operativo para la cual se ha compilado el código binario. Las máquinas virtuales se pueden aprovechar para eludir estas limitaciones agregando una capa de software que admite los requisitos de arquitectura del programa de aplicación y/o traduce la arquitectura de conjunto de instrucciones del programa de aplicación como la arquitectura de conjunto de instrucciones admitida por el hardware.

[18] La figura 1 ilustra un diagrama de arquitectura en capas de un procesador que muestra unos componentes lógicos e interfaces en un sistema informático típico adecuado para implementar los diversos aspectos. La arquitectura 100 del sistema informático ilustrada incluye componentes de hardware y componentes de software. Los componentes de hardware pueden incluir hardware 102 de ejecución (por ejemplo, un procesador de aplicación, un procesador de señales digitales, etc.), dispositivos 106 de entrada/salida y una o más memorias 104. Los componentes de software pueden incluir un sistema operativo 108, un módulo 110 de biblioteca y uno o más programas 112 de aplicación.

[19] Los programas 112 de aplicación utilizan una interfaz de programa de aplicación (API) para emitir llamadas de biblioteca de lenguaje de alto nivel (HLL) al módulo 110 de biblioteca. El módulo 110 de biblioteca utiliza una interfaz binaria de aplicaciones (ABI) para invocar servicios (por ejemplo, a través de llamadas de sistema operativo) en el sistema operativo 108. El sistema operativo 108 se comunica con los componentes de hardware utilizando una arquitectura de conjunto de instrucciones (ISA) específica, que es una lista de códigos de operación (opcodes) específicos y mandatos nativos implementados por el hardware 102 de ejecución.

[20] La interfaz binaria de aplicaciones define la máquina tal como la perciben los procesos de programa de aplicación, mientras que la interfaz del programa de aplicación especifica las características de la máquina tal como las percibe un programa de lenguaje de alto nivel. La ISA define la máquina tal como la percibe el sistema operativo.

[21] Las figuras 2A y 2B son diagramas de flujo de procesos que ilustran la conversión de las aplicaciones de software escritas en un lenguaje de alto nivel (por ejemplo, Java, C++, etc.) en código distribuible. Como se ha mencionado anteriormente, los programas de aplicación de dispositivo móvil se distribuyen típicamente como archivos binarios compilados (denominados «código objeto») que están vinculados a una ISA e interfaz de sistema operativo (OSI) específicos.

[22] La figura 2A ilustra un procedimiento 200 para convertir un código de un lenguaje 202 de alto nivel en el código objeto 206 distribuible para la entrega a un dispositivo móvil. Los programadores de aplicaciones pueden escribir código fuente 202 utilizando un lenguaje de alto nivel (Java, C++, etc.), que un compilador puede convertir en código objeto 206. El compilador puede estar organizado lógicamente en un componente frontal, un componente central y un componente final. El componente frontal del compilador puede recibir el código fuente 202 y realizar unas operaciones de verificación de tipo, verificar la sintaxis y la semántica del código fuente y generar una representación intermedia 204 del código fuente. El componente central del compilador puede realizar unas operaciones para optimizar el código intermedio 204, tal como eliminar un código inútil o inalcanzable, reubicar cálculos, etc. El componente final del compilador puede traducir el código intermedio 204 optimizado como un código binario/objeto 206, que codifica las instrucciones de máquina específicas que se ejecutarán mediante una combinación específica de hardware y OSI. El código binario/objeto 206 puede entonces distribuirse a dispositivos que admiten la combinación específica de ISA y OSI para la que se ha generado el binario, y puede almacenarse en una memoria física y recuperarse mediante un cargador como una imagen 208 de memoria.

[23] La figura 2B ilustra un procedimiento 250 de aspecto para convertir un código de lenguaje 252 de alto nivel en el código distribuible 256 para su entrega a un dispositivo móvil que tiene software de virtualización. Un módulo compilador puede recibir un código fuente 252 escrito en lenguaje de alto nivel y generar un código de máquina abstracta en una arquitectura de conjunto de instrucciones virtuales (código ISA virtual) y/o un código 254 de bytes que especifica una interfaz de máquina virtual. El módulo compilador puede generar el código ISA virtual/código 254 de bytes sin realizar ningún procesamiento compilador de componente central y componente final complejo que vincule el código a una arquitectura o un sistema operativo específicos. El código ISA virtual/código 254 de bytes generado se puede distribuir a dispositivos móviles que tienen una amplia variedad de plataformas y entornos de ejecución, siempre que los dispositivos móviles incluyan software de virtualización que admita la ISA virtual utilizada para generar el código ISA virtual/código 254 de bytes.

[24] Un dispositivo informático que tiene instalado software de virtualización puede recibir el código 254 de distribución y almacenar el código recibido en la memoria. El software de virtualización puede incluir un intérprete/compilador para traducir las instrucciones de ISA virtual como las instrucciones de ISA reales utilizadas por el hardware subyacente. Un cargador de máquina virtual puede cargar una imagen 254 de memoria virtual del código recibido y pasar el código recibido al intérprete/compilador de máquina virtual, que puede interpretar la imagen de memoria virtual y/o compilar el código de ISA virtual contenido en ella, para generar un código de máquina 258 anfitriona para ejecución directa en la plataforma principal.

[25] La compilación del código se puede realizar en dos etapas, una antes de la distribución y otra después de la distribución. Esto permite que las aplicaciones de software se puedan transferir fácilmente a cualquier dispositivo informático que tenga software de virtualización que admita la ISA virtual utilizada por el primer compilador, independientemente del hardware subyacente y la interfaz de sistema operativo del dispositivo. Además, el compilador de máquina virtual puede estar configurado para procesar el código de forma considerablemente más rápida que el compilador completo, porque el compilador de máquina virtual solo necesita convertir la ISA virtual en las instrucciones de máquina anfitriona.

[26] Por lo tanto, en el procedimiento 200 ilustrado en la figura 2A, el código se distribuye como código de máquina/objeto (por ejemplo, ejecutable ARM), mientras que en el procedimiento 250 de aspecto ilustrado en la figura 2B, el código se distribuye como código de máquina abstracta/código de bytes (por ejemplo, código de bytes Dalvik). En cualquier caso, un optimizador estático puede optimizar el código antes de la distribución (por ejemplo, durante la compilación). Sin embargo, las características específicas del hardware en el que se ejecutará el código no están disponibles para el optimizador estático, y en general no pueden conocerse hasta que se está en tiempo de ejecución. Por este motivo, los optimizadores estáticos en general utilizan rutinas de optimización genéricas que optimizan el código para que se ejecute de manera más eficiente (es decir, más rápida) en una amplia diversidad de plataformas y entornos de ejecución. Estas rutinas genéricas de optimización no pueden tomar en consideración las características específicas del hardware individual en el que se ejecuta el código, tales como las características de consumo de energía de un procesador específico. Los diversos aspectos utilizan técnicas de virtualización para optimizar el código en tiempo de ejecución, utilizando las características específicas del hardware en el que se ejecutará el código para reducir la cantidad de energía requerida para ejecutar el código.

[27] Las figuras 3A y 3B ilustran los componentes lógicos de un sistema informático típico que implementa una máquina virtual. Como se ha analizado anteriormente, las máquinas virtuales permiten que unos programas de aplicación vinculados a una ISA específica se ejecuten en hardware implementando una arquitectura de conjunto de instrucciones diferente. Estas máquinas virtuales se pueden clasificar en dos categorías generales: máquinas virtuales de sistema y máquinas virtuales de proceso. Las máquinas virtuales de sistema permiten compartir el hardware físico subyacente entre diferentes procesos o aplicaciones, mientras que las máquinas virtuales de proceso admiten un único proceso o aplicación.

[28] La figura 3A es un diagrama de arquitectura en capas que ilustra unas capas lógicas de un dispositivo informático 300 que implementa una máquina 310 virtual de proceso. El sistema informático 300 puede incluir componentes de hardware 308 (por ejemplo, hardware de ejecución, memoria, dispositivos de E/S, etc.) y componentes de software que incluyen un módulo 304 de virtualización, un sistema operativo 306 y un módulo 302 de aplicación.

[29] Como se ha analizado anteriormente con referencia a la figura 1, unos componentes de hardware solo son visibles para los programas de aplicación a través del sistema operativo, y las ABI y API definen con eficacia las características de hardware disponibles para el programa de aplicación. El módulo 304 de software de virtualización realiza operaciones lógicas en el nivel de ABI/API y emula llamadas de sistema operativo y/o llamadas de biblioteca, de tal forma que el proceso 302 de aplicación se comunica con el módulo 304 de software de virtualización de la misma manera que se comunicaría con componentes de hardware (es decir, a través de llamadas de sistema/biblioteca). De esta manera, el proceso 302 de aplicación percibe la combinación del módulo 304 de virtualización, el sistema operativo 306 y el hardware 308 como una única máquina, tal como la máquina 310 virtual de proceso ilustrada en la figura 3A.

[30] Como se ha mencionado anteriormente, la máquina 310 virtual de proceso existe únicamente para admitir un único proceso 302 de aplicación. La máquina 310 virtual de proceso se crea con el proceso 302 y termina cuando el proceso 302 finaliza la ejecución. El proceso 302 que se ejecuta en la máquina virtual 310 se denomina «huésped» y la plataforma subyacente se denomina «anfitrión». El software 304 de virtualización que implementa la máquina virtual de proceso se denomina típicamente software de tiempo de ejecución (o simplemente «tiempo de ejecución»).

[31] En un ejemplo, Dalvik es una máquina virtual (VM) de proceso en el sistema operativo Android de Google™. El sistema operativo Android convierte el código de bytes Dalvik en código objeto ejecutable ARM antes de la ejecución. Sin embargo, las características de consumo de energía del hardware no se toman en consideración al generar el código objeto ARM. Además, dado que la máquina virtual 310 de proceso se crea con el proceso 302 y termina cuando el proceso 302 finaliza, la información sobre la ejecución del proceso 302 no se puede utilizar para optimizar otros procesos concurrentes.

[32] La figura 3B es un diagrama de arquitectura en capas que ilustra las capas lógicas de un dispositivo informático 350 que implementa una máquina virtual 360 de sistema. El sistema informático puede incluir componentes de hardware 358 (por ejemplo, hardware de ejecución, memoria, dispositivos de E/S, etc.) y componentes de software que incluyen un módulo 356 de virtualización, un sistema operativo 354 y un módulo 352 de programas de aplicación. El software que se ejecuta en la parte superior del módulo 356 de virtualización se denomina software «huésped» y la plataforma subyacente que admite el módulo de virtualización se denomina hardware «anfitrión».

[33] El módulo 356 de software de virtualización puede estar situado lógicamente entre el hardware anfitrión y el software huésped. El software de virtualización puede ejecutarse en el hardware real (nativo) o en la parte superior de un sistema operativo (hospedado), y típicamente se denomina «hipervisor» o monitor de máquina virtual (VMM). El hipervisor proporciona al software huésped recursos de hardware virtualizados y/o emula la ISA de hardware, de tal forma que el software huésped puede ejecutar una ISA diferente a la ISA implementada en el hardware anfitrión.

[34] A diferencia de las máquinas virtuales de proceso, una máquina virtual 360 de sistema proporciona un entorno completo en el que los múltiples sistemas operativos pueden coexistir. Del mismo modo, la plataforma de hardware anfitrión puede estar configurada para admitir simultáneamente múltiples entornos de sistema operativo huésped aislados. El aislamiento entre los sistemas operativos que se ejecutan simultáneamente agrega un nivel de seguridad al sistema. Por ejemplo, si se infringe la seguridad en un sistema operativo huésped, o si un sistema operativo huésped experimenta un fallo, el software que se ejecuta en otros sistemas huésped no se ve afectado por la infracción o el fallo. Además, la máquina virtual de sistema puede utilizar información obtenida de la ejecución de un proceso para optimizar otros procesos simultáneos.

[35] Como se ha mencionado anteriormente, el software de virtualización puede ejecutarse en el hardware real (nativo) o en la parte superior de un sistema operativo (hospedado). En configuraciones nativas, el software de virtualización se ejecuta en el modo de privilegio más alto disponible y los sistemas operativos huésped se ejecutan con privilegios reducidos, de tal forma que el software de virtualización puede interceptar y emular todas las acciones de sistema operativo huésped que normalmente accederían o manipularían los recursos de hardware. En las configuraciones hospedadas, el software de virtualización se ejecuta en la parte superior de un sistema operativo anfitrión existente y puede depender del sistema operativo anfitrión para proporcionar controladores de dispositivos y otros servicios de nivel inferior. En ambos casos, cada uno de los sistemas operativos huésped (por ejemplo, el sistema operativo 354) se comunica con el módulo 356 de software de virtualización de la misma manera que se comunicaría con el hardware físico 358. Esto permite que cada sistema operativo huésped (por ejemplo, el sistema operativo 354) perciba la combinación del módulo 356 de

virtualización y el hardware 358 como una sola máquina virtual, tal como la máquina virtual 360 de sistema ilustrada en la figura 3B.

[36] El hardware huésped puede emularse mediante interpretación, traducción binaria dinámica (DBT) o una combinación de estas. En configuraciones de interpretación, la máquina virtual incluye un intérprete que obtiene, decodifica y emula la ejecución de las instrucciones de huésped individuales. En las configuraciones de traducción binaria dinámica, la máquina virtual incluye un traductor binario dinámico que convierte instrucciones de huésped escritas en una primera ISA en instrucciones de anfitrión escritas en una segunda ISA. El traductor binario dinámico puede traducir las instrucciones de huésped en grupos o bloques (en lugar de instrucción en instrucción), que pueden guardarse en una memoria caché de software y reutilizarse. Esto permite realizar ejecuciones repetidas de instrucciones traducidas previamente sin retraducir el código, lo que mejora la eficiencia y reduce la carga de procesamiento.

[37] Como se ha analizado anteriormente, los traductores binarios dinámicos convierten las instrucciones de huésped escritas en una primera ISA (por ejemplo, ISA virtual, SPARC, etc.) en instrucciones de anfitrión escritas en una segunda ISA (por ejemplo, ARM, etc.). En los diversos aspectos, el traductor 414 binario dinámico puede estar configurado para convertir instrucciones de huésped escritas en una primera ISA (por ejemplo, ARM) en instrucciones de anfitrión escritas en la misma ISA (por ejemplo, ARM). En los diversos aspectos, como parte de este proceso de traducción, el traductor 414 binario dinámico puede realizar uno o más procedimientos de optimización de código para optimizar el rendimiento del código binario basándose en un modelo de la cantidad de energía consumida en tiempo de ejecución por un elemento específico de hardware en la realización de un segmento o secuencia de código particular. En este procesamiento, el traductor 414 binario dinámico puede determinar las operaciones de máquina que consumen la mayor cantidad de energía (por ejemplo, operaciones de multiplicación), determinar si hay operaciones de máquina alternativas que logran los mismos resultados (por ejemplo, cambio y adición) y realizar las operaciones de traducción de tal forma que el código traducido se optimiza para el consumo de energía (por ejemplo, reemplazando todas las operaciones de multiplicación por operaciones de cambio y agregación, etc.). En un aspecto, el traductor 414 binario dinámico puede optimizar el código realizando una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto escrito en una primera ISA (por ejemplo, ARM) como un código objeto escrito en la misma ISA (por ejemplo, ARM).

[38] La figura 4 es un diagrama de componentes que ilustra los componentes lógicos de un dispositivo informático 400 que implementa una máquina 402 virtual de sistema configurada para optimizar el comportamiento de energía de las aplicaciones 404 en tiempo de ejecución de acuerdo con los diversos aspectos. La máquina 402 virtual de sistema puede funcionar en el nivel de hipervisor, debajo del sistema operativo 406, e incluir uno o más modelos del consumo 410 de energía. La máquina 402 virtual de sistema también puede incluir un generador de código dinámico/compilador 412 de tiempo de ejecución configurado para generar y/o seleccionar uno o más procedimientos de optimización adaptados específicamente a las características de ejecución de un programa de aplicación específico. La máquina virtual de sistema también puede incluir un traductor 414 binario dinámico configurado para traducir el código objeto como un código objeto de energía optimizada, adaptando unos programas de aplicación al hardware exacto en el que se ejecutan las aplicaciones. En un aspecto, el generador de código/compilador 412 de tiempo de ejecución y el traductor 414 binario dinámico pueden implementarse como una sola unidad compiladora 416. En un aspecto, la máquina virtual de sistema puede estar configurada de tal forma que la unidad compiladora 416 actúa sobre un código objeto (en lugar de un código fuente) y genera un nuevo código objeto optimizado para eficiencia energética (en lugar de para rendimiento/velocidad).

[39] Las características de consumo de energía de unos procesadores pueden depender del tipo de hardware y de cómo el hardware procesa el código objeto específico. Por ejemplo, la cantidad de energía consumida para realizar una tarea de procesamiento dada puede variar de un tipo de dispositivo a otro, dependiendo de sus arquitecturas. Además, las características de consumo de energía del mismo tipo de procesador pueden variar de un lote a otro y de un chip a otro, en algunos casos hasta un treinta por ciento. Debido a estas variaciones, los desarrolladores de aplicaciones no pueden escribir código fuente optimizado para un dispositivo particular o un conjunto de dispositivos particular, ya que dicha información en general no está disponible hasta que se está en tiempo de ejecución.

[40] En un aspecto, el compilador de máquina virtual 402 de sistema puede estar configurado para optimizar el código en tiempo de ejecución, basándose en las características de consumo de energía real del hardware. La máquina virtual 402 puede funcionar en la capa de la máquina (en lugar de la capa de lenguaje), permitiendo además que el traductor 414 binario dinámico realice procedimientos de optimización que optimizan la eficiencia energética, en lugar de o además de optimizar la velocidad de procesamiento. En un aspecto, la unidad compiladora 416 puede utilizar una o más rutinas de optimización de compilador para mejorar la utilización de energía basándose en el rendimiento en tiempo de ejecución de una ejecución de código.

[41] En un aspecto, el traductor 414 binario dinámico puede utilizar información de perfil recopilada durante la interpretación y/o traducción para optimizar el código binario durante una ejecución. Además, el traductor 414

binario dinámico puede utilizar información de consumo de energía recopilada en tiempo de ejecución para modificar los procedimientos de optimización, que el traductor 414 binario dinámico y/o el generador 412 de código pueden utilizar para optimizar futuras traducciones y/o generar versiones reoptimizadas de la traducción actual. Mientras el traductor 414 binario dinámico extrae datos de generación de perfiles, el generador 412 de código puede generar etiquetas que permiten que la máquina virtual asocie los datos de generación de perfiles con un fragmento particular de código. En un aspecto, el traductor 414 binario dinámico puede utilizar parámetros de generación de perfiles y etiquetas generadas para medir la cantidad de energía requerida para ejecutar un fragmento específico de código en un elemento específico de hardware, y ajustar los procedimientos de optimización basándose en las características de energía real del hardware específico.

[42] La máquina virtual 402 puede mantener un inventario de núcleos de procesador y/o procesadores disponibles, que pueden incluir uno o más sistemas en chips (SOC). El término «sistema en chip» (SOC) se utiliza para referirse a un único chip de circuito integrado (IC) que contiene múltiples recursos y procesadores integrados en un único sustrato. Los modelos 410 de consumo de energía pueden incluir una estructura de datos (por ejemplo, lista, matriz, tabla, mapa, etc.) utilizada para almacenar información para seguir cada unidad de código objeto que se procesa en los diferentes núcleos/procesadores, y la cantidad de energía necesaria para procesar cada unidad de código objeto en un núcleo/procesador específico o identificar procesadores alternativos de mayor eficiencia energética. El generador de código puede realizar unas operaciones de traducción de binario a binario basándose en los modelos de consumo de energía para generar un código de energía optimizada.

[43] En diversos aspectos, unos modelos 410 de consumo de energía pueden ser suministrados por los autores de la máquina virtual o los autores de los programas de aplicación, y/o construidos por la máquina virtual 402 en tiempo de ejecución utilizando un procedimiento de aprendizaje automático. El procedimiento de aprendizaje automático puede generarse y/o actualizarse a medida que el dispositivo móvil se ejecuta y ejecuta el código objeto. Por ejemplo, la máquina virtual 402 puede estar configurada para construir los modelos 410 de consumo de energía basándose en información recopilada de ejecuciones previas de código similar, mediante técnicas de aprendizaje automático y datos empíricos.

[44] La figura 5 ilustra un procedimiento 500 de aspecto de optimización de código objeto en tiempo de ejecución para un consumo de energía mejorado. Como se ha mencionado anteriormente, los programas de aplicación de dispositivos móviles se distribuyen típicamente como archivos binarios compilados que están vinculados a una arquitectura de conjunto de instrucciones (ISA) específica y dependen de una interfaz de sistema operativo (OSI) específica. En el bloque 502, el dispositivo móvil puede recibir un archivo de código binario compilado, en una ISA o un código de bytes virtual (por ejemplo, Dalvik), o como un código objeto (por ejemplo, ejecutable ARM). El archivo binario compilado puede ser recibido por un sistema operativo de dispositivo móvil y proporcionado a una máquina virtual de sistema que funciona en la capa de máquina del dispositivo móvil. En el bloque 504, la máquina virtual puede compilar/traducir el código y/o generar un código objeto optimizado aplicando uno o más modelos de consumo de energía al código, que la máquina virtual puede utilizar para determinar cómo el hardware ejecutará el código objeto y/o para reconocer patrones o segmentos dentro en el código objeto que deberían traducirse como una secuencia de operaciones alternativa de energía optimizada. Por ejemplo, la máquina virtual puede utilizar los modelos de consumo de energía para identificar operadores en el código (por ejemplo, operaciones de multiplicación) que consumen una cantidad excesiva de energía, identificar operaciones alternativas adecuadas (por ejemplo, operaciones de cambio y agregación) a los operadores identificados, y realizar una traducción de binario a binario del código objeto para generar un código objeto optimizado. El código traducido puede guardarse a medida que se genera de tal forma que la próxima ejecución de la aplicación no requiera repetir los procesos de análisis y optimización del código. En el bloque 506, el código optimizado puede cargarse en el hardware y ejecutarse en un procesador/núcleo.

[45] En un aspecto, la máquina virtual puede actualizar continuamente los modelos de consumo de energía y regenerar el código objeto basándose en unas mediciones de energía real consumida por ejecuciones previas de código objeto optimizado. De esta manera, el rendimiento real de los procesadores de dispositivo móvil puede utilizarse para optimizar el código objeto, en lugar de depender de un modelo fijo que puede no reflejar la variabilidad de un lote a otro en un rendimiento de procesador.

[46] La figura 6 es un diagrama de flujo de componentes/procesos que ilustra ejemplos de componentes lógicos y flujos de datos en un dispositivo informático configurado para realizar un procedimiento 600 de aspecto de actualización continua de los modelos de consumo de energía y regeneración del código objeto. Como se ha mencionado anteriormente, la máquina virtual puede implementarse en un dispositivo informático móvil que tiene múltiples núcleos y/o procesadores, que pueden incluir uno o más sistemas en chips (SOC). En el ejemplo ilustrado en la figura 6, el dispositivo informático móvil incluye una unidad 602 de procesador central, un núcleo 604 QDSP Hexagon y una unidad 606 de procesamiento de gráficos (GPU). Cada uno de estos procesadores puede estar instrumentado para medir la energía consumida durante la ejecución del código objeto generado.

[47] Una unidad 618 de compilación puede generar fragmentos de representación intermedios de compilador y enviar los fragmentos de código a un selector 616 de destino. El selector 616 de destino puede seguir la

disponibilidad de los procesadores y seleccionar el procesador más adecuado para ejecutar un segmento de código (por ejemplo, el procesador menos utilizado, el procesador que requiere la menor cantidad de energía, etc.). El selector 616 de destino puede enviar un fragmento de código a un módulo 608, 610, 612 generador de código, que puede recibir el fragmento de código, y realizar una traducción de secuencia de instrucciones en secuencia de instrucciones del código para optimizar el código para el núcleo/procesador seleccionado 602, 604, 606. A continuación, el código optimizado puede cargarse en el núcleo/procesador seleccionado 602, 604, 606 para su ejecución.

[48] Durante la ejecución, se puede recopilar información sobre la cantidad de energía consumida por cada procesador al procesar cada fragmento de código. Se pueden incluir, en los fragmentos de código ejecutados, etiquetas y/o anotaciones con la información de consumo de energía recopilada. La información de consumo de energía medida puede enviarse a un módulo 614 de predicción de rendimiento, que compara la información de consumo medida con un modelo de predicción de rendimiento. Unos resultados de la comparación entre el modelo de consumo de energía real y el modelo de rendimiento previsto se pueden retroalimentar al selector 616 de destino. El selector 616 de destino puede utilizar los resultados de comparación para actualizar los modelos de consumo de energía y los procedimientos de optimización, de tal forma que se mejoran las características de consumo de energía de fragmentos de códigos de objeto generados subsiguientemente.

[49] Diversos aspectos pueden utilizar la suma del ahorro de energía y el coste de energía de realizar las operaciones de compilación/traducción y optimización para determinar una función de energía. La función de energía puede utilizarse para determinar el ahorro de energía neto asociado con cada modelo de energía y/o para determinar si se deben realizar las optimizaciones. Por ejemplo, la función de energía puede utilizarse para determinar si la cantidad de energía requerida para realizar una optimización excede la cantidad de energía ahorrada por la optimización, en cuyo caso el rendimiento de la optimización puede cancelarse o retrasarse.

[50] En un aspecto, la máquina virtual de sistema puede calcular un ahorro de energía basándose en unos valores de consumo de energía recopilados en tiempo de ejecución. La máquina virtual de sistema puede actualizar periódicamente los modelos de consumo de energía y regenerar los fragmentos de código basándose en una combinación de características de consumo de energía medidas y un ahorro de energía calculado. El ahorro de energía se puede calcular mediante una función polinómica lineal, o como la cantidad de energía ahorrada durante un plazo de tiempo calculado, compensado por la cantidad de trabajo requerido para realizar las operaciones de compilación/traducción y optimización.

[51] En un aspecto, el hardware puede estar instrumentado con circuitos adicionales para medir las características de consumo de energía de ejecución de un código. La máquina virtual de sistema puede estar configurada para leer las mediciones realizadas por los circuitos adicionales, y para utilizar las características de consumo de energía medidas para realizar procedimientos de optimización.

[52] La figura 7 ilustra un procedimiento 700 de aspecto para actualizar los modelos de consumo de energía y reoptimizar continuamente el código objeto en tiempo de ejecución. En el bloque 702, la unidad de compilación de máquina virtual puede generar unidades de código objeto nativo para su ejecución en uno o más procesadores. Se pueden incluir etiquetas y/o anotaciones en las unidades de código objeto cuando estas se optimizan y/o transforman. La máquina virtual puede utilizar las etiquetas/anotaciones para seguir la cantidad de energía consumida por cada unidad de código. En el bloque 704, la máquina virtual puede ejecutar una o más de las unidades de código generadas en un procesador instrumentado con unos circuitos adicionales para medir las características de consumo de energía de ejecución de las unidades de código. En el bloque 706, la información de consumo de energía puede recopilarse desde el procesador. En el bloque 708, la información de consumo de energía medido puede compararse con un modelo de predicción de rendimiento, y los resultados de la comparación entre el consumo de energía real y el rendimiento previsto pueden almacenarse en una memoria. En el bloque 710, los resultados almacenados se pueden utilizar para actualizar los modelos de rendimiento y/o modelos de energía utilizados por la unidad de compilación de máquina virtual para generar unidades de código objeto nativo. En el bloque 712, la unidad de compilación de máquina virtual puede regenerar un código objeto optimizado previamente o segmentos de código que aún no se han ejecutado, y ejecutar el código regenerado en el bloque 704. De esta manera, las comparaciones entre el consumo de energía real y el rendimiento previsto pueden retroalimentarse a la unidad de compilación, de tal forma que pueden identificarse y utilizarse mejores procedimientos de optimización para subsiguientes optimizaciones de código objeto.

[53] Las comparaciones también se pueden utilizar para identificar procedimientos de optimización previos que han dado como resultado un consumo de energía mayor, en lugar de menor. Por ejemplo, la máquina virtual puede ejecutar una unidad de código optimizada en un procesador instrumentado con los circuito adicionales para medir el consumo de energía, medir la cantidad real de energía consumida por el código optimizado y, si la cantidad real de energía consumida excede la cantidad de energía consumida por el código original, retroceder y realizar transformaciones similares en otras unidades de código. De esta manera, la máquina virtual puede estar configurada para aprender a lo largo del tiempo qué transformaciones binarias son efectivas en una unidad específica de hardware y qué transformaciones no lo son, y hacer ajustes a los modelos según sea necesario.

[54] En los diversos aspectos, se puede utilizar una diversidad de técnicas de retroalimentación y aprendizaje automático. Unos conjuntos de reglas de optimización de código se pueden cambiar o actualizar cuando los resultados medidos se apartan del modelo previsto. Se pueden realizar experimentos de procedimiento de alteración y prueba automáticos, tales como cambiar una regla de optimización, comparar el consumo de energía medido del código optimizado antes y después del cambio de la regla de optimización y seleccionar para el uso la regla de optimización que arroja los mejores resultados. En un aspecto, se puede comparar el rendimiento de energía de diferentes longitudes de código optimizado para reconocer patrones y permitir una mejor optimización.

[55] Los mecanismos de retroalimentación y aprendizaje presentan una serie de ventajas. Por ejemplo, el desarrollador del dispositivo móvil no está obligado a generar el modelo de consumo de energía específico del dispositivo, porque los modelos son generados automáticamente por el propio dispositivo móvil a través de aprendizaje automático, lo que simplifica el desarrollo del dispositivo. En otro ejemplo, los mecanismos de retroalimentación y aprendizaje permiten que los diversos aspectos se adapten a los cambios de hardware que se producen tras establecerse el diseño inicial (por ejemplo, agregación de nueva memoria, sustitución de un procesador, etc.) una vez que se ha diseñado el modelo. Los mecanismos de retroalimentación y aprendizaje también permiten que los diversos aspectos respondan mejor a la variabilidad de un lote a otro y de una línea a otra en unas características de consumo de energía de procesador, que puede variar hasta en un veinte por ciento. Por ejemplo, aunque algunos de los chips fabricados en una pastilla particular pueden beneficiarse de la optimización de un código objeto de una manera particular (por ejemplo, utilizando operaciones de cambio y agregación en lugar de operaciones de multiplicación), algunos pueden experimentar un mayor consumo de energía a partir de la misma optimización, debido a la variabilidad de un lote a otro y de una línea a otra. Los diversos aspectos pueden tener en cuenta dicha variabilidad optimizando el código basándose en las características individuales del chip/hardware.

[56] Tal como se ha mencionado anteriormente, diversos aspectos pueden utilizar la suma del ahorro de energía y el coste de energía de realizar las operaciones de compilación/traducción y optimización para determinar una función de energía, que se puede utilizar para determinar si se deben realizar las optimizaciones. Por ejemplo, la función de energía puede utilizarse para determinar si la cantidad de energía requerida para realizar la optimización excede la cantidad de energía ahorrada por la optimización. En un aspecto, los modelos asociados con procedimientos de optimización que requieren más energía para funcionar que la cantidad de energía conservada por el código optimizado pueden almacenarse en una memoria y realizarse cuando el dispositivo informático no está funcionando con energía de la batería.

[57] La figura 8 es un diagrama de flujo de procesos que ilustra un procedimiento 800 de aspecto para realizar las optimizaciones de código objeto cuando el dispositivo móvil está enchufado a una fuente de energía y no está funcionando con energía de batería. En el bloque 802, la máquina virtual puede generar las unidades de código objeto optimizado y comenzar la ejecución, de acuerdo con cualquiera de los aspectos analizados anteriormente. En el bloque 804, la máquina virtual puede determinar que hay más optimizaciones disponibles, y almacenar en una memoria información relativa a las optimizaciones disponibles. En el bloque 806, la máquina virtual puede continuar la ejecución del código sin realizar ninguna de las optimizaciones identificadas. En el bloque 808 de determinación, el dispositivo informático puede realizar unas operaciones para determinar si hay una nueva fuente de energía (por ejemplo, una conexión cableada de alimentación) disponible para el dispositivo. Si no hay ninguna nueva fuente de energía disponible (bloque 808 de determinación = «No»), en el bloque 806 la máquina virtual puede continuar ejecutando el código sin realizar ninguna de las optimizaciones identificadas. Si, por otro lado, el procesador determina que hay disponible una nueva fuente de energía (bloque 808 de determinación = «Sí»), en el bloque 810 la máquina virtual puede recuperar la información de optimización almacenada de la memoria y realizar las optimizaciones (actualizar los modelos de energía, regenerar código objeto, etc.) y almacenar el código optimizado en la memoria. De esta manera, los procedimientos de optimización pueden realizarse solo si los costes de energía asociados con la realización de las optimizaciones no exceden las ganancias de energía resultantes de las optimizaciones.

[58] En un aspecto, el código objeto optimizado que resulta de las optimizaciones puede guardarse en memoria y utilizarse para subsiguientes ejecuciones del código. Las optimizaciones pueden realizarse en conjunción con un modelo de consumo de energía que es específico para el hardware particular, que puede ser proporcionado desde fábrica y/o aprendido por el dispositivo móvil durante la ejecución. De esta manera, los diversos procedimientos de optimización analizados anteriormente pueden realizarse en tiempo de ejecución, antes del tiempo de ejecución, cuando se carga el código o la primera vez que se ejecuta el proceso. Los diversos procedimientos de optimización pueden formar parte del proceso de generación de código de tiempo de ejecución o parte del proceso de generación de código estático.

[59] Debe entenderse que, en los diversos aspectos, realizar optimizaciones cuando se está conectado a la energía no es exclusivo para la realización de la optimización en tiempo de ejecución. Por ejemplo, la máquina virtual de sistema puede realizar optimizaciones según sea necesario (por ejemplo, durante la ejecución) o con anticipación (por ejemplo, cuando está conectada a la energía e inactiva).

[60] También debe entenderse que las decisiones sobre cuándo aplicar la optimización pueden ser independientes de las decisiones sobre cuándo recopilar datos de rendimiento. Los diversos aspectos pueden recopilar datos de consumo de energía durante una ejecución del código y no actuar sobre los datos recopilados hasta que se cumple una condición (por ejemplo, el dispositivo está conectado a la energía).

[61] Unos dispositivos móviles 900 típicos adecuados para su uso con los diversos aspectos tendrán en común los componentes ilustrados en la figura 9. Por ejemplo, un dispositivo móvil 900 a modo de ejemplo puede incluir un procesador 902 acoplado a una memoria interna 901, a una pantalla 904 y a un altavoz 964. Adicionalmente, el dispositivo móvil puede tener una antena 924 para enviar y recibir radiación electromagnética acoplada al procesador 902. En algunos aspectos, el dispositivo móvil 900 puede incluir uno o más procesadores 905, 924 especializados o de uso general que pueden incluir sistemas en chips. Los dispositivos móviles típicamente incluyen también un teclado o teclado en miniatura y botones de selección de menú o interruptores basculantes para recibir entradas de usuario.

[62] La figura 10 ilustra otro dispositivo móvil 1000 a modo de ejemplo adecuado para su uso con los diversos aspectos. Por ejemplo, el dispositivo móvil 1000 puede incluir un procesador 1002 acoplado a la memoria interna 1001, y una pantalla 1009. Adicionalmente, el dispositivo móvil puede tener un puerto 1005 de comunicación para enviar y recibir información. El dispositivo móvil 1000 también puede incluir un teclado 1008 y botones 1007 de selección para recibir entradas de usuario.

[63] Los procesadores 902, 905, 924, 1002 pueden ser cualquier microprocesador programable, microordenador o uno o más chips de procesador múltiple que se pueden configurar mediante instrucciones de software ejecutables por procesador (aplicaciones) para realizar una diversidad de funciones, incluidas las funciones de los diversos aspectos descritos en el presente documento. Típicamente, unas aplicaciones de software e instrucciones ejecutables por procesador pueden almacenarse en la memoria interna 901, 1001 antes de que se acceda a ellas y se carguen en los procesadores 902, 905, 924, 1002. En algunos dispositivos móviles, los procesadores 902, 905, 924, 1002 pueden incluir suficiente memoria interna para almacenar las instrucciones de software de aplicación. En algunos dispositivos móviles, la memoria segura puede estar en un chip de memoria separado acoplado al procesador 902, 905, 924, 1002. En muchos dispositivos móviles, la memoria interna 901, 1001 puede ser una memoria volátil o no volátil, tal como memoria flash, o una mezcla de ambas. Para los propósitos de esta descripción, una referencia general a una memoria se refiere a toda la memoria accesible por los procesadores 902, 905, 924, 1002 incluida memoria interna, memoria extraíble enchufada al dispositivo móvil y memoria en los procesadores.

[64] Las descripciones de procedimiento anteriores y los diagramas de flujo de procesos se proporcionan simplemente como ejemplos ilustrativos y no pretenden requerir o implicar que las etapas de los diversos aspectos deben realizarse en el orden presentado. Como apreciará un experto en la técnica, el orden de las etapas en los aspectos anteriores pueden ser cualquier orden. Palabras tales como «a continuación», «entonces», «seguidamente», etc. no pretenden limitar el orden de las etapas, sino que se utilizan simplemente para guiar al lector a través de la descripción de los procedimientos. Además, ninguna referencia en singular a elementos de las reivindicaciones, por ejemplo, mediante los artículos «un», «una», «el» o «la» debe interpretarse como limitación del elemento al singular.

[65] Los diversos bloques lógicos, módulos, circuitos y etapas de algoritmo ilustrativos, descritos en relación con los aspectos divulgados en el presente documento, pueden implementarse como hardware electrónico, software informático o combinaciones de ambos. Para ilustrar claramente esta intercambiabilidad de hardware y software, anteriormente se han descrito diversos componentes, bloques, módulos, circuitos y etapas ilustrativos en general desde el punto de vista de su funcionalidad. Si dicha funcionalidad se implementa como hardware o software depende de la aplicación y las restricciones de diseño particulares impuestas al sistema global. Los expertos en la materia pueden implementar la funcionalidad descrita de formas diversas para cada aplicación particular, pero no debería interpretarse que dichas decisiones de implementación suponen apartarse del alcance de la presente invención.

[66] El hardware utilizado para implementar las diversas lógicas, bloques lógicos, módulos y circuitos ilustrativos descritos en relación con los aspectos divulgados en el presente documento pueden implementarse o realizarse con un procesador de uso general, un procesador de señales digitales (DSP), un DSP en un chip receptor de radiodifusión multimedia, un circuito integrado específico de aplicación (ASIC), una matriz de puertos programables *in situ* (FPGA) u otro dispositivo de lógica programable, lógica de puerta o transistor discretos, componentes de hardware discretos, o cualquier combinación de los mismos diseñada para realizar las funciones descritas en el presente documento. Un procesador de uso general puede ser un microprocesador pero, de forma alternativa, el procesador puede ser cualquier procesador, controlador, microcontrolador o máquina de estados convencional. Un procesador también puede implementarse como una combinación de dispositivos informáticos, por ejemplo, una combinación de un DSP y un microprocesador, una pluralidad de microprocesadores, uno o más microprocesadores junto con un núcleo de DSP o cualquier otra configuración de este tipo. De forma alternativa, unos circuitos que son específicos de una función determinada pueden realizar algunas etapas o procedimientos.

[67] En uno o más aspectos a modo de ejemplo, las funciones descritas pueden implementarse en hardware, software, firmware o cualquier combinación de estos. Si se implementan en software, las funciones se pueden almacenar en, o transmitir a través de, un medio legible por ordenador, como una o más instrucciones o códigos. Las etapas de un procedimiento o algoritmo divulgadas en el presente documento pueden implementarse en un módulo de software ejecutable por procesador que puede residir en un medio legible por ordenador. Los medios legibles por ordenador incluyen tanto medios de almacenamiento informático como medios de comunicación, incluido cualquier medio que facilita la transferencia de un programa informático de un lugar a otro. Un medio de almacenamiento puede ser cualquier medio disponible al que pueda accederse mediante un ordenador. A modo de ejemplo, y no de limitación, dichos medios legibles por ordenador pueden comprender RAM, ROM, EEPROM, CD-ROM u otros dispositivos de almacenamiento en disco óptico, almacenamiento en disco magnético u otros dispositivos de almacenamiento magnético, o cualquier otro medio que pueda utilizarse para transportar o almacenar un código de programa deseado en forma de instrucciones o estructuras de datos y al que pueda accederse mediante un ordenador. Asimismo, cualquier conexión recibe debidamente la denominación de medio legible por ordenador. Por ejemplo, si el software se transmite desde un sitio web, servidor u otra fuente remota, mediante un cable coaxial, un cable de fibra óptica, un par trenzado, una línea de abonado digital (DSL) o unas tecnologías inalámbricas tales como infrarrojos, radio y microondas, entonces el cable coaxial, el cable de fibra óptica, el par trenzado, la DSL o las tecnologías inalámbricas, tales como infrarrojos, radio y microondas, se incluyen en la definición de medio. El término disco, utilizado en el presente documento, incluye un disco compacto (CD), un disco láser, un disco óptico, un disco versátil digital (DVD), un disquete y un disco Blu-ray. Las combinaciones de los anteriores también deben incluirse dentro del alcance de los medios legibles por ordenador. Adicionalmente, las operaciones de un procedimiento o algoritmo pueden residir como un código o cualquier combinación o conjunto de códigos y/o instrucciones en un medio legible por máquina y/o un medio legible por ordenador, que pueden incorporarse a un producto de programa informático.

[68] La anterior descripción de los modos de realización divulgados se proporciona para permitir que cualquier experto en la materia realice o use la presente invención. Diversas modificaciones de estos modos de realización resultarán muy evidentes a los expertos en la materia, y los principios genéricos definidos en el presente documento pueden aplicarse a otros modos de realización sin apartarse del alcance de la presente invención. Por lo tanto, la presente invención no pretende limitarse a los modos de realización mostrados en el presente documento, sino que se le ha de conceder el alcance más amplio conforme a las reivindicaciones adjuntas.

[69] A continuación se describen otros ejemplos para facilitar la comprensión de la presente invención.

[70] En un primer ejemplo adicional, se describe un procedimiento para optimizar un código objeto para ahorrar energía durante la ejecución en un dispositivo informático, comprendiendo el procedimiento recibir un código objeto binario compilado en un software de sistema de un dispositivo informático; analizar el código objeto recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que se pueden optimizar para ahorrar energía; realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada; y ejecutar el código objeto de energía optimizada en un procesador del dispositivo informático. Además, el software de sistema que recibe el código objeto binario compilado puede ser uno de una máquina virtual de sistema o un hipervisor. Asimismo, el software de sistema que recibe el código objeto binario compilado puede ser un sistema operativo. Además, realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada puede comprender traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones. Asimismo, la primera arquitectura de conjunto de instrucciones puede ser la misma arquitectura de conjunto de instrucciones que la segunda arquitectura de conjunto de instrucciones. Además, analizar el código objeto recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que pueden optimizarse para ahorrar energía puede comprender determinar si hay operaciones alternativas que logran los mismos resultados que las operaciones de código objeto identificadas, y en el que realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada puede comprender reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas. El procedimiento puede comprender además detectar una conexión a una nueva fuente de energía, en el que realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada tiene lugar cuando se detecta una conexión a la nueva fuente de energía. Asimismo, analizar el código objeto recibido puede comprender utilizar un modelo de consumo de energía para identificar segmentos de código objeto que se pueden optimizar para eficiencia energética. El procedimiento puede comprender además medir una cantidad de energía consumida en la ejecución de segmentos de código objeto de energía optimizada; comparar la cantidad medida de energía consumida con unas predicciones del modelo de consumo de energía; y modificar el modelo de consumo de energía basándose en el resultado de la comparación.

[71] En otro ejemplo adicional, se describe un dispositivo informático configurado para optimizar un código objeto durante una ejecución para un ahorro de energía mejorado, que comprende: medios para recibir un código objeto binario compilado en software de sistema; medios para analizar el código objeto recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que pueden optimizarse para ahorrar energía; medios para realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada; y medios para ejecutar el código objeto de energía optimizada en un procesador del dispositivo informático. Asimismo, unos medios para recibir un código objeto binario compilado en software de sistema pueden comprender medios que reciben el código objeto binario compilado en uno de una máquina virtual de sistema o un hipervisor. Además, unos medios para recibir código objeto binario compilado en software de sistema pueden comprender medios que reciben el código objeto binario compilado en un sistema operativo. Asimismo, unos medios para realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada pueden comprender medios para traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones. Además, unos medios para traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones pueden comprender medios para traducir la primera arquitectura de conjunto de instrucciones como una arquitectura de conjuntos de instrucciones que es la misma que la segunda arquitectura de conjuntos de instrucciones. Asimismo, el dispositivo informático puede comprender además medios para analizar el código objeto recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que pueden optimizarse para ahorrar energía comprende medios para determinar si hay operaciones alternativas que logran los mismos resultados que las operaciones del código objeto identificadas; y medios para realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada que comprende medios para reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas. El dispositivo informático puede comprender además medios para detectar una conexión a una nueva fuente de energía, en el que unos medios para realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada pueden comprender medios para traducir el código recibido para generar un código objeto de energía optimizada cuando se detecta una conexión a la nueva fuente de energía. En el que unos medios para analizar el código objeto recibido pueden comprender medios para utilizar un modelo de consumo de energía para identificar segmentos de código objeto que pueden optimizarse para eficiencia energética. Asimismo, el dispositivo informático puede comprender además medios para medir una cantidad de energía consumida en la ejecución de segmentos de código objeto de energía optimizada; medios para comparar la cantidad medida de energía consumida con unas predicciones del modelo de consumo de energía; y medios para modificar el modelo de consumo de energía basándose en un resultado de la comparación.

[72] En todavía un ejemplo más, se describe un dispositivo informático que comprende una memoria; y uno o más procesadores acoplados a la memoria, en el que el uno o más procesadores están configurados con instrucciones ejecutables por procesador de tal forma que el dispositivo informático realiza operaciones que comprenden recibir código objeto binario compilado en software de sistema; analizar el código objeto recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que se pueden optimizar para ahorrar energía; realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada; y ejecutar el código objeto de energía optimizada. Asimismo, el uno o más procesadores pueden estar configurados con instrucciones ejecutables por procesador para que el dispositivo informático realice unas operaciones de tal forma que recibir código objeto binario compilado en software de sistema comprende recibir el código objeto binario compilado en uno de una máquina virtual de sistema o un hipervisor. Además, el uno o más procesadores pueden estar configurados con instrucciones ejecutables por procesador para que el dispositivo informático realice unas operaciones de tal forma que recibir código objeto binario compilado en software de sistema comprende recibir el código objeto binario compilado en un sistema operativo. Asimismo, uno o más procesadores pueden estar configurados con instrucciones ejecutables por procesador para que el dispositivo informático realice unas operaciones de tal forma que realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada comprende traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones. Asimismo, el uno o más procesadores pueden estar configurados con instrucciones ejecutables por procesador para que el dispositivo informático realice unas operaciones de tal forma que la primera arquitectura de conjunto de instrucciones es la misma que la segunda arquitectura de conjunto de instrucciones. Además, el uno o más procesadores pueden estar configurados con instrucciones ejecutables por procesador para que el dispositivo informático realice unas operaciones de tal forma que analizar el código objeto recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que pueden optimizarse para ahorrar energía comprende determinar si hay operaciones alternativas que logran los mismos resultados que las operaciones de código objeto identificadas; y realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía

optimizada comprende reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas. Asimismo, el uno o más procesadores pueden estar configurados con instrucciones ejecutables por procesador para que el dispositivo informático realice unas operaciones que comprenden además detectar una conexión a una nueva fuente de energía; y el uno o más procesadores pueden estar configurados con instrucciones ejecutables por procesador para que el dispositivo informático realice unas operaciones de tal forma que realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada tiene lugar cuando se detecta una conexión a la nueva fuente de energía. Además, el uno o más procesadores pueden estar configurados con instrucciones ejecutables por procesador para que el dispositivo informático pueda realizar unas operaciones de tal forma que analizar el código objeto recibido comprende usar un modelo de consumo de energía para identificar segmentos de código objeto que pueden optimizarse para eficiencia energética. Asimismo, el uno o más procesadores pueden estar configurados con instrucciones ejecutables por procesador para que el dispositivo informático realice unas operaciones que comprenden además medir una cantidad de energía consumida en la ejecución de segmentos de código objeto de energía optimizada; comparar la cantidad medida de energía consumida con unas predicciones del modelo de consumo de energía; y modificar el modelo de consumo de energía basándose en el resultado de la comparación.

[73] En otro ejemplo más, se describe un medio de almacenamiento no transitorio legible por procesador que tiene almacenadas instrucciones de software ejecutables por procesador configuradas para hacer que un procesador realice unas operaciones para optimizar un código objeto para ahorrar energía durante la ejecución en un dispositivo informático, en el que las operaciones pueden comprender recibir un código objeto binario compilado en software de sistema; analizar el código objeto recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que se pueden optimizar para ahorrar energía; realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada; y ejecutar el código objeto de energía optimizada en un procesador del dispositivo informático. Asimismo, las instrucciones de software ejecutables por procesador almacenadas pueden estar configuradas para hacer que un procesador realice unas operaciones de tal forma que recibir un código objeto binario compilado en software de sistema comprende recibir el código objeto binario compilado en uno de una máquina virtual de sistema o un hipervisor. Además, las instrucciones de software ejecutables por procesador almacenadas pueden estar configuradas para hacer que un procesador realice unas operaciones de tal forma que recibir un código objeto binario compilado en software de sistema comprende recibir el código objeto binario compilado en un sistema operativo. Asimismo, las instrucciones de software ejecutables por procesador almacenadas pueden estar configuradas para hacer que un procesador realice unas operaciones de tal forma que realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada comprende traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones. Además, las instrucciones de software ejecutables por procesador almacenadas pueden estar configuradas para hacer que un procesador realice unas operaciones de tal forma que la primera arquitectura de conjunto de instrucciones es la misma que la segunda arquitectura de conjunto de instrucciones. Asimismo, las instrucciones de software ejecutables por procesador almacenadas pueden estar configuradas para hacer que un procesador realice unas operaciones de tal forma que analizar el código objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que pueden optimizarse para ahorrar energía comprende determinar si hay operaciones alternativas que logran los mismos resultados que las operaciones de código objeto identificadas; y realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada comprende reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas. Asimismo, las instrucciones de software ejecutables por procesador almacenadas pueden estar configuradas para hacer que un procesador realice unas operaciones que comprenden detectar una conexión a una nueva fuente de energía; y las instrucciones de software ejecutables por procesador almacenadas pueden estar configuradas además para hacer que un procesador realice unas operaciones de tal forma que realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada tiene lugar cuando se detecta una conexión a la nueva fuente de energía. Además, las instrucciones de software ejecutables por procesador almacenadas pueden estar configuradas para hacer que un procesador realice unas operaciones de tal forma que analizar el código objeto recibido comprende utilizar un modelo de consumo de energía para identificar segmentos de código objeto que pueden optimizarse para eficiencia energética. Asimismo, las instrucciones de software ejecutables por procesador almacenadas pueden estar configuradas para hacer que un procesador realice unas operaciones que comprenden además: medir una cantidad de energía consumida en la ejecución de segmentos de código objeto de energía optimizada; comparar la cantidad medida de energía consumida con unas predicciones del modelo de consumo de energía; y modificar el modelo de consumo de energía basándose en un resultado de la comparación.

[74] En todavía un ejemplo más, se describe un sistema en chip, que comprende una memoria; y uno o más núcleos acoplados a la memoria, en el que el uno o más núcleos están configurados con instrucciones ejecutables por procesador, para que el sistema en chip realice unas operaciones que comprenden recibir en un

sistema operativo un código objeto binario compilado; analizar el código objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que se pueden optimizar para ahorrar energía; realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada; y ejecutar el código objeto de energía optimizada. Asimismo, el uno o más núcleos pueden estar configurados con instrucciones ejecutables por procesador para que el sistema en chip realice unas operaciones de tal forma que realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada comprende traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones. Además, el uno o más núcleos pueden estar configurados con instrucciones ejecutables por procesador para que el sistema en chip realice unas operaciones de tal forma que la primera arquitectura de conjunto de instrucciones es la misma que la segunda arquitectura de conjunto de instrucciones. Asimismo, el uno o más núcleos pueden estar configurados con instrucciones ejecutables por procesador para que el sistema en chip realice unas operaciones de tal forma que analizar el código objeto recibido en un proceso de traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que puedan optimizarse para ahorrar energía comprende determinar si hay operaciones alternativas que logran los mismos resultados que las operaciones de código objeto identificadas; y realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada comprende reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas. Además, el uno o más núcleos pueden estar configurados con instrucciones ejecutables por procesador para que el sistema en chip realice unas operaciones que comprenden además detectar una conexión a una nueva fuente de energía; y el uno o más núcleos pueden estar configurados con instrucciones ejecutables por procesador para que el sistema en chip realice unas operaciones de tal forma que realizar en el proceso de traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto recibido para generar un código objeto de energía optimizada se realiza cuando se detecta una conexión a la nueva fuente de energía. Asimismo, el uno o más núcleos pueden estar configurados con instrucciones ejecutables por procesador para que el sistema en chip pueda realizar unas operaciones de tal forma que analizar el código objeto recibido comprende usar un modelo de consumo de energía para identificar segmentos de código objeto que pueden optimizarse para eficiencia energética. Asimismo, el uno o más núcleos pueden estar configurados con instrucciones ejecutables por procesador para que el sistema en chip realice unas operaciones que comprenden medir una cantidad de energía consumida en la ejecución de segmentos de código objeto de energía optimizada; comparar la cantidad medida de energía consumida con unas predicciones del modelo de consumo de energía; y modificar el modelo de consumo de energía basándose en el resultado de la comparación.

REIVINDICACIONES

1. Un procedimiento para optimizar un código objeto para ahorrar energía durante una ejecución en un dispositivo informático, que comprende:

recibir (502) un código objeto binario compilado en software de sistema de un dispositivo informático;

analizar el código objeto binario compilado recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que pueden optimizarse para ahorrar energía;

detectar una conexión a una fuente de energía diferente;

realizar en el traductor binario dinámico una traducción (800) de secuencia de instrucciones en secuencia de instrucciones del código objeto binario compilado recibido para generar (802) un código objeto de energía optimizada como respuesta a una detección de la conexión a una fuente de energía diferente; y

ejecutar (506) el código objeto de energía optimizada en un procesador del dispositivo informático.

2. El procedimiento de la reivindicación 1, en el que el software de sistema que recibe el código objeto binario compilado es uno de una máquina virtual de sistema o un hipervisor.

3. El procedimiento de la reivindicación 1, en el que el software de sistema que recibe el código objeto binario compilado es un sistema operativo.

4. El procedimiento de la reivindicación 1, en el que realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto binario compilado recibido para generar un código objeto de energía optimizada comprende traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones.

5. El procedimiento de la reivindicación 4, en el que la primera arquitectura de conjunto de instrucciones es la misma arquitectura de conjunto de instrucciones que la segunda arquitectura de conjunto de instrucciones.

6. El procedimiento de la reivindicación 1, en el que analizar el código objeto binario compilado recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que se pueden optimizar para ahorrar energía comprende determinar (802) si hay operaciones alternativas que logran los mismos resultados que las operaciones de código objeto identificadas, y

en el que realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto binario compilado recibido para generar un código objeto de energía optimizada comprende reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas.

7. El procedimiento de la reivindicación 1, en el que analizar el código objeto binario compilado recibido comprende utilizar un modelo de consumo de energía para identificar segmentos de código objeto que pueden optimizarse para eficiencia energética.

8. El procedimiento de la reivindicación 7, que comprende además:

medir (706) una cantidad de energía consumida en la ejecución de segmentos de código objeto de energía optimizada;

comparar (708) la cantidad medida de energía consumida con unas predicciones del modelo de consumo de energía; y

modificar (710) el modelo de consumo de energía basándose en un resultado de la comparación.

9. Un dispositivo informático configurado para optimizar un código objeto durante una ejecución para un ahorro de energía mejorado, que comprende:

medios para recibir un código objeto binario compilado en software de sistema;

medios para analizar el código objeto binario compilado recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que se pueden optimizar para ahorrar energía;

medios para detectar una conexión a una fuente de energía diferente;

medios para realizar en el traductor binario dinámico una traducción (800) de secuencia de instrucciones en secuencia de instrucciones del código objeto binario compilado recibido para generar (802) un código objeto de energía optimizada como respuesta a una detección de la conexión a la fuente de energía diferente; y

medios para ejecutar el código objeto de energía optimizada en un procesador del dispositivo informático.

10. El dispositivo informático de la reivindicación 9, en el que medios para recibir un código objeto binario compilado en software de sistema comprenden medios que reciben el código objeto binario compilado en uno de una máquina virtual de sistema o un hipervisor.

11. El dispositivo informático de la reivindicación 9, en el que medios para recibir un código objeto binario compilado en software de sistema comprenden medios que reciben el código objeto binario compilado en un sistema operativo.

12. El dispositivo informático de la reivindicación 9, en el que medios para realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto binario compilado recibido para generar un código objeto de energía optimizada comprenden medios para traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones.

13. Dispositivo informático de la reivindicación 12, en el que medios para traducir una primera arquitectura de conjunto de instrucciones como una segunda arquitectura de conjunto de instrucciones comprenden medios para traducir la primera arquitectura de conjunto de instrucciones como una arquitectura de conjunto de instrucciones que es la misma que la segunda arquitectura de conjunto de instrucciones.

14. El dispositivo informático de la reivindicación 9, en el que:

medios para analizar el código objeto binario compilado recibido en un traductor binario dinámico que funciona en la capa de máquina para identificar segmentos de código que se pueden optimizar para ahorrar energía comprenden medios para determinar si hay operaciones alternativas que logran los mismos resultados que las operaciones de código objeto identificadas; y

medios para realizar en el traductor binario dinámico una traducción de secuencia de instrucciones en secuencia de instrucciones del código objeto binario compilado recibido para generar un código objeto de energía optimizada comprenden medios para reemplazar, durante una traducción, las operaciones de código objeto identificadas por las operaciones alternativas.

15. Un producto de programa informático, que comprende:

un medio legible por ordenador que tiene almacenadas instrucciones en el mismo configuradas para hacer que un ordenador realice un procedimiento de acuerdo con cualquiera de las reivindicaciones 1 a 8 cuando se ejecutan.

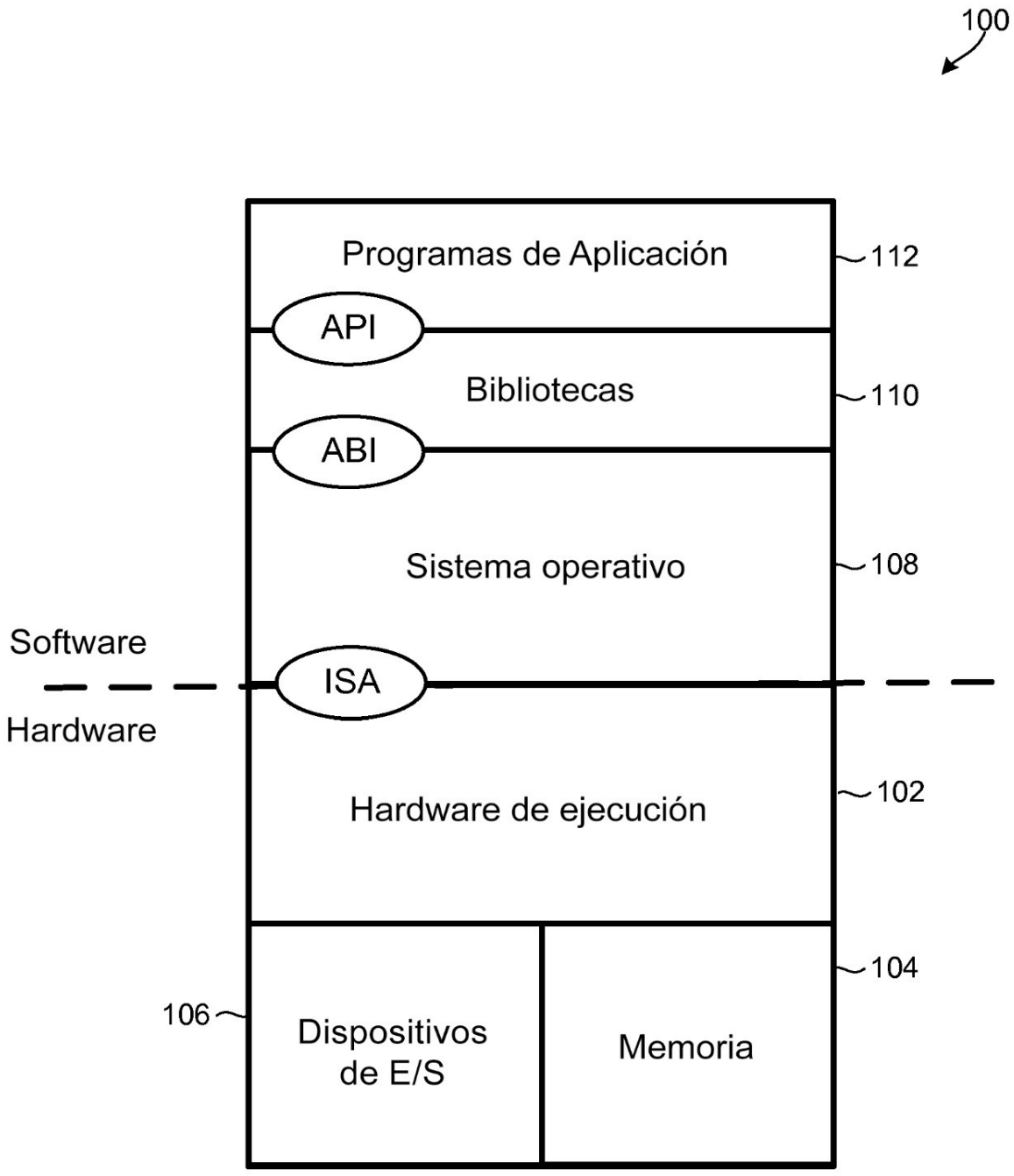
100
Programas de Aplicación
112
API
Bibliotecas
110
ABI
Sistema operativo
108
Software
ISA
Hardware
Hardware de ejecución
102
104
106
Dispositivos de E/S
Memoria

FIG. 1

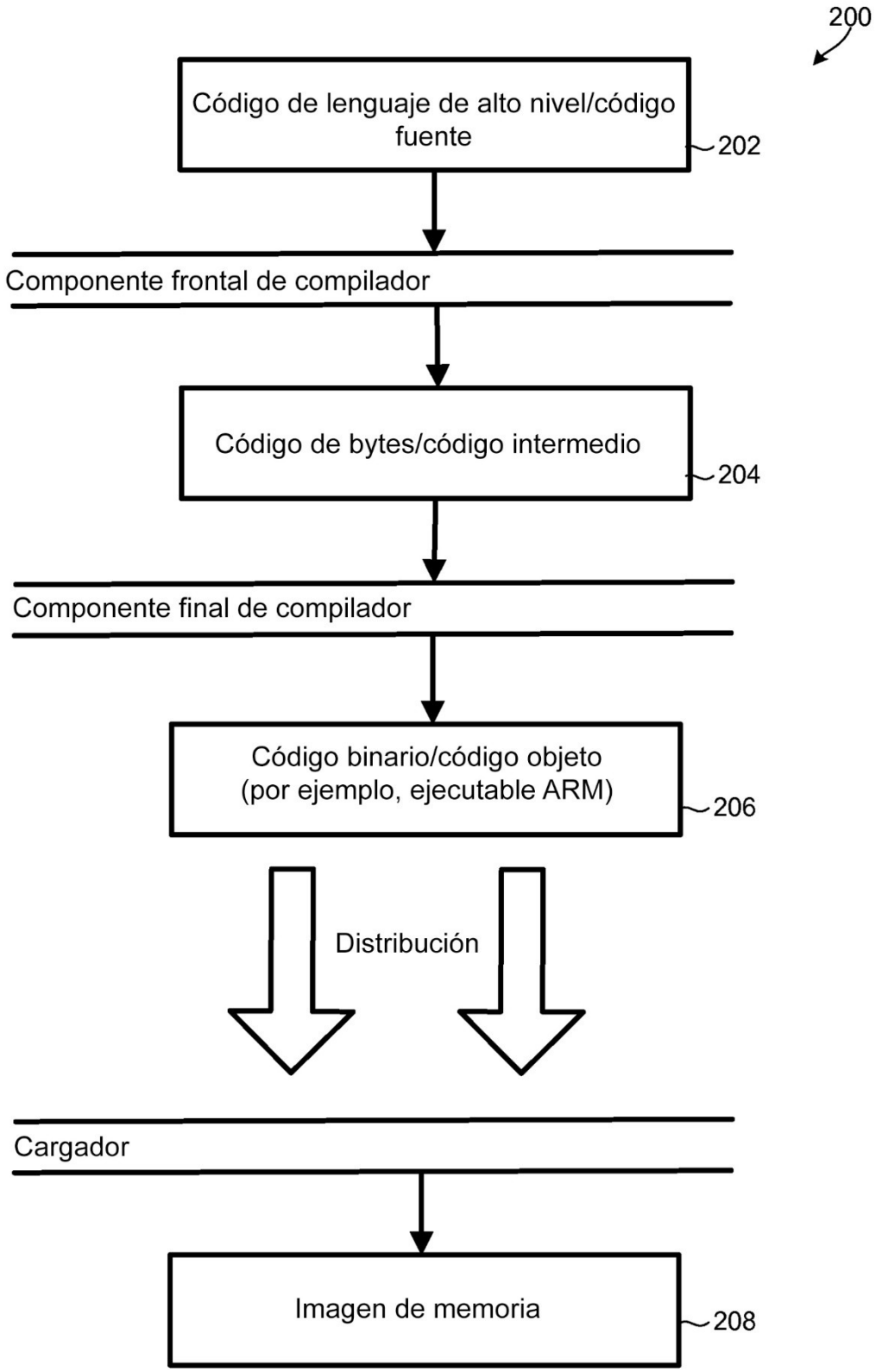
200
Código de lenguaje de alto nivel/código fuente
202
Componente frontal de compilador
Código de bytes/código intermedio
204
Componente final de compilador
Código binario/código objeto (por ejemplo, ejecutable ARM)
206
Distribución
Cargador
Imagen de memoria
208

FIG. 2A

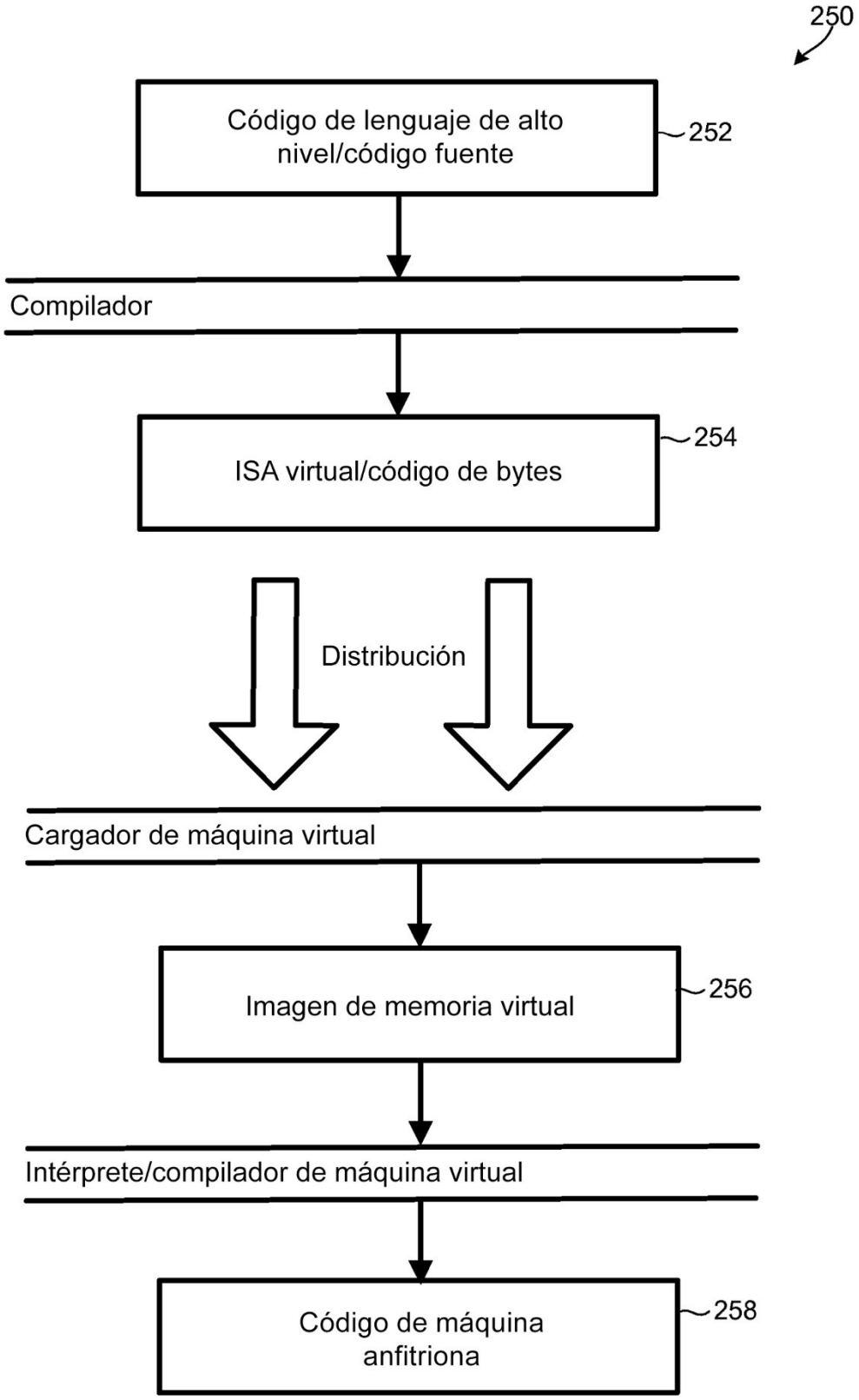
250
Código de lenguaje de alto nivel/código fuente
252
Compilador
ISA virtual/código de bytes
254
Distribución
Cargador de máquina virtual
Imagen de memoria virtual
256
Intérprete/compilador de máquina virtual
Código de máquina anfitriona
258

FIG. 2B

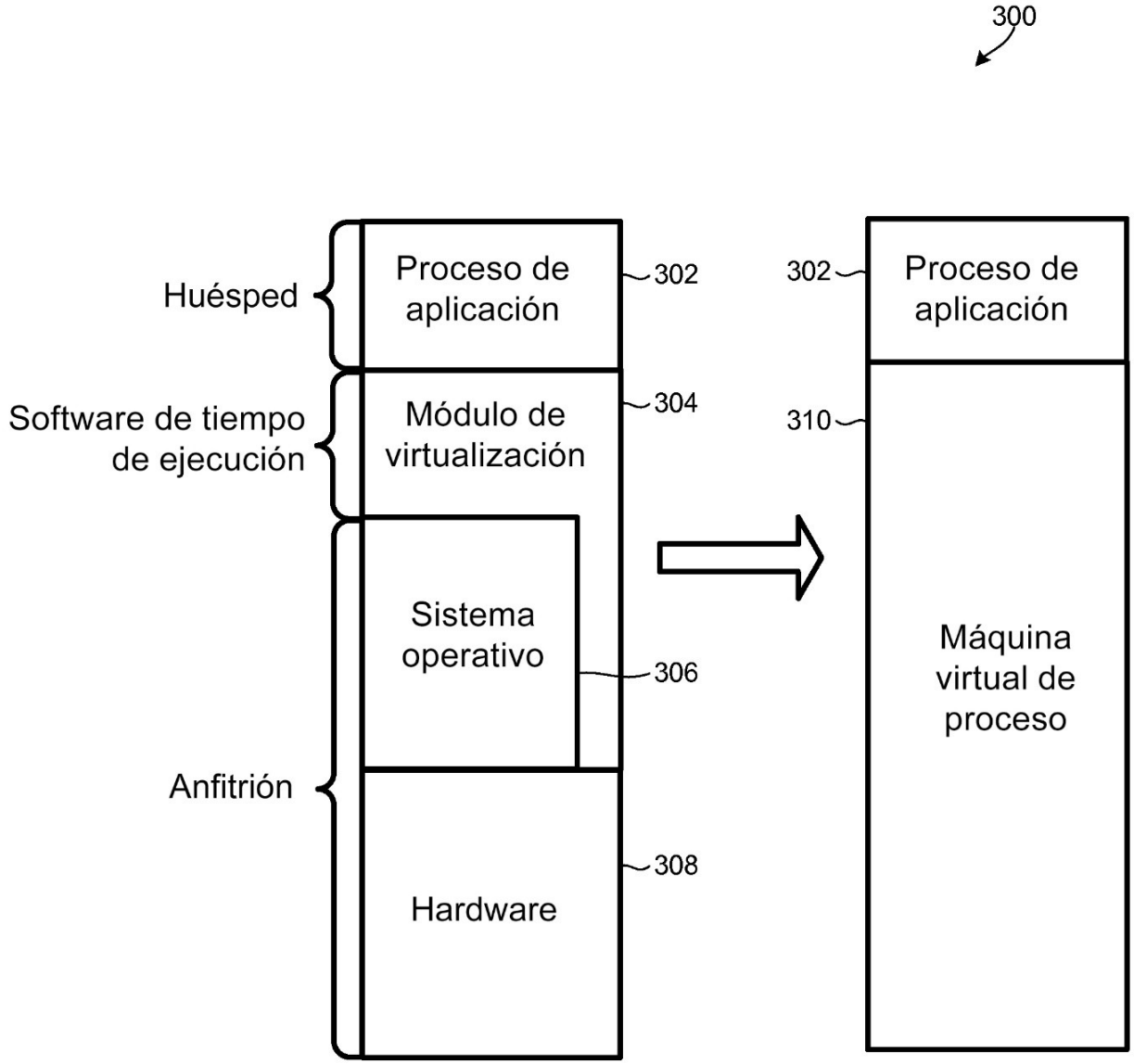
300
Huésped
Software de tiempo de ejecución
Anfitrión
Proceso de aplicación
302
Módulo de virtualización
304
Sistema operativo
306
Hardware
308
302
Proceso de aplicación
310
Máquina virtual de proceso

FIG. 3A

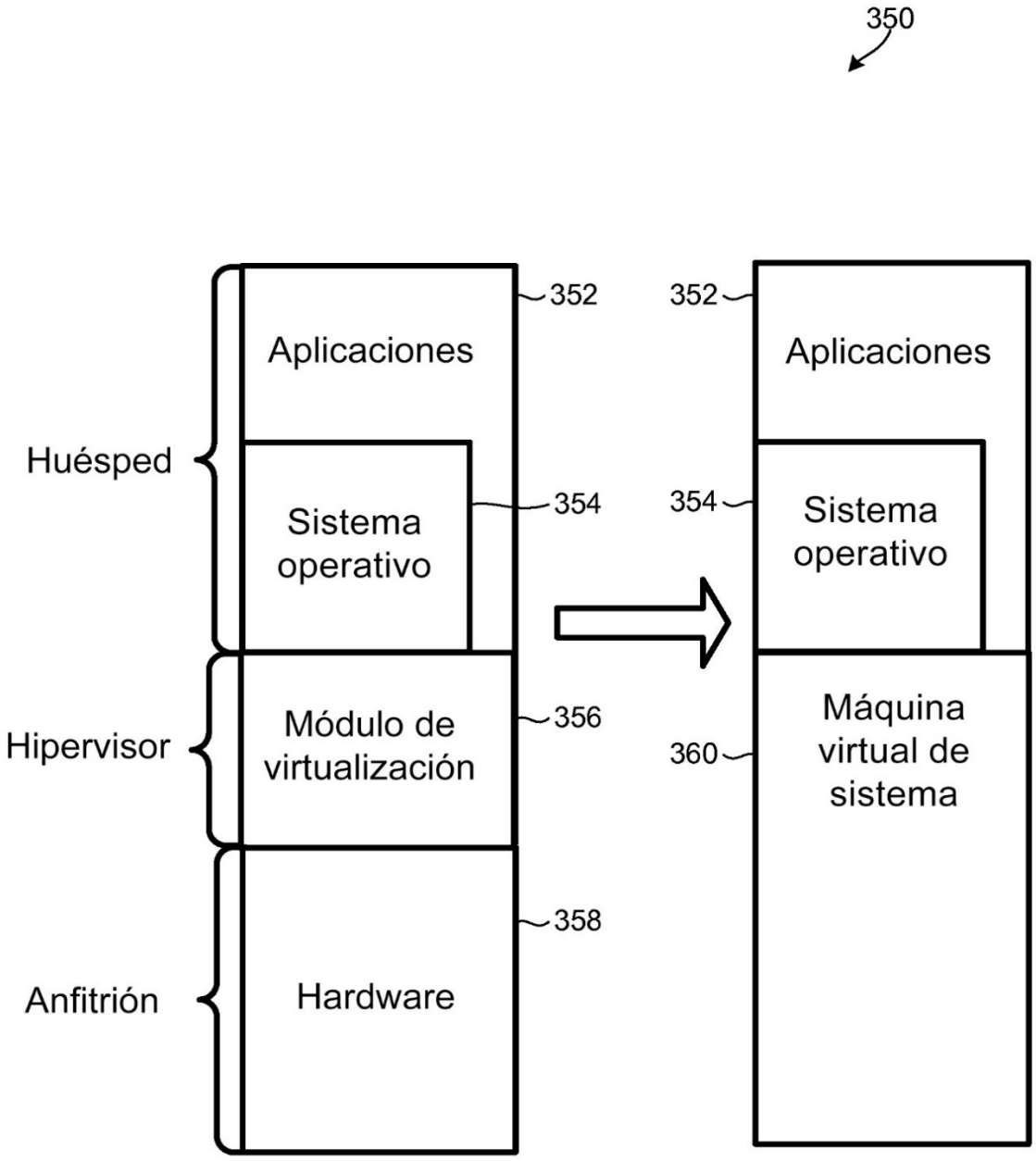
350
352
Aplicaciones
Huésped
354
Sistema operativo
Hipervisor
356
Módulo de virtualización
358
Anfitrión
Hardware
352
Aplicaciones
354
Sistema operativo
360
Máquina virtual de sistema

FIG. 3B

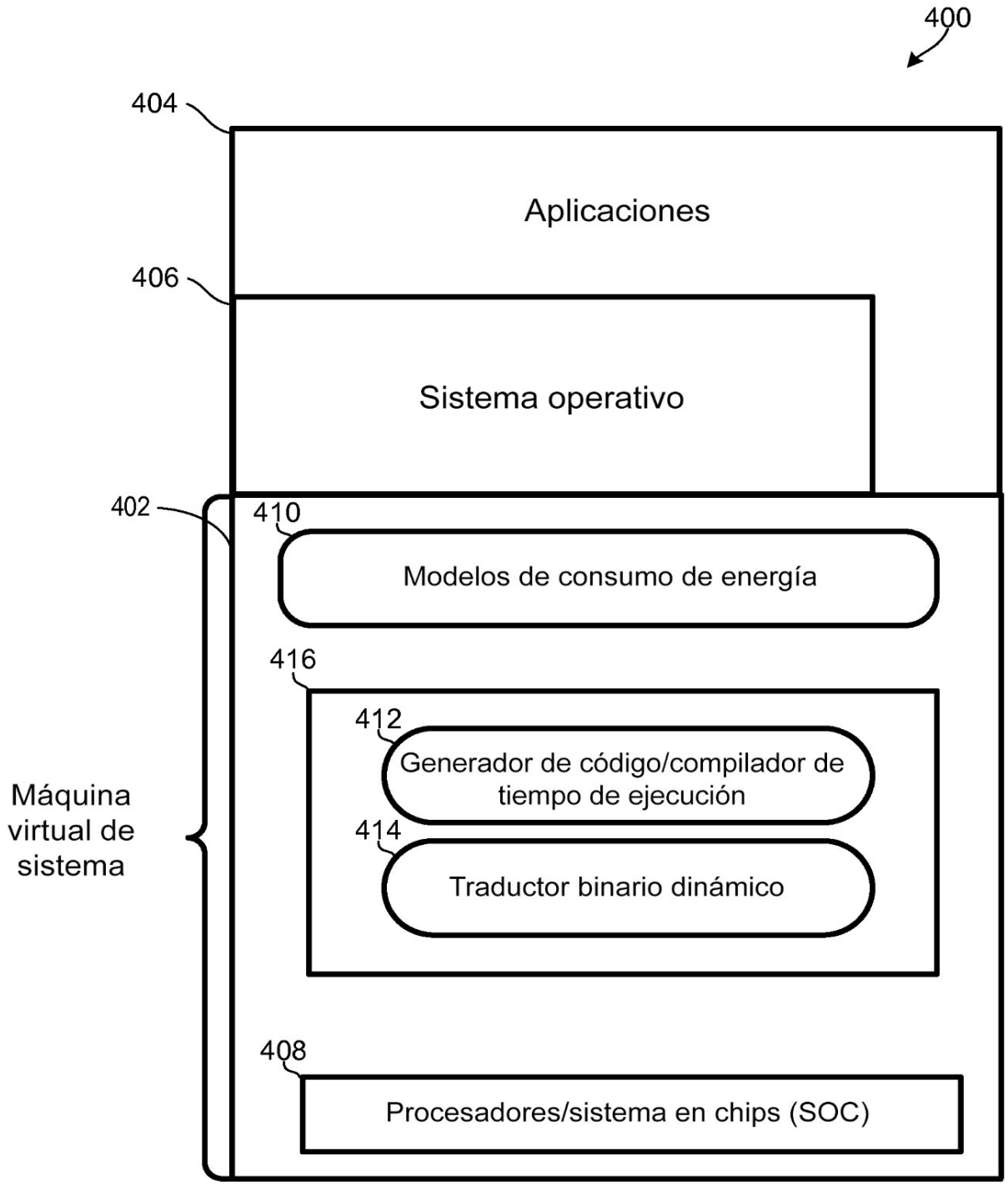
400
404
Aplicaciones
406
Sistema operativo
402
410
Modelos de consumo de energía
416
412
Generador de código/compilador de tiempo de ejecución
414
Traductor binario dinámico
Máquina virtual de sistema
408
Procesadores/sistema en chips (SOC)

FIG. 4

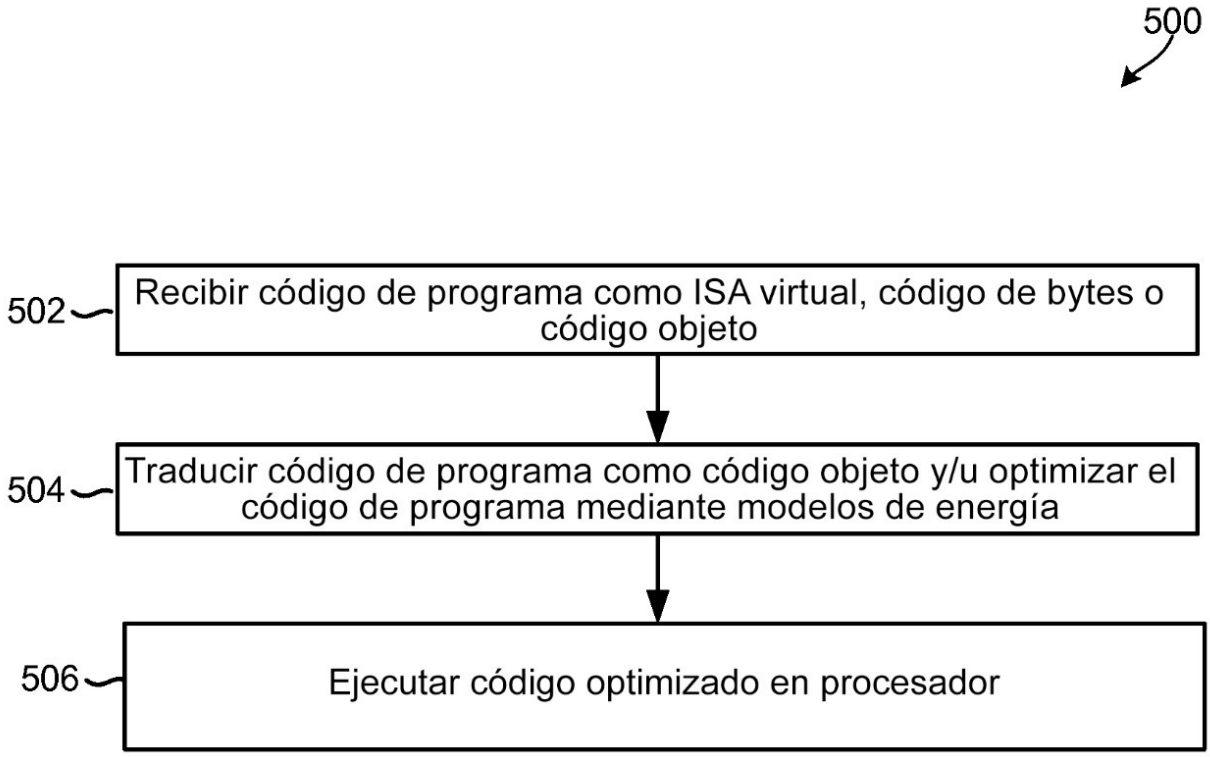
500
502
Recibir código de programa como ISA virtual, código de bytes o código objeto
504
Traducir código de programa como código objeto y/u optimizar el código de programa mediante modelos de energía
506
Ejecutar código optimizado en procesador

FIG. 5

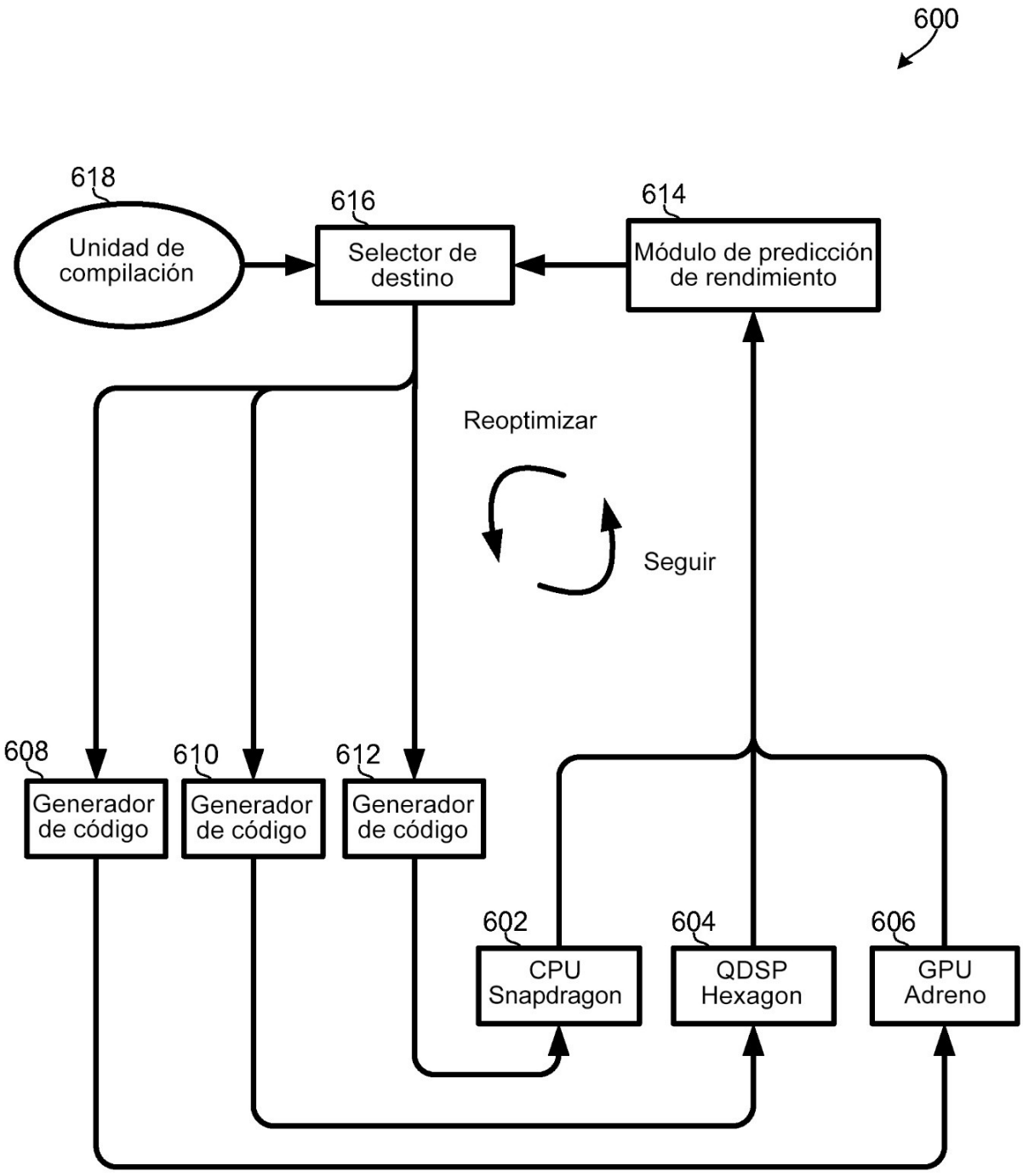
600
618
Unidad de compilación
616
Selector de destino
614
Módulo de predicción de rendimiento
Reoptimizar
Seguir
608
Generador de código
610
Generador de código
612
Generador de código
602
CPU Snapdragon
604
QDSP Hexagon
606
GPU Adreno

FIG. 6

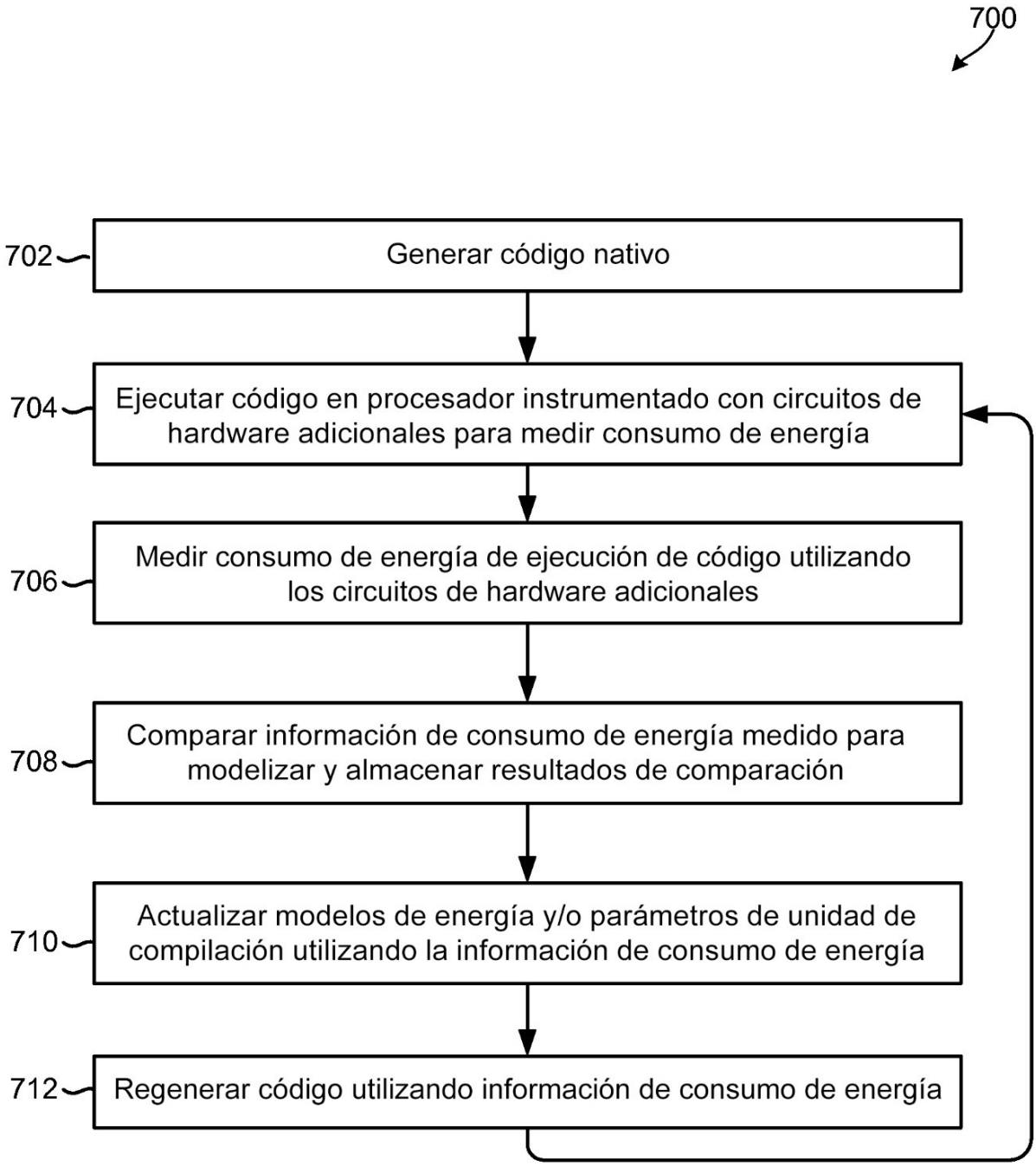
700
702
Generar código nativo
704
Ejecutar código en procesador instrumentado con circuitos de hardware adicionales para medir consumo de energía
706
Medir consumo de energía de ejecución de código utilizando los circuitos de hardware adicionales
708
Comparar información de consumo de energía medido para modelizar y almacenar resultados de comparación
710
Actualizar modelos de energía y/o parámetros de unidad de compilación utilizando la información de consumo de energía
712
Regenerar código utilizando información de consumo de energía

FIG. 7

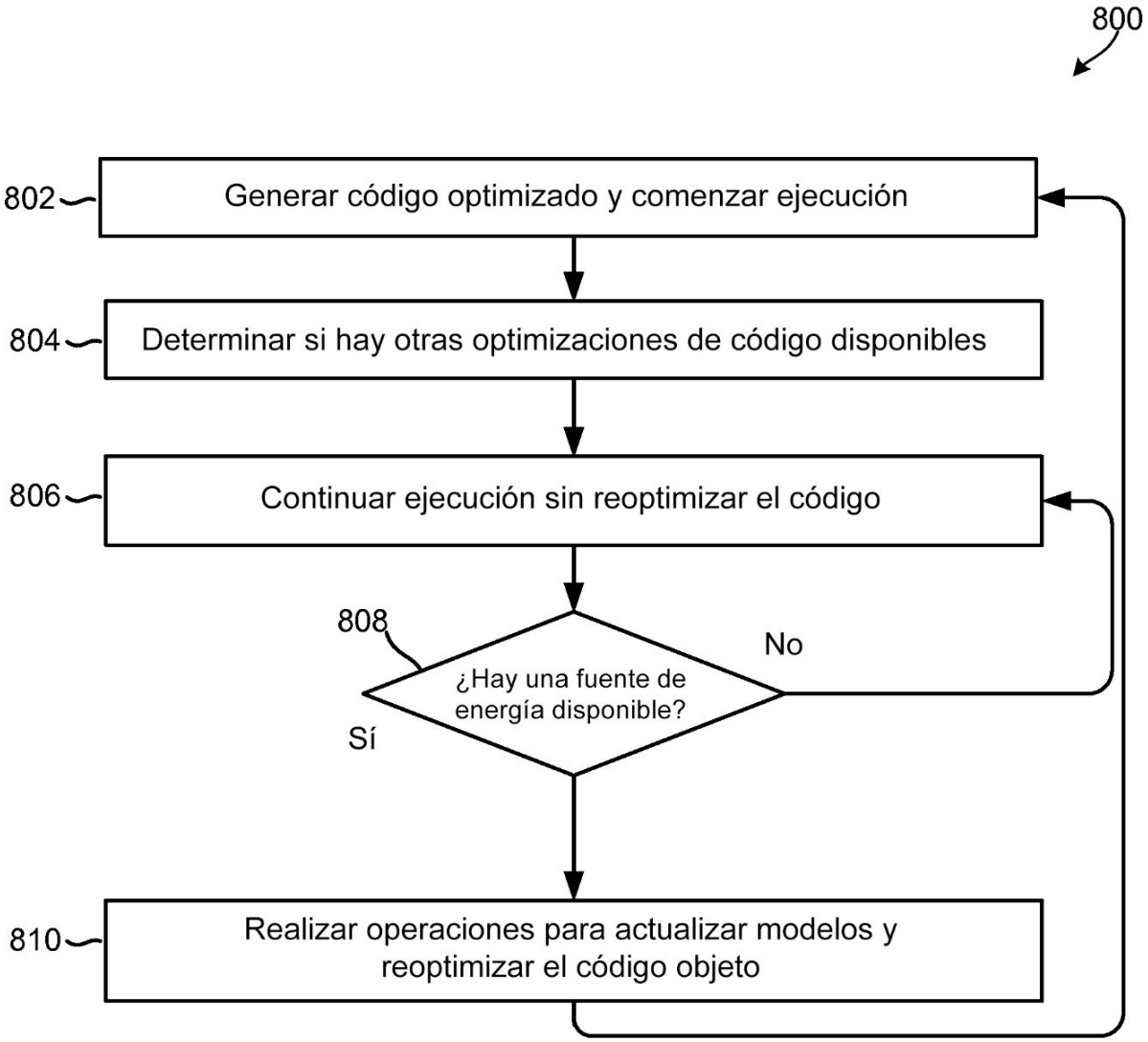
800
802
Generar código optimizado y comenzar ejecución
804
Determinar si hay otras optimizaciones de código disponibles
806
Continuar ejecución sin reoptimizar el código
808
¿Hay una fuente de energía disponible?
No
Sí
810
Realizar operaciones para actualizar modelos y reoptimizar el código objeto

FIG. 8

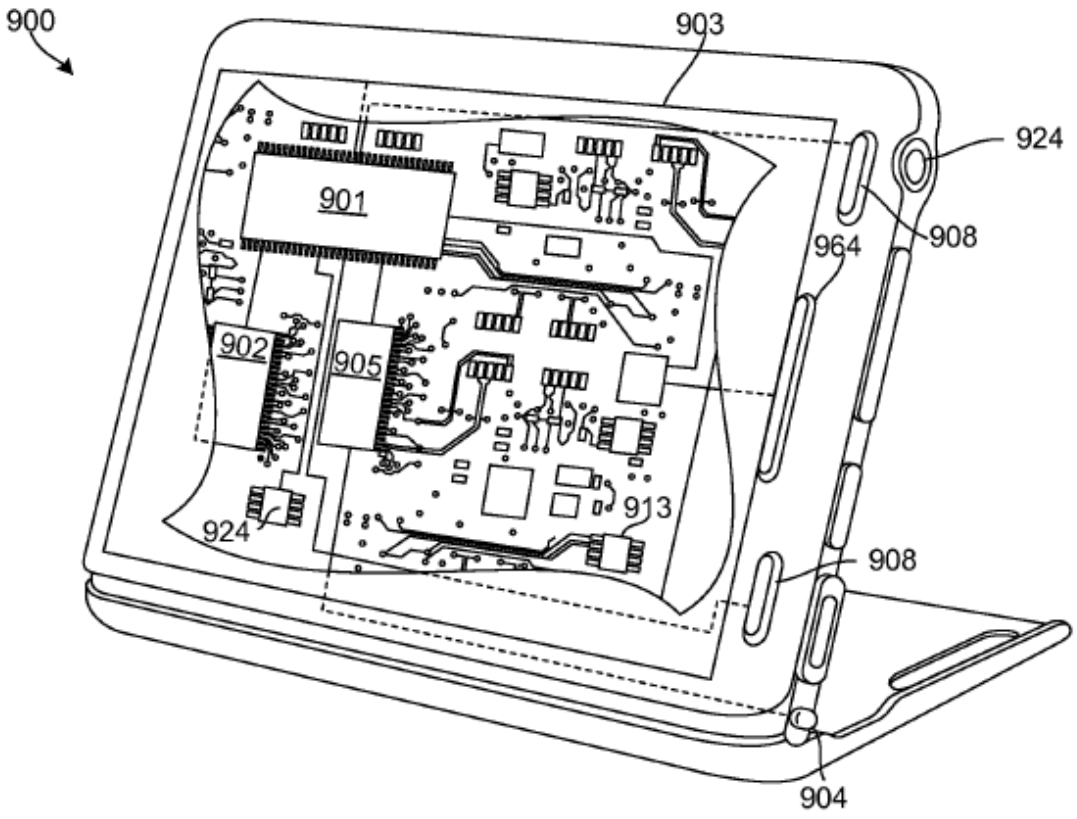

FIG. 9

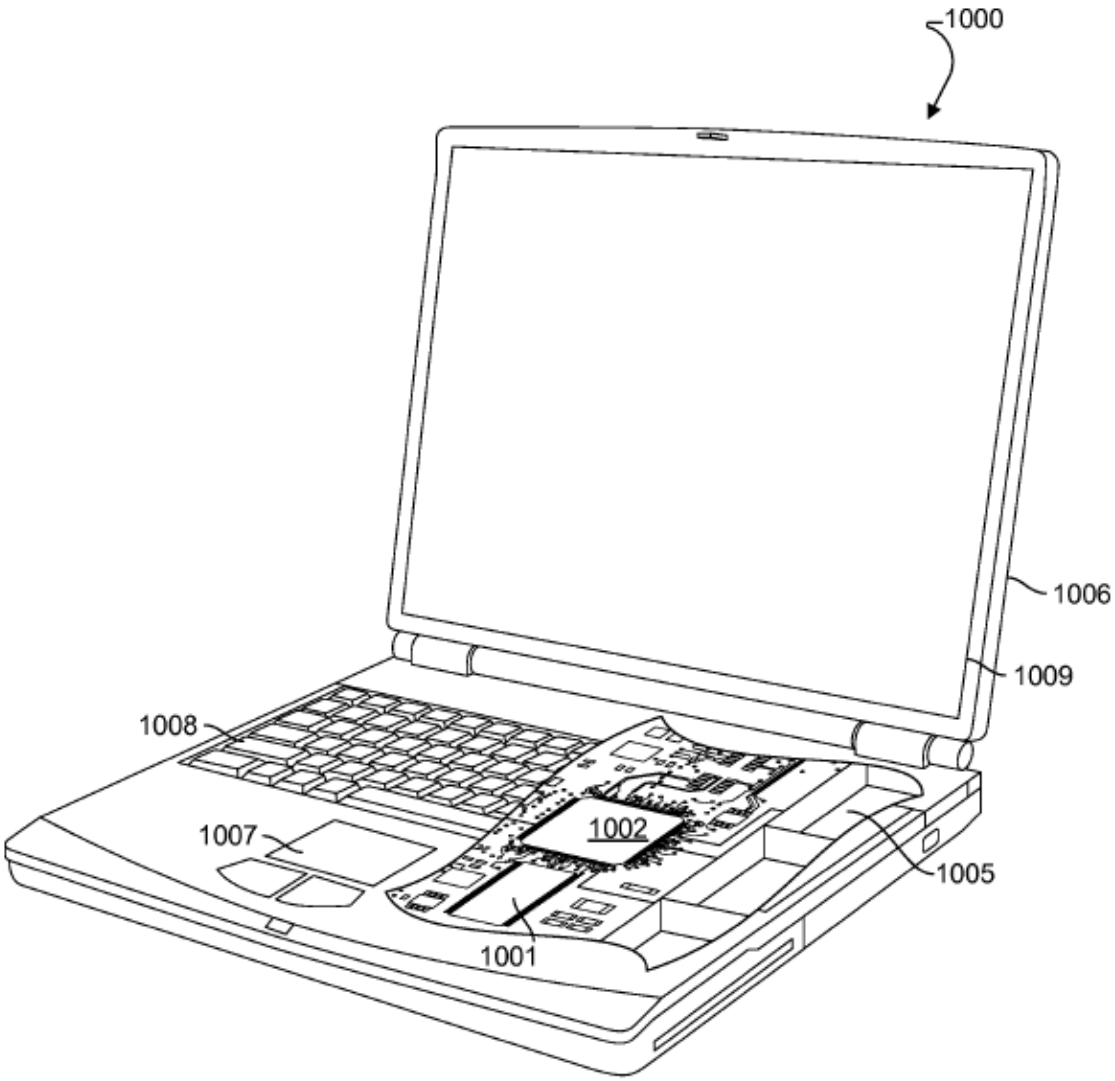

FIG. 10