(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 279 365**

(51) Int. Cl.:
G06F 9/30 (2006.01)
H04L 9/32 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Número de solicitud europea: **04731046 .1**
(86) Fecha de presentación : **04.05.2004**
(87) Número de publicación de la solicitud: **1623316**
(87) Fecha de publicación de la solicitud: **08.02.2006**

(54) Título: **Tratamiento de instrucciones de resumen de mensaje.**

(30) Prioridad: **12.05.2003 US 436230**

(45) Fecha de publicación de la mención BOPI:
16.08.2007

(45) Fecha de la publicación del folleto de la patente:
16.08.2007

(73) Titular/es:
International Business Machines Corporation
New Orchard Road
Armonk, New York 10504, US

(72) Inventor/es: **Lundvall, Shawn;**
Smith, Ronald y
Yeh, Phil Chi-Chung

(74) Agente: **Elzaburu Márquez, Alberto**

DESCRIPCIÓN

Tratamiento de instrucciones de resumen de mensaje.

Este invento se refiere a una arquitectura de sistema de ordenador y particularmente al tratamiento de nuevas instrucciones que aumentan la Arquitectura z de IBM y puede ser emulado por otras arquitecturas.

Antes de nuestro invento IBM ha creado a través del trabajo de muchos ingenieros de mucho talento que comienza con máquinas conocidas como el Sistema IBM 360 en los años 60 hasta la actualidad, una arquitectura especial que, debido a su naturaleza esencial para un sistema de cálculo, resultó conocida como "el ordenador central" cuyos principios de funcionamiento establecen la arquitectura de la máquina describiendo las instrucciones que pueden ser ejecutadas al producirse la puesta en práctica de las instrucciones del "ordenador central" que habían sido inventadas por los inventores de IBM y adoptadas por IBM, debido a su significativa contribución para mejorar el estado de la máquina de cálculo representada por "el ordenador central", como contribuciones significativas por inclusión en los Principios de Funcionamiento de IBM como se ha establecido durante años. La Primera Edición de los Principios de Funcionamiento de la Arquitectura z que fue publicada en Diciembre de 2000 ha resultado la referencia estándar publicada como SA22-7832-00.

Hemos determinado que otras nuevas instrucciones ayudarían a la técnica y podrían ser incluidas en una máquina de Arquitectura z y también emuladas por otros en máquinas más simples, como se ha descrito aquí.

Una instrucción que realiza una operación de elegir aleatoriamente el mismo elemento o "aleatorización" adecuada para calcular un resumen de mensaje es conocida a partir del documento GB 1494750.

El presente invento proporciona un método según la reivindicación 1ª.

Las características de las realizaciones preferidas del invento serán evidentes para un experto en la técnica a partir de la descripción detallada siguiente tomada en unión con los dibujos adjuntos en los que:

La fig. 1 es una representación de la instrucción Cálculo de Resumen de Mensaje Intermedio (KIMD) en el formato de instrucciones RRE;

La fig. 2 es una representación de la instrucción de Cálculo de Resumen de Último Mensaje (KLMD) en el formato de instrucciones RRE;

La fig. 3 es una tabla que muestra los códigos de función de la instrucción KIMD de la fig. 1;

La fig. 4 es una tabla que muestra los códigos de función de la instrucción KLMD de la fig. 2;

La fig. 5 es una representación de la asignación de registro general para las instrucciones KIMD y KLMD de las figs. 1 y 2;

La fig. 6 ilustra el símbolo para el Algoritmo de Resumen de Bloque SHA-1;

La fig. 7 ilustra el formato del bloque de parámetros para KIMD-Query;

La fig. 8 ilustra el formato del bloque de parámetros para KIMD-SHA-1;

La fig. 9 ilustra la operación KIMD-SHA-1;

La fig. 10 ilustra el formato para el bloque de parámetros para KLMD-Query;

La fig. 11 ilustra el formato para el bloque de parámetros para KLMD-SHA-1;

La fig. 12 ilustra la operación del Bloque Lleno de KLMD-SHA-1;

La fig. 13 ilustra la operación del Bloque Vacío de KLMD-SHA-1;

La fig. 14 ilustra la operación del Caso 1 de Bloque Parcial KLMD-SHA-1;

La fig. 15 ilustra la operación del Caso 2 de Bloque Parcial KLMD-SHA-1;

La fig. 16 es una tabla que muestra la prioridad de ejecución de las instrucciones KIMD y KLMD;

La fig. 17 ilustra nuestro coprocesador criptográfico; y

La fig. 18 muestra la realización preferida generalizada de un almacenamiento de memoria de ordenador que contiene instrucciones de acuerdo con la realización preferida y datos, así como el mecanismo para buscar, descodificar y ejecutar estas instrucciones, bien en un sistema de ordenador que emplea estas instrucciones de arquitectura o bien cuando es usado en la emulación de nuestras instrucciones de arquitectura.

Las instrucciones de resumen de mensaje descritas aquí son para calcular una representación condensada de un mensaje o archivo de datos. Las instrucciones de cálculo de resumen de mensaje intermedio y de cálculo de resumen de último mensaje serán descritas en primer lugar, seguidas por una descripción del sistema de ordenador preferido para ejecutar estas instrucciones. Como alternativa, se describirá un segundo sistema de ordenador preferido que emula a otro sistema de ordenador para ejecutar estas instrucciones.

Cálculo de resumen de mensaje intermedio (KIMD)

La fig. 1 es una representación de la instrucción de Cálculo de Resumen de Mensaje Intermedio (KIMD) en el formato de instrucciones RRE;

Cálculo de resumen de último mensaje (KLMD)

La fig. 2 es una representación de la instrucción de Cálculo de Resumen de Último Mensaje (KLMD) en el formato de instrucciones RRE;

Es realizada una función especificada por el código de función en el registro general 0.

Los Bits 16 a 23 de la instrucción y el campo R1 son ignorados.

Las posiciones de bit 57 a 63 del registro general 0 contienen el código de función. Las figs. 3 y 4 muestran los códigos de función asignados para CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO y CÁLCULO DE RESUMEN DE ÚLTIMO MENSAJE, respectivamente. Todos los demás códigos de función están sin asignar. El Bit 56 del registro general 0 debe ser cero; de otro modo, es reconocida una excepción de especificación. Todos los demás bits del registro general 0 son ignorados. El registro general 1 contiene la dirección lógica del byte más a la izquierda del bloque de parámetros en almacenamiento. En el modo de direccionamiento de 24-bits, el contenido de las posiciones de bit 40 a 63 del registro general 1 constituye la dirección, y el contenido de las posiciones de bit 0-39 es ignorado. En el modo de direccionamiento de 31-bits, el contenido de las posiciones de bit 33 a 63 del registro general 1 constituye la dirección, y el contenido de las posiciones de bit 0 a 32 es ignorado. En el modo de direccionamiento de 64-bits, el contenido de las posiciones de bit 0 a 63 del registro general 1 constituye la dirección.

Los códigos de función para CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO están mostrados en la fig. 3.

Los códigos de función para CÁLCULO DE RESUMEN DE ÚLTIMO MENSAJE están mostrados en la fig. 4.

Todos los demás códigos de función están sin asignar.

La función de consulta proporciona los medios que indican la disponibilidad de las otras funciones. Los contenidos de registros generales R2 y R2 + 1 son ignorados para la función de consulta.

Para todas las demás funciones, el segundo operando es procesado como se ha especificado por el código de función que usa un valor de encadenamiento inicial en el bloque de parámetros, y el resultado sustituye el valor de encadenamiento. Para CÁLCULO DE RESUMEN DE ÚLTIMO MENSAJE, la operación también usa una longitud de bit de mensaje en el bloque de parámetros. La operación prosigue hasta que el final de la ubicación del segundo operando es alcanzado o un número de bytes determinado por la CPU ha sido procesado, lo que tiene lugar en primer lugar. El resultado está indicado en el código de condición.

El campo R2 designa un par impar-par de registros generales y debe designar un registro con numeración impar; de otro modo, es reconocida una excepción de especificación.

La ubicación del byte más a la izquierda del segundo operando es especificada por el contenido del registro general R2. El número de bytes en la ubicación del segundo-operando está especificado en el registro general R2 + 1.

Como parte de la operación, la dirección en el registro general R2 es incrementada por el número de bytes procesado desde el segundo operando, y la longitud en el registro general R2 + 1 es disminuida por el mismo número. La formación y actualización de la dirección y longitud es dependiente del modo de direccionamiento.

En el modo de direccionamiento de 24 bits, el contenido de posiciones de bit 40-63 del registro general R2 constituye la dirección del segundo operando, y el contenido de posiciones de bit 0 a 39 son ignorados; los bits 40 a 63 de la dirección actualizada sustituyen a los bits correspondientes en el registro general R2, las puestas en práctica de la posición de bit 40 de la dirección actualizada son ignoradas, y el contenido de las posiciones de bit 32 a 39 del registro

general R2 son puestos a cero. En el modo de direccionamiento de 31 bits, el contenido de las posiciones de bit 33-63 del registro general R_ constituye la dirección del segundo operando, y el contenido de las posiciones de bit 0 a 32 es ignorado; los bits 33 a 63 de la dirección actualizada sustituyen a los bits correspondientes en el registro general R2, las puestas en práctica de la posición de bit 33 de la dirección actualizada son ignoradas, y el contenido de la posición de bit 32 del registro general R2 es puesto a cero. En el modo de direccionamiento de 64 bits, el contenido de las posiciones de bit 0 a 63 del registro general R2 constituye la dirección del segundo operando; los bits 0 a 63 de la dirección actualizada sustituyen al contenido del registro general R_ y las puestas en práctica de la posición 0 son ignoradas.

En ambos modos de direccionamiento de 24 bits y de 31 bits, el contenido de las posiciones de bit 32 a 63 del registro general R2 + 1 forma un número entero binario sin signo de 32 bits que especifica el número de bytes en el segundo operando; y el valor actualizado sustituye al contenido de las posiciones de bit 32 a 63 del registro general R2 + 1. En el modo de direccionamiento de 64 bits, el contenido de las posiciones de bit 0 a 63 del registro general R2 + 1 forma un número entero binario sin signo de 64 bits que especifica el número de bytes en el segundo operando; y el valor actualizado sustituye al contenido del registro general R2 + 1.

En el modo de direccionamiento de 24 bits o 31 bits, el contenido de las posiciones de bit 0 a 31 de los registros generales R2 y R2 +1, siempre permanecen sin cambios.

La fig. 5 muestra el contenido de los registros generales recién descritos.

En el modo de registro de acceso, los registros de acceso 1 y R2 especifican los espacios de dirección que contienen el bloque de parámetros y el segundo operando, respectivamente.

El resultado es obtenido como si el tratamiento comienza en el extremo izquierdo del segundo operando y prosigue hacia la derecha, bloque por bloque. La operación es finalizada cuando todos los bytes fuente en el segundo operando han sido procesados (denominada terminación normal), o cuando un número de bloques de determinado por la CPU que es menor que la longitud del segundo operando ha sido procesado (denominada terminación parcial). El número de bloques de determinado por la CPU depende del modelo, y puede ser un número diferente cada vez que la instrucción es ejecutada. El número de bloques determinado por la CPU normalmente no es cero. En ciertas situaciones inusuales, este número puede ser cero, y el código de condición 3 puede ser ajustado sin progreso. Sin embargo, la CPU protege contra la nueva ocurrencia sin fin de este caso sin progreso.

Cuando el campo del valor de encadenamiento se superpone a cualquier parte del segundo operando, el resultado en el campo del valor de encadenamiento es impredecible.

Para CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO, la terminación normal tiene lugar cuando el número de bytes en el segundo operando como se ha especificado en el registro general R2 + 1 ha sido procesado. Para CÁLCULO DE RESUMEN DE ÚLTIMO MENSAJE, después de que todos los bytes como se ha especificado en el registro general R2 + 1 hayan sido procesados, la operación de relleno es realizada, y a continuación tiene lugar la terminación normal.

Cuando la operación finaliza debido a la terminación normal, el código de condición 0 es ajustado y el valor resultante en R2 + 1 es cero. Cuando la operación finaliza debido a la terminación parcial, el código de condición 3 es ajustado y el valor resultante en R2 + 1 no es cero.

Cuando la longitud del segundo operando es inicialmente cero, el segundo operando no es accedido, los registros generales R2 y R2 + 1 no son cambiados, y el código de condición 0 es ajustado. Para CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO, el bloque de parámetros no es accedido. Sin embargo, para CÁLCULO DE RESUMEN DE ÚLTIMO MENSAJE, la operación de relleno en el caso del bloque vacío (L = 0) es realizada y el resultado es almacenado en el bloque de parámetros.

Como se ha observado por otros programas de CPU y de canal, las referencias al bloque de parámetros y operandos de almacenamiento pueden ser referencias de múltiples accesos, los accesos a estas ubicaciones de almacenamiento no son necesariamente concurrentes en el bloque, y la secuencia de estos accesos o referencias está sin definir.

La excepciones de acceso pueden ser informadas para una parte mayor del segundo operando que es procesada en una única ejecución de la instrucción; sin embargo, las excepciones de acceso no son reconocidas para ubicaciones más allá de la longitud del segundo operando ni para ubicaciones más de 4K bytes más allá de la posición actual que está siendo procesada.

Símbolo Usados en Descripciones de Función

Los símbolos de la fig. 6 son usados en la descripción subsiguiente de las funciones de CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO y CÁLCULOS DE RESUMEN DE ÚLTIMO MENSAJE. Puede encontrarse otra descripción del algoritmo aleatorio seguro en la Norma de Aleatoriedad Segura, publicación 180-1 de Normas de Tratamiento de Información General, Instituto Nacional de Normas y Tecnología, Washington DC, 17 de Abril de 1995.

KIMD-Query (Código 0 de Función de KIMD)

Las ubicaciones de los operandos y las direcciones usadas por la instrucción son como se ha mostrado en la fig. 5.

El bloque de parámetros de KIMD-Query tiene el formato mostrado en la fig.7.

Una palabra de estado de 128 bits es almacenada en el bloque de parámetros. Los bits 0-127 de este campo corresponden a códigos de función 0-127, respectivamente, de la instrucción CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO. Cuando un bit es uno, la función correspondiente es instalada; de otro modo, la función no es instalada.

El código de condición 0 es ajustado cuando la ejecución de la función KIMD-Query se completa; el código de condición 3 no es aplicable a esta función.

KIMD-SHA (Código 1 de Función KIMD)

Las ubicaciones de los operandos y direcciones usadas por la instrucción son como se ha mostrado en la fig. 5.

El bloque de parámetros usado para la función KIMD-SHA tiene el formato mostrado en la fig. 8.

Un resumen de mensaje intermedio de 20 bytes es generado para los bloques de mensajes de 64 bytes en el operando 2 usando el algoritmo de resumen de bloque SHA-1 con el valor de encadenamiento de 20 bytes en el bloque de parámetros. El resumen de mensaje intermedio generado, también llamado el valor de encadenamiento de salida (OV), es almacenado en el campo de valor de encadenamiento del bloque de parámetros. La operación KIMD-SHA-1 está mostrada en la fig. 9.

KLMD-Query (Código 0 de Función KLMD)

Las ubicaciones de los operandos y direcciones usadas por la instrucción son como se ha mostrado en la fig. 5.

El bloque de parámetros usado para la función KLMD-Query tiene el formato mostrado en la fig. 10.

Una palabra de estado de 128 bits es almacenada en el bloque de parámetros. Los bits 0-127 de este campo corresponden a códigos de función 0-127, respectivamente, de la instrucción CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO. Cuando un bit es uno, la función correspondiente es instalada; de otro modo, la función no es instalada.

El código de condición 0 es ajustado cuando la ejecución de la función KLMD-Query se completa; el código de condición 3 no es aplicable a esta función.

KLMD-SHA (Código 1 de Función KLMD)

Las ubicaciones de los operandos y direcciones usadas por la instrucción son como se ha mostrado en la fig. 5.

El bloque de parámetros usado para la función KLMD-SHA-1 tiene el formato mostrado en la fig. 11.

El resumen de mensaje para el mensaje (M) en el operando 2 es generado usando el algoritmo SHA-1 con el valor de encadenamiento e información de longitud de bits del mensaje en el bloque de parámetros.

Si la longitud del mensaje en el operando 2 es igual o mayor de 64 bytes, un resumen de mensaje intermedio es generado para cada bloque de mensajes de 64 bytes usando el algoritmo de resumen de bloques SHA-1 con el valor de encadenamiento de 20 bytes en el bloque de parámetros, y el resumen de mensaje intermedio generado, también llamado el valor de encadenamiento de salida (OV), es almacenado en el campo de valor de encadenamiento del bloque de parámetros. Esta operación está mostrada en la fig. 12 y se repite hasta que el mensaje restante es menor de 64 bytes. Si la longitud del mensaje o el mensaje restante es de cero bytes, entonces la operación de la fig. 13 es realizada.

Si la longitud del mensaje o el mensaje restante están entre un byte y 55 bytes inclusive, entonces la operación de la fig. 14 es realizada; si la longitud está entre 56 bytes y 63 bytes inclusive, entonces la operación de la fig. 15 es realizada. El resumen de mensaje, también llamado el valor de encadenamiento de salida (OCV), es almacenado en el campo de valor de encadenamiento del bloque de parámetros.

Símbolos Adicionales Usados en Funciones KLMD

Los siguientes símbolos adicionales son usados en la descripción de las funciones de CÁLCULO DE RESUMEN DE ÚLTIMO MENSAJE.

Explicación de Símbolos para las Figuras de Función de KLMD

L Longitud en bytes del operando 2 en almacenamiento.

P<n> n bytes de relleno; el byte más a la izquierda es 80 hex; todos los otros bytes son 00 hex.

Z<56> 56 bytes de relleno de cero

Mb1 un valor de 8 bytes que especifica la longitud en bits del mensaje total

q<64> un bloque de relleno, consistente de 56 bytes de cero seguidos por un mb1 de 8 bytes.

Condiciones Especiales para KIMD y KLMD

Una excepción de especificación es reconocida y ninguna otra acción es tomada si ocurre cualquiera de las siguientes cosas:

1. El bit 56 del registro general 0 no es cero.

2. Los bits 57-63 del registro general 0 especifican un código de función sin asignar o sin instalar.

3. El campo R2 designa un registro de numeración par o registro general 0.

4. Para CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO, la longitud del segundo operando no es un múltiplo del tamaño del bloque de datos de la función designada (véase fig. 3 para determinar los tamaños del bloque de datos para CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO). Esta condición de excepción de especificación no se aplica a la función de consulta, no se aplica a CÁLCULO DE RESUMEN DE ÚLTIMO MENSAJE.

Código de condición resultante:

0 Terminación normal

1 --

2 --

3 Terminación parcial

Excepciones del programa:

Acceso (buscar, operando 2 y longitud en bits de mensaje, buscar y almacenar, valor de encadenamiento)

Operación (si la ayuda de seguridad del mensaje no está instalada)

Especificación

Notas de programación:

1. El bit 56 del registro general 0 está reservado para extensión futura y debería ser ajustado a cero.

2. Cuando el código 3 de condición es ajustado, la dirección y longitud del segundo operando en los registros generales R2 y R2+1, respectivamente, y el valor de encadenamiento en el bloque de parámetros, son actualizados usualmente de tal manera que el programa puede simplemente bifurcarse de nuevo a la instrucción para continuar la operación.

Para situaciones inusuales, la CPU se protege contra nuevas ocurrencias sin fin del caso sin progreso. Así, el programa puede bifurcarse de manera segura de nuevo a la instrucción siempre que el código 3 de condición esté ajustado sin exposición a un bucle sin fin.

3. Si la longitud del segundo operando no es cero inicialmente y el código 0 de condición es ajustado, los registros son actualizados de la misma manera que para el código 3 de condición; el valor de encadenamiento en este caso es tal que los operandos adicionales pueden ser procesados como si fuesen parte de la misma cadena.

4. Las instrucciones CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO y CÁLCULO DE RESUMEN

DE ÚLTIMO MENSAJE están diseñadas para ser usadas por un enlace de programación de aplicación de servicio de seguridad (API). Estos API proporcionan el programa con medios para calcular el resumen de mensajes de tamaño casi ilimitado, incluyendo aquellos demasiado largos para ajustarse en almacenamiento todos de una vez. Esto se consigue permitiendo que el programa pase el mensaje al API en partes. Las siguientes notas de programación están descritas en términos de estos API.

5. Antes de procesar la primera parte de un mensaje, el programa debe ajustar los valores iniciales del campo de valor de encadenamiento. Para SHA-1, Los valores de encadenamiento iniciales son los siguientes:

H0 = x'6745 2301'

H1 = x'EFCD AB89'

H2 = x'98BA DCFE'

H3 = x'1032 5476'

H4 = x'C3D2 E1F0'

6. Cuando se procesan partes de mensaje distintas de la última, el programa debe procesar partes del mensaje en múltiplos de 512 bits (64 bytes) y usar la instrucción de CÁLCULO DE RESUMEN DE MENSAJE INTERMEDIO.

7. Cuando se procesa la última parte del mensaje, el programa debe calcular la longitud del mensaje original en bits y colocar este valor de 64 bits en el campo longitud en bits de mensaje del bloque de parámetros, y usar la instrucción CÁLCULO DE RESUMEN DE ÚLTIMO MENSAJE.

8. La instrucción CÁLCULO DE RESUMEN DE ÚLTIMO MENSAJE no requiere que el segundo operando sea un múltiplo del tamaño del bloque. Procesa en primer lugar bloques completos, y puede ajustar el código 3 de condición antes de procesar todos los bloques. Después de procesar todos los bloques completos, realiza la operación de relleno que incluye la parte restante del segundo operando. Esto puede requerir una o más interacciones del algoritmo de resumen de bloque SHA-1.

9. La instrucción CÁLCULO DE RESUMEN DE ÚLTIMO MENSAJE proporciona el relleno de SHA-1 para mensajes que son un múltiplo de ocho bits de longitud. Si SHA-1 ha de ser aplicado a una cadena de bits que no es un múltiplo de ocho bits, el programa debe realizar el relleno y usar la instrucción CALCULAR RESUMEN DE MENSAJE INTERMEDIO.

Cripto-coprocesador

La realización preferida proporciona un cripto-coprocesador que puede ser usado con las instrucciones descritas aquí y para ejecutar mensajes cifrados y ayudar en una variedad de tareas de mensajes de encadenamiento que pueden ser empleadas para uso encadenado y criptográfico con las instrucciones apropiadas.

La fig. 17 ilustra nuestro coprocesador criptográfico que está sujeto directamente a un trayecto de datos común a todas las unidades de ejecución internas en el microprocesador de propósito general, que tiene múltiples segmentos (pipelines) de ejecución. La línea (1) de transmisión interna del microprocesador que es común a todas las otras unidades de ejecución está unida a la unidad de control criptográfico (2), y la unidad de control mira la línea de transmisión para las instrucciones de procesador que debería ejecutar.

La unidad de control criptográfica proporciona un coprocesador criptográfico directamente unido a un trayecto de datos común a todas las unidades de ejecución internas de la unidad de tratamiento central en un microprocesador de propósito general que proporciona el hardware disponible (E_0....E_n), o desde una combinación del mismo en la realización preferida que tiene múltiples segmentos de ejecución) para la unidad de tratamiento central. Cuando una instrucción criptográfica es encontrada en el registro (3) de comandos u órdenes, la unidad de control (2) invoca al algoritmo apropiado desde el hardware disponible. Los datos del operando son entregados sobre la misma línea de transmisión del microprocesador interno mediante un registro FIFO de entrada (4). Cuando una operación es completada una banderola es ajustada en un registro de estado (6) y los resultados están disponibles para ser leídos desde el registro FIFO de salida (5).

La realización preferida ilustrada de nuestro invento está diseñada para ser extensible para incluir tantos motores de hardware como se requiera por una implantación particular que depende de los objetivos de prestaciones del sistema. Los trayectos de datos a los registros de entrada y salida (7) son comunes entre todos los motores.

La realización preferida de las funciones criptográficas del invento son implantadas en el hardware de la unidad de ejecución en la CPU y esta implantación permite una latencia inferior para llamar y ejecutar operaciones de encriptación o codificación y aumenta la eficiencia.

Esta latencia disminuida mejora ampliamente la capacidad de los procesadores de propósito general en sistemas que hacen frecuentemente muchas operaciones de encriptación, particularmente cuando sólo están involucradas cantidades pequeñas de datos. Esto permite una implantación que puede acelerar significativamente los procesos implicados en asegurar transacciones en línea. La mayoría de los métodos comunes para asegurar transacciones en línea implica un conjunto de tres algoritmos. El primer algoritmo es solamente usado una vez por sesión, y puede ser implantado en el hardware o software, mientras las otras operaciones son invocadas con cada transacción de la sesión, y el coste en latencia de hardware externo que llama así como el coste en tiempo de ejecutar el algoritmo en el software son ambos eliminados con este invento.

En la fig. 18 hemos mostrado conceptualmente cómo poner en práctica lo que tenemos en una realización preferida implantada en un ordenador principal que tiene el microprocesador descrito antes que puede ser usado de manera efectiva, como hemos probado experimentalmente dentro de IBM, en una implantación comercial del formato de inspección de arquitectura del ordenador de facilidad de desplazamiento largo las instrucciones son usadas por programadores, usualmente hoy programadores "C". Estos formatos de instrucciones almacenados en el medio de almacenamiento pueden ser ejecutados originalmente en un Servidor IBM de Arquitectura Z, o alternativamente en máquinas que ejecutan otras arquitecturas. Pueden ser emulados en los servidores de ordenador principal de IBM existentes y futuros y en otras máquinas de IBM (por ejemplo servidores pSeries y servidores xSeries). Pueden ser ejecutados en máquinas que funcionan en Linux en una amplia variedad de máquinas que usan hardware fabricado por IBM, Intel, AMD, Sun Microsystems y otros. Además de la ejecución en ese hardware bajo una Arquitectura Z, Linux puede ser usado así como máquinas que usan emulación por Hércules, UMX, FXI o Soluciones de Plataforma, donde la ejecución generalmente es en un modo de emulación. En el modo de emulación la instrucción específica que es emulada es descodificada, y una subrutina construida para implantar la instrucción individual, como en una subrutina "C" o controlador, o algún otro método de proporcionar un controlador para el hardware especifico como queda dentro de la experiencia de los técnicos después de la comprensión de la descripción de la realización preferida. Distintas patentes de emulación de software y hardware que incluyen, pero no están limitadas a los documentos US5551013, US6009261, US5574873, US6308255, US6463582 y US5790825, ilustran la variedad de modos conocidos para conseguir la emulación de un formato de instrucción con arquitectura para una máquina diferente para una máquina objetivo disponible para los expertos en la técnica, así como a aquellas técnicas comerciales de software usadas por los citados anteriormente.

En la arquitectura de ordenador preferida los formatos de instrucción de desplazamiento largo existentes para una instrucción no superescalar forma la dirección de almacenamiento del operando sumando el registro base y el desplazamiento sin signo de 12 bits o el registro de base, el registro de índice, y el desplazamiento sin signo de 12 bits y los nuevos formatos de instrucción de desplazamiento largo forman la dirección de almacenamiento de operando mediante la suma del registro de base y el desplazamiento con signo de 20 bits o el registro base, el registro de índice, y el desplazamiento con signo de 20 bits.

Como se ha ilustrado en la fig. 18, estas instrucciones son ejecutadas en hardware por un procesador o por emulación de dicha instrucción ajustada por software que se ejecuta en un ordenador que tiene un ajuste de instrucción original diferente.

En la fig. 18, #501 muestra un almacenamiento de memoria de ordenador que contiene instrucciones y datos. Las instrucciones de desplazamiento largo descritas en este invento se almacenarían inicialmente en este ordenador. #502 muestra un mecanismo para buscar instrucciones desde una memoria de ordenador y puede también contener memoria tampón local de estas instrucciones que ha buscado. A continuación las instrucciones nuevas son transferidas a un descodificador de instrucción, #503, donde determina qué tipo de instrucción ha sido buscado. #504, muestra un mecanismo para ejecutar instrucciones. Esto puede incluir cargar datos a un registro desde la memoria, #501, que almacena datos de nuevo a la memoria desde un registro, o que realiza algún tipo de operación aritmética o lógica. Este tipo exacto de operación que ha de ser realizada ha sido previamente determinado por el descodificador de instrucciones. Las instrucciones de desplazamiento largo descritas en este invento serían ejecutadas aquí. Si las instrucciones de desplazamiento largo están siendo ejecutadas originalmente en un sistema de ordenador, entonces este diagrama es completo como se ha descrito antes. Sin embargo, si una arquitectura de ajuste de instrucción, que contiene instrucciones de desplazamiento largo, está siendo emulada en otro ordenador, el proceso anterior sería puesto en práctica en el software en un ordenador anfitrión, #505. En este caso, los mecanismos antes establecidos serían típicamente implantados como una o más subrutinas de software dentro del software del emulador. En ambos casos una instrucción es buscada, descodificada y ejecutada.

Más particularmente, estas instrucciones de arquitectura pueden ser usadas con una arquitectura de ordenador con formatos de instrucción existentes con un desplazamiento sin signo de 12 bits usado para formar la dirección de almacenamiento del operando y también uno que tienen formatos de instrucción adicional que proporcionan un desplazamiento de bits adicional, preferiblemente 20 bits, que comprende un desplazamiento con signo extendido usado para formar la dirección de almacenamiento del operando. Estas instrucciones de arquitectura de ordenador comprenden software de ordenador, almacenado en un medio de almacenamiento de ordenador, para producir el código que funciona del procesador que utiliza el software del ordenador, y que comprende el código de instrucción para usar por un compilador o emulador/intérprete que es almacenado en un medio 501 de almacenamiento de ordenador, y en el que la primera parte del código de instrucción comprende un código de operación que ha especificado la operación que ha de ser realizada y una segunda parte que designa los operandos para los que participa. Las instrucciones de

desplazamiento largo permiten direcciones adicionales para ser accedidas directamente con el uso de la instrucción de facilidad de desplazamiento largo.

Como se ha ilustrado en la fig. 18, estas instrucciones son ejecutadas en el hardware por un procesador o por emulación de dicha instrucción ajustada por el software que se ejecuta en un ordenador que tiene un conjunto de instrucción original diferente.

De acuerdo con la arquitectura de ordenador preferida el campo de desplazamiento está definido como en dos partes, siendo la parte menos significativa de 12 bits llamada el DL, DL1 para el operando 1 o DL2 para el operando 2, y siendo la parte más significativa de 8 bits llamada el DH, DH1 para el operando 1 o DH2 para el operando 2.

Además, la arquitectura de ordenador preferida tiene un formato de instrucción de tal manera que el código de operación está en las posiciones de bits 0 a 7 y de 40 a 47, un registro objetivo llamado R1 en posiciones de bits 8 a 11, un registro de índice llamado X2 en posiciones de bits 12 a 15, un registro de base llamado B2 en posiciones de bits de 16 a 19, un desplazamiento compuesto de dos partes con la primera parte llamada DL2 en posiciones de bits 20 a 31 y la segunda parte llamada DH2 en posiciones de bits 32 a 39.

Esta arquitectura de ordenador tiene un formato de instrucción de tal manera que el código de operación está en posiciones de bits de 0 a 7 y de 40 a 47, un registro objetivo llamado R1 en posiciones de bits de 8 a 11, un registro fuente llamado R3 en posiciones de bits de 12 a 15, un registro de base llamado B2 en posiciones de bits de 16 a 19, un desplazamiento compuesto de dos partes con la primera parte llamada DL2 en posiciones de bits de 20 a 31 y la segunda parte llamada DH2 en posiciones de bits de 32 a 39.

Además, nuestras instrucciones de arquitectura de ordenador que tienen una facilidad de desplazamiento largo tienen un formato de instrucción tal que el código de operación está en posiciones de bits 0 a 7 y 40 a 47, un registro objetivo llamado R1 en posiciones de bits 8 a 11, un valor de máscara llamado M3 en posiciones de bits 12 a 15, un registro base llamado B2 en posiciones de bits 16 a 19, un desplazamiento compuesto de dos partes con la primera parte llamada DL2 en posiciones de bits 20 a 31 y la segunda parte llamada DH2 en posiciones de bits 32 a 39.

Como se ha ilustrado, nuestra arquitectura de ordenador preferida con su facilidad de desplazamiento largo tiene un formato de instrucción de tal manera que el código de operación está en posiciones de bits de 0 a 7 y de 40 a 47, un valor inmediato llamado I2 en posiciones de bits de 8 a 15, un registro base llamado B2 en posiciones de bits de 16 a 19, un desplazamiento compuesto de dos partes con la primera parte llamada DL2 en posiciones de bits 20 a 31 y la segunda parte llamada DH2 en posiciones de bits 32 a 39.

Nuestra arquitectura de ordenador de facilidad de desplazamiento largo funciona de manera efectiva cuando se usan nuevas instrucciones que son creadas que sólo usan el formato de instrucción con el nuevo desplazamiento sin signo de 20 bits.

Una realización específica de nuestra arquitectura de ordenador utiliza instrucciones existentes que tienen los formatos de instrucción que sólo tienen los desplazamientos sin signo de 12 bits y son ahora definidos para estar en los nuevos formatos de instrucción para tener o bien el valor de desplazamiento sin signo de 12 bits existente cuando los 8 bits de alto orden del desplazamiento, el campo DH, son todos cero, o un valor con signo de 20 bits cuando los 8 bits de alto orden del desplazamiento, el campo DH, no es cero.

Un aparato para resumir almacenamiento de un entorno de ordenador, comprendiendo dicho aparato:

medios para especificar, mediante una instrucción, una unidad de almacenamiento que ha de ser resumida; y

medios para resumir datos en la unidad de almacenamiento.

REIVINDICACIONES

1. Un método para ejecutar una instrucción de resumen de mensaje en un sistema de ordenador, comprendiendo la instrucción de resumen de mensaje una función de consulta, comprendiendo el sistema de ordenador una pluralidad de registros generales, un procesador de propósito general en comunicación con una memoria de ordenador, comprendiendo el procesador una o más unidades de ejecución, ejecutando las unidades de ejecución instrucciones buscadas en la memoria del ordenador, **caracterizado** porque, el método comprende las operaciones de: buscar una instrucción de resumen de mensaje; que responde a buscar la instrucción de resumen de mensaje, y determinar a partir de un código de función definido previamente una operación de resumen de mensaje que ha de ser ejecutada, definiendo el código de función definido previamente cualquiera de una operación de resumen de mensaje o una operación de consulta de función; cuando la operación de cálculo de resumen de mensaje determinado que ha de ser ejecutada es una operación de cálculo de resumen de mensajes, realizar la operación de cálculo de resumen de mensaje sobre un operando, comprendiendo la operación de cálculo de resumen de mensaje un algoritmo de aleatoriedad; cuando la operación de cálculo determinada de resumen de mensaje que ha de ser ejecutada es una operación de consulta de función, guardando los bits de la palabra de estado en un bloque de parámetros, correspondiendo los bits de la palabra de estado a uno o más códigos de función instalados en el ordenador.

2. Un método según la reivindicación 1ª, en el que la realización comprende las operaciones adicionales de: x1) obtener un valor de encadenamiento de 20 bytes; x2) obtener un bloque de 64 bytes del operando; x3) usar el valor de encadenamiento de 20 bytes, eligiendo de modo aleatorio directamente el bloque de 64 bytes del operando para producir un nuevo valor de encadenamiento de 20 bytes; x4) repetir las operaciones x2 – x3 para bloques sucesivos del operando; y x5) almacenar el nuevo valor de encadenamientos de 20 bytes producido.

3. Un método según la reivindicación 2ª en el que la operación de elegir aleatoriamente comprende un algoritmo SHA-1.

4. Un método según la reivindicación 2ª que comprende la operación adicional de almacenar un valor hexadecimal de 20 bytes de 6795 2301”EFCD AB89”DCFE’13032 5476”C3D2 E1F0’ como el valor de encadenamiento.

5. Un método según la reivindicación 1ª en el que la instrucción de resumen de mensaje consiste en cualquiera de una instrucción de cálculo de resumen de mensaje intermedio o una instrucción de cálculo de resumen de último mensaje.

6. Un método según la reivindicación 5ª en el que cuando la instrucción de mensaje es una instrucción de cálculo del último mensaje, que realiza las operaciones adicionales de, cuando hay menos de 64 bytes en el operando obtener los menos de 64 bytes; relleno bytes de valor “00” a los menos de 64 bytes obtenidos para crear un operando de 64 bytes.

7. Un método según la reivindicación 1ª, en el que la instrucción de resumen de mensaje comprende: un campo de código óptico; un campo R2, especificando el campo R2 un par de registros generales, comprendiendo el par de registros generales un primer registro general y un segundo registro general, conteniendo el primer registro general una dirección del operando, especificando el segundo registro general una longitud del operando, en el que además el código de función previamente definido es obtenido a partir del primero registro general predeterminado de la pluralidad de registros generales del procesador; y en el que un segundo registro general predeterminado de la pluralidad de registros generales contiene la dirección de un bloque de parámetros en almacenamiento, comprendiendo el bloque de parámetros el valor de encadenamiento, comprendiendo las operaciones adicionales de: obtener la dirección de almacenamiento del bloque de parámetros; obtener inicialmente el valor de encadenamiento de 20 bytes a partir del bloque de parámetros en almacenamiento en la ubicación especificada por la dirección de almacenamiento obtenida, obteniendo inicialmente el código de función previamente definido a partir del primer registro general predeterminado; obtener inicialmente la longitud del operando a partir del segundo registro general; y obtener inicialmente el bloque de 64 bytes del operando en la ubicación especificada por la dirección del operando obtenida.

8. Un método según la reivindicación 7ª en el que el primer registro general predeterminado es el registro general 0 y en el que el segundo registro general predeterminado es el registro general 1.

9. Un método según la reivindicación 7ª que comprende las operaciones adicionales de: aumentar el contenido del primer registro general de acuerdo con un número de bytes del operando procesado en las operaciones realizadas; y disminuir el contenido del segundo registro general de acuerdo al número de bytes del operando procesado en las operaciones realizadas.

10. Un método según la reivindicación 1ª que comprende las operaciones adicionales de: cuando la operación de realización ha sido realizada sólo sobre una parte del operando, ajustar un código de condición de terminación parcial como código de condición, indicando el valor de código de condición de terminación parcial que la operación de realización está incompleta; y cuando la operación de realización ha sido realizada en la totalidad del operando, ajustar un valor de código de condición de terminación normal como código de condición, indicando el valor de código de condición de terminación normal que la operación de realización está completada.

11. Un método según la reivindicación 1ª en el que la instrucción de resumen de mensaje es un formato original a la arquitectura de instrucción del procesador.

12. Un método según la reivindicación 1ª en el que cuando la instrucción de resumen de mensaje no es original de la arquitectura de instrucción de máquina del procesador, el método comprende las operaciones adicionales de: interpretar las instrucciones de resumen de mensaje para identificar una rutina de software predeterminada para emular la operación de la instrucción de resumen de mensaje, comprendiendo la rutina de software predeterminada una pluralidad de instrucciones; y ejecutar la rutina de software predeterminada.

13. Un programa de ordenador que comprende instrucciones para llevar a la práctica todas las operaciones del método según cualquier reivindicación del método precedente, cuando dicho programa de ordenador es ejecutado en un sistema de ordenador.

14. Un aparato que comprende medios destinados a realizar todas las operaciones del método según cualquiera de las reivindicaciones 1ª a 12ª.

KIMD R_1, R_2 [RRE]

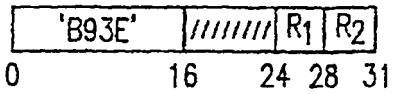


FIG.1

KLMD R_1, R_2 [RRE]

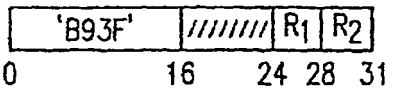


FIG.2

CÓDI-GO	FUNCIÓN	TAMAÑO DE BLOQUE DE PARAM. (BYTES)	TAMAÑO DE BLOQUE DE DATOS (BYTES)
0	KIMD-QUERY	16	—
1	KIMD-SHA-1	20	64
EXPLICACIÓN: -- NO APLICABLE			

FIG.3

CÓDI-GO	FUNCIÓN	TAMAÑO DE BLOQUE DE PARÁM. (BYTES)	TAMAÑO DE BLOQUE DE DATOS (BYTES)
0	KLMD-QUERY	16	—
1	KLMD-SHA-1	28	64
EXPLICACIÓN: -- NO APLICABLE			

FIG.4

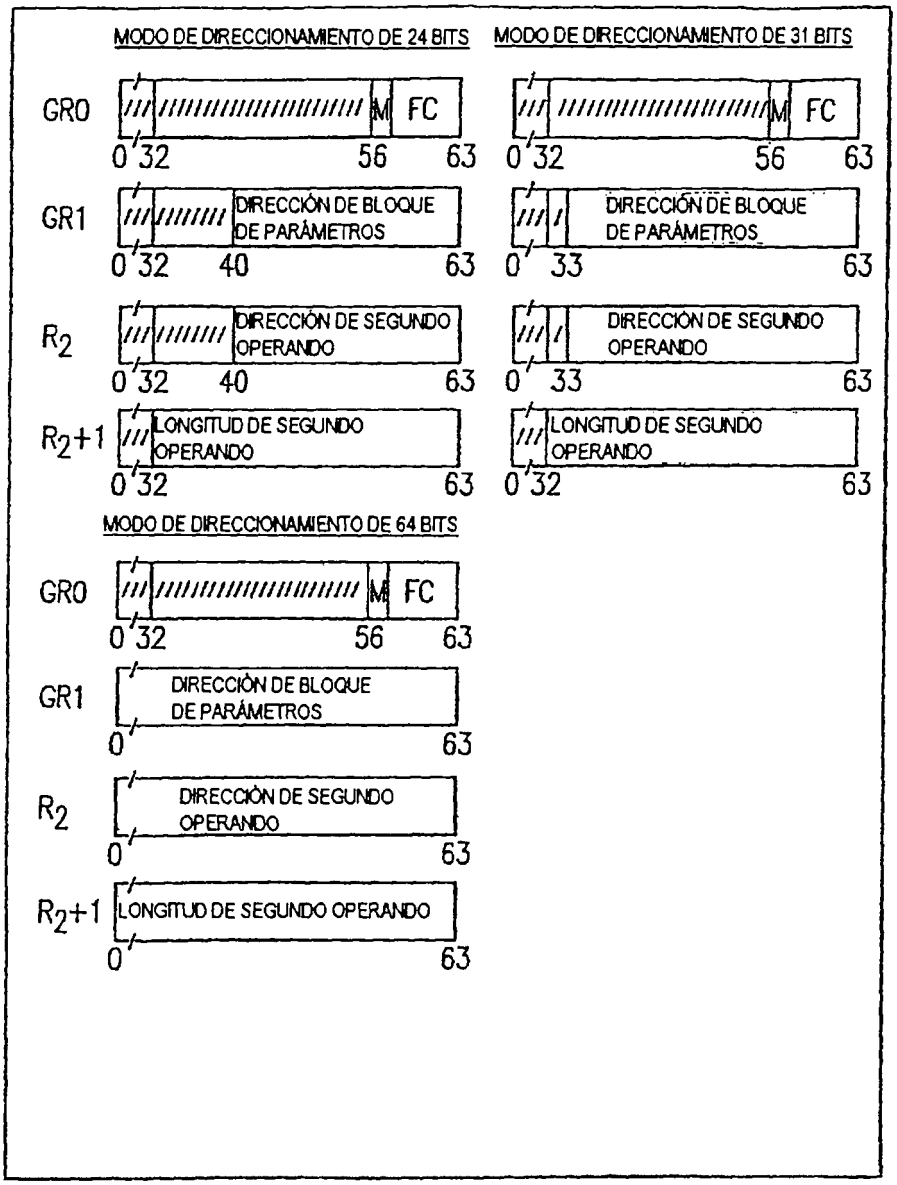

MODO DE DIRECCIONAMIENTO DE 24 BITS
MODO DE DIRECCIONAMIENTO DE 31 BITS
GR0
M
FC
0 32 56 63
GR1
DIRECCIÓN DE BLOQUE DE PARÁMETROS
0 32 40 63
0 33 63
R_2
DIRECCIÓN DE SEGUNDO OPERANDO
R_2+1
LONGITUD DE SEGUNDO OPERANDO
MODO DE DIRECCIONAMIENTO DE 64 BITS

FIG.5

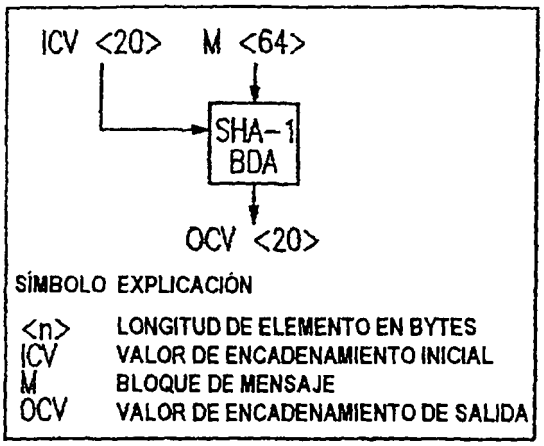


FIG.6

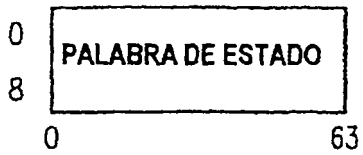


FIG.7

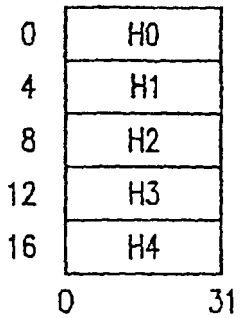


FIG.8

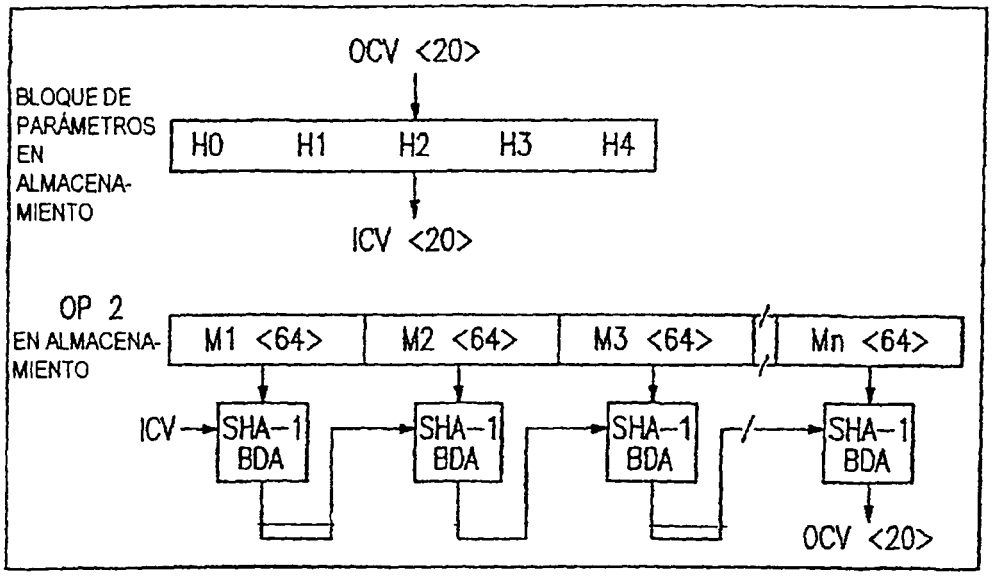


FIG.9

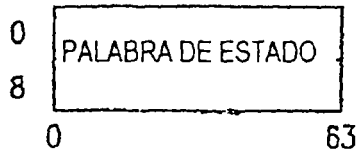


FIG.10

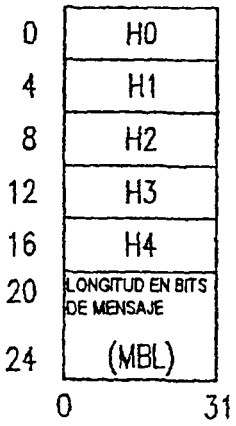


FIG.11

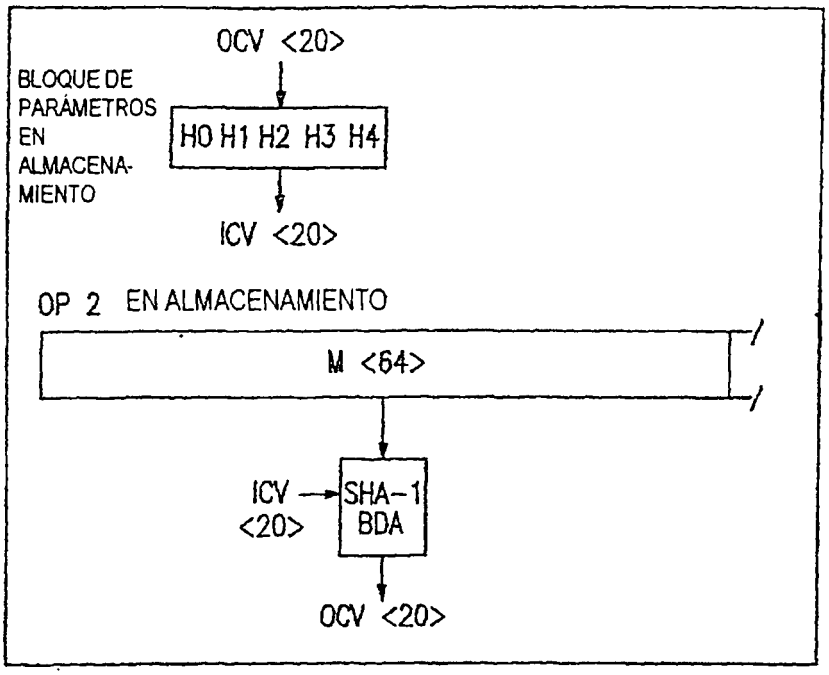

OCV <20>
BLOQUE DE PARÁMETROS EN ALMACENA-MIENTO
H0 H1 H2 H3 H4
ICV <20>
OP 2 EN ALMACENAMIENTO
M <64>
ICV <20>
SHA-1 BDA
OCV <20>

FIG.12

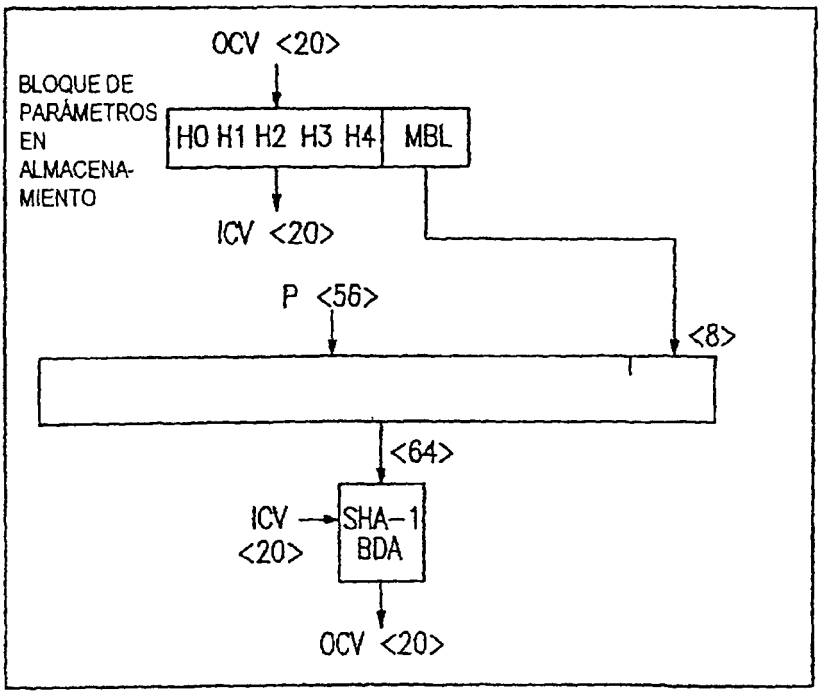

OCV <20>
BLOQUE DE PARÁMETROS EN ALMACENA-MIENTO
H0 H1 H2 H3 H4
MBL
ICV <20>
P <56>
<8>
<64>
ICV <20>
SHA-1 BDA
OCV <20>

FIG.13

FIG.14

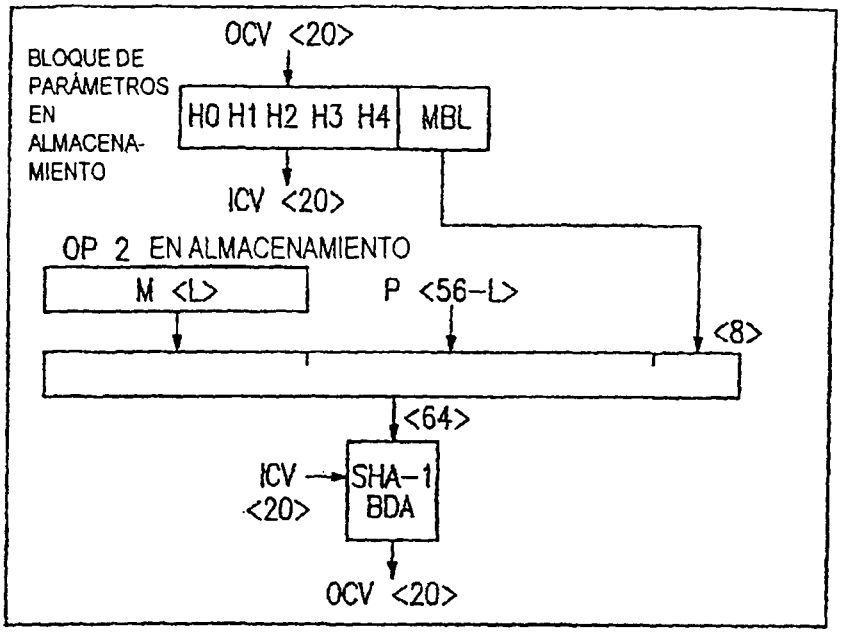
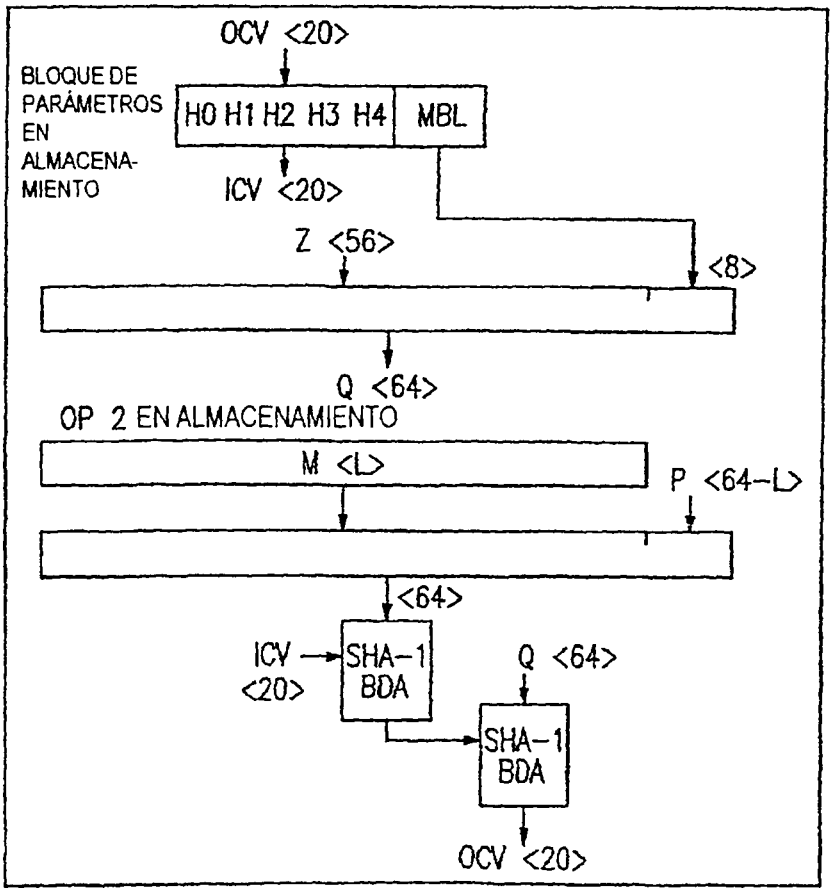

OCV <20>
BLOQUE DE PARÁMETROS EN ALMACENA-MIENTO
H0 H1 H2 H3 H4
MBL
ICV <20>
OP 2 EN ALMACENAMIENTO
M <L>
P <56-L>
<8>
<64>
ICV <20>
SHA-1 BDA
OCV <20>

FIG.15


OCV <20>
BLOQUE DE PARÁMETROS EN ALMACENA-MIENTO
H0 H1 H2 H3 H4
MBL
ICV <20>
Z <56>
<8>
Q <64>
OP 2 EN ALMACENAMIENTO
M <L>
P <64-L>
<64>
ICV <20>
SHA-1 BDA
Q <64>
SHA-1 BDA
OCV <20>

1.–6. EXCEPCIONES CON LA MISMA PRIORIDAD QUE LA PRIORIDAD DE CONDICIONES DE INTERRUPCIÓN DE PROGRAMA PARA EL CASO GENERAL

7.A EXCEPCIONES DE ACCESO PARA SEGUNDA MEDIA PALABRA DE INSTRUCCIÓN

7.B EXCEPCIÓN DE OPERACIÓN

8. EXCEPCIÓN DE ESPECIFICACIÓN DEBIDO A CÓDIGO DE FUNCIÓN INVÁLIDO O NÚMERO DE REGISTRO INVÁLIDO

9. EXCEPCIÓN DE ESPECIFICACIÓN DEBIDO A LONGITUD DE OPERANDO INVÁLIDA

10. CÓDIGO DE CONDICIÓN O DEBIDO A LONGITUD ORIGINALMENTE CERO DEL SEGUNDO OPERANDO

11. EXCEPCIONES DE ACCESO PARA UN ACCESO AL BLOQUE DE PARÁMETROS, PRIMER O SEGUNDO OPERANDO

12. CÓDIGO DE CONDICIÓN O DEBIDO A TERMINACIÓN NORMAL (LONGITUD DE SEGUNDO OPERANDO ORIGINALMENTE NO CERO, PERO ESCALONADO A CERO)

13. CÓDIGO DE CONDICIÓN 3 DEBIDO A TERMINACIÓN PARCIAL (LONGITUD DE SEGUNDO OPERANDO AÚN NO CERO)

FIG. 16

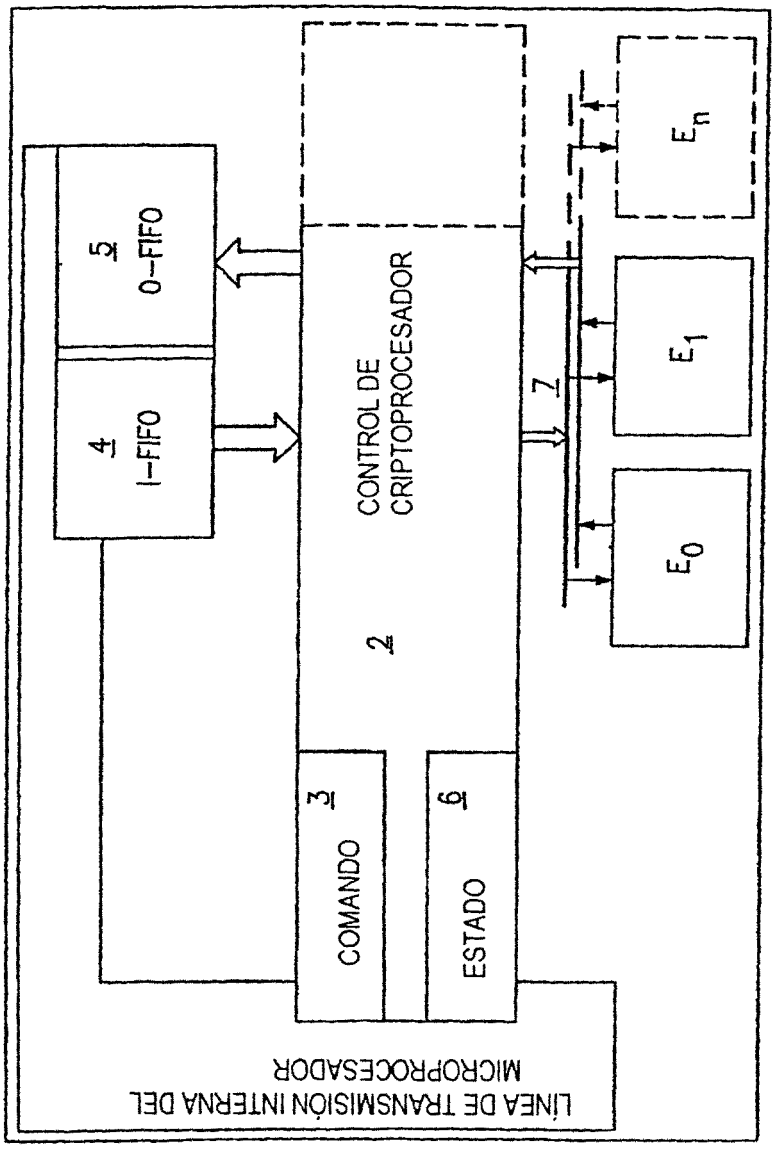

LÍNEA DE TRANSMISIÓN INTERNA DEL MICROPROCESADOR
3 COMANDO
6 ESTADO
2
CONTROL DE CRIPTOPROCESADOR
4 I-FIFO
5 O-FIFO
7
E_0
E_1
E_n

FIG.17

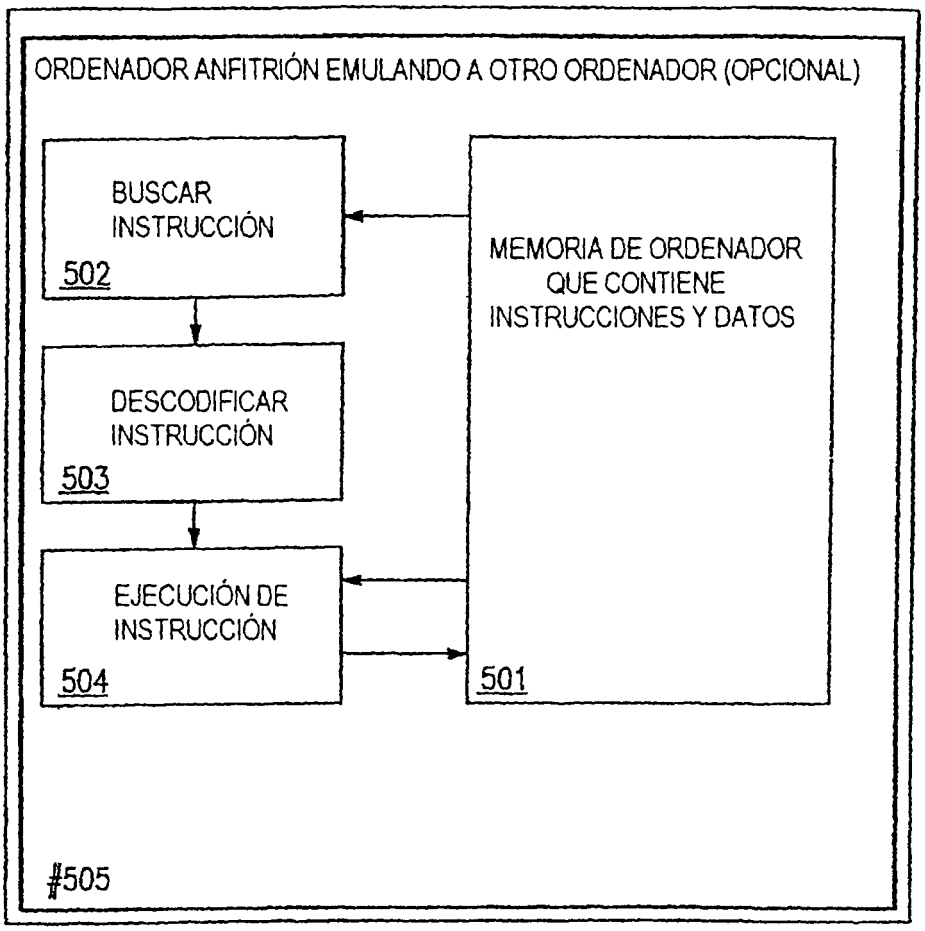

ORDENADOR ANFITRIÓN EMULANDO A OTRO ORDENADOR (OPCIONAL)
BUSCAR INSTRUCCIÓN
502
DESCODIFICAR INSTRUCCIÓN
503
EJECUCIÓN DE INSTRUCCIÓN
504
MEMORIA DE ORDENADOR QUE CONTIENE INSTRUCCIONES Y DATOS
501
#505

FIG.18