(11) Número de Publicação: **PT 2430555 E**

(51) Classificação Internacional:
***G06F 13/16*** (2013.01) ***G06F 12/10*** (2013.01)

(12) **FASCÍCULO DE PATENTE DE INVENÇÃO**

(22) Data de pedido: **2010.11.08**

(30) Prioridade(s): **2010.06.23 US 821170**

(43) Data de publicação do pedido: **2012.03.21**

(45) Data e BPI da concessão: **2013.08.28**
**189/2013**

(73) Titular(es):
**INTERNATIONAL BUSINESS MACHINES CORPORATION**
**NEW ORCHARD PLACE ARMONK, NY 10504 US**

(72) Inventor(es):

| | |
|---|---|
| **DAVID CRADDOCK** | **US** |
| **THOMAS GREGG** | **US** |
| **DAN GREINER** | **US** |
| **ERIC NORMAN LAIS** | **US** |

(74) Mandatário:
**LUÍS MANUEL DE ALMADA DA SILVA CARVALHO**
**RUA VÍCTOR CORDON, 14 1249-103 LISBOA** **PT**

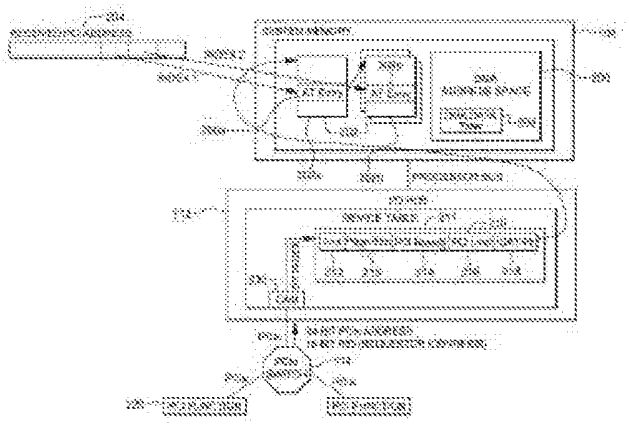
(54) Epígrafe: **TRADUÇÃO DE ENDEREÇOS DE ENTRADA/SAÍDA PARA ENDEREÇOS DE MEMÓRIA**

(57) Resumo:
UM ENDEREÇO FORNECIDO NUM PEDIDO EMITIDO POR UM ADAPTADOR É CONVERTIDO NUM ENDEREÇO DIRETAMENTE UTILIZÁVEL NO ACESSO À MEMÓRIA DE SISTEMA. O ENDEREÇO INCLUI UMA PLURALIDADE DE BITS, EM QUE A PLURALIDADE DE BITS INCLUI UMA PRIMEIRA PORÇÃO DE BITS E UMA SEGUNDA PORÇÃO DE BITS. A SEGUNDA PORÇÃO DE BITS É UTILIZADA PARA INDEXAR EM UM OU MAIS NÍVEIS DE TABELAS DE TRADUÇÃO DE ENDEREÇOS PARA REALIZAR A TRADUÇÃO, AO PASSO QUE A PRIMEIRA PORÇÃO DE BITS É IGNORADA PARA A CONVERSÃO. A PRIMEIRA PORÇÃO DE BITS É UTILIZADA PARA A VALIDAÇÃO DO ENDEREÇO.

# RESUMO

## "TRADUÇÃO DE ENDEREÇOS DE ENTRADA/SAÍDA PARA ENDEREÇOS DE MEMÓRIA"

Um endereço fornecido num pedido emitido por um adaptador é convertido num endereço diretamente utilizável no acesso à memória de sistema. O endereço inclui uma pluralidade de *bits*, em que a pluralidade de *bits* inclui uma primeira porção de *bits* e uma segunda porção de *bits*. A segunda porção de *bits* é utilizada para indexar em um ou mais níveis de tabelas de tradução de endereços para realizar a tradução, ao passo que a primeira porção de *bits* é ignorada para a conversão. A primeira porção de *bits* é utilizada para a validação do endereço.



# DESCRIÇÃO

## "TRADUÇÃO DE ENDEREÇOS DE ENTRADA/SAÍDA PARA ENDEREÇOS DE MEMÓRIA"

### ANTECEDENTES

Esta invenção refere-se, em geral, à tradução de endereços num ambiente informático, e em particular, à tradução de um endereço de entrada/saída para um endereço de memória utilizável no acesso à memória de sistema do ambiente.

A memória de sistema é acessível através de pedidos de leitura e de escrita. Estes pedidos podem ser provenientes de vários componentes de um ambiente informático, incluindo unidades centrais de processamento, assim como adaptadores. Cada pedido inclui um endereço que irá ser utilizado para aceder à memória de sistema. Este endereço, no entanto, normalmente não tem uma correspondência um-para-um com uma localização física na memória de sistema. Portanto, a tradução de endereços é realizada.

A tradução de endereços é utilizada para traduzir um endereço que é fornecido de uma forma não diretamente utilizável no acesso à memória de sistema para uma outra forma que pode ser utilizada diretamente no acesso a uma

localização física na memória de sistema. Por exemplo, um endereço virtual incluído num pedido fornecido por uma unidade central de processamento é traduzido para um endereço real ou absoluto na memória de sistema. Como exemplo adicional, um endereço de Interconexão de Componentes Periféricos (*Peripheral Component Interconnect* – PCI), fornecido num pedido proveniente de um adaptador pode ser traduzido para um endereço absoluto na memória de sistema.

A Publicação dos EUA No. 2008/0168208 A1, publicada em 10 de julho de 2008, Gregg, "*I/O Adapter LPAR Isolation In a Hypertransport Environment With Assigned Memory Space Indexing A TVT Via Unit IDs*" descreve um sistema e método de processamento de dados para isolamento de uma pluralidade de adaptadores de I/O no sistema. O sistema de processamento de dados compreende também um conjunto de processadores, uma ponte anfitriã (*host bridge*), e um barramento (*bus*) de sistema que liga o conjunto de processadores e a *host bridge*. Cada um dos adaptadores de I/O tem um ID respetivo e envia comandos para a *host bridge* que incluem um ou mais dos IDs dos adaptadores de I/O. Na forma de realização preferida, estes IDs são IDs de Unidade (*Unit IDs*) definidos como *HyperTransport*, e os comandos emitidos pelos Adaptadores de I/O incluem um campo de *Unit ID* que contém um ou mais dos *Unit IDs* dos Adaptadores de I/O. Cada um dos *Unit IDs* é utilizado para indexar um TVT para identificar espaços de memória de sistema únicos e independentes.

A Publicação dos EUA No. 2009/0182966 A1, publicada em 16 de julho de 2009, Greiner et al., "*Dynamic Address Translation with Frame Management*" divulga uma função de gestão de tabela definida para uma arquitetura de máquina de um sistema informático. Numa forma de realização, uma instrução de gestão de tabela é obtida, a qual identifica um primeiro e um segundo registo geral. O primeiro registo geral contém um campo de gestão de tabela com um campo da chave com *bits* de proteção de acesso e uma indicação de tamanho de bloco. Se a indicação de tamanho de bloco indicar um bloco grande então um endereço do operando de um bloco grande de dados é obtido a partir do segundo registo geral. O bloco grande de dados tem uma pluralidade de blocos pequenos cada um dos quais está associado com uma chave correspondente de armazenamento tendo uma pluralidade de *bits* de proteção de acesso da chave de armazenamento. Se a indicação de tamanho de bloco indicar um bloco grande, os *bits* de proteção de acesso da chave de armazenamento de cada chave de armazenamento correspondente de cada bloco pequeno dentro do bloco grande são definidos com os *bits* de proteção de acesso do campo da chave.

US 6 658 521 divulga método e aparelho para tradução de endereços sobre o *bus* PCI sobre uma rede *infiniband*.

US 2008/0114906 A1 divulga acessos de sistema especiais mapeados na memória, controlados eficientemente.

**BREVE SUMÁRIO**

As deficiências da técnica anterior são ultrapassadas e vantagens adicionais são proporcionadas através do fornecimento de um método tal como reivindicado na reivindicação 1, e produto de programa de computador e sistema correspondentes.

Características e vantagens adicionais são realizadas através das técnicas da presente invenção. Outras formas de realização e aspetos da invenção são aqui descritos em pormenor e são considerados parte da invenção reivindicada.

**DESCRIÇÃO RESUMIDA DAS DIVERSAS VISTAS DOS DESENHOS**

Um ou mais aspetos da presente invenção são particularmente salientados e distintamente reivindicados como exemplos nas reivindicações na conclusão da especificação. O anterior e outros objetivos, características e vantagens da invenção são evidentes a partir da seguinte descrição detalhada tomada em conjunto com os desenhos acompanhantes nos quais:

A FIG. 1 representa uma forma de realização de um ambiente informático para incorporar e utilizar um ou mais aspetos da presente invenção;

A FIG. 2 representa uma forma de realização de

detalhes adicionais sobre a memória de sistema e o *hub* de I/O do sistema da FIG. 1, de acordo com um aspeto da presente invenção;

A FIG. 3A representa uma forma de realização de uma vista geral da lógica para registar um espaço de endereçamento DMA (Acesso Direto de Memória – *Direct Memory Access*) para um adaptador, de acordo com um aspeto da presente invenção;

A FIG. 3B representa uma forma de realização de vários detalhes do registo do espaço de endereçamento DMA para o adaptador, de acordo com um aspeto da presente invenção;

A FIG. 4 representa uma forma de realização da lógica para processar uma operação DMA, de acordo com um aspeto da presente invenção;

A FIG. 5A representa um exemplo dos níveis de tradução empregues quando um endereço inteiro é utilizado para indexar em tabelas de tradução de endereços para traduzir o endereço e para aceder à página;

A FIG. 5B representa um exemplo de níveis de tradução empregues quando uma parte do endereço é ignorada durante a indexação nas tabelas de tradução de endereços, de acordo com um aspeto da presente invenção;

A FIG. 5C representa exemplos de vários formatos compatíveis com CPU DAT utilizáveis de acordo com um ou mais aspetos da presente invenção;

A FIG. 5D representa exemplos de vários formatos estendidos de tradução de endereços de I/O utilizáveis de acordo com um ou mais aspetos da presente invenção;

A FIG. 6A representa uma forma de realização de uma instrução de Modificar Controlos de Função PCI utilizada de acordo com um aspeto da presente invenção;

A FIG. 6B representa uma forma de realização de um campo utilizado pela instrução de Modificar Controlos de Função PCI da FIG. 6A, de acordo com um aspeto da presente invenção;

A FIG. 6C representa uma forma de realização de um outro campo utilizado pela instrução de Modificar Controlos de Função PCI da FIG. 6A, de acordo com um aspeto da presente invenção;

A FIG. 6D representa uma forma de realização dos conteúdos de um bloco de informação de função (*Function Information Block* – FIB) utilizado de acordo com um aspeto da presente invenção;

A FIG. 7 representa uma forma de realização de uma visão geral da lógica da instrução Modificar Controlos

de Função PCI, de acordo com um aspeto da presente invenção;

A FIG. 8 representa uma forma de realização da lógica associada a uma operação de registo de parâmetros de tradução de endereços de I/O que pode ser especificada por meio da instrução Modificar Controlos de Função PCI, de acordo com um aspeto da presente invenção;

A FIG. 9 representa uma forma de realização da lógica associada a uma operação de cancelamento do registo de parâmetros de tradução de endereços de I/O que pode ser especificada pela instrução Modificar Controlos de Função PCI, de acordo com um aspeto da presente invenção;

A FIG. 10 representa uma forma de realização de um produto de programa de computador que incorpora um ou mais aspetos da presente invenção;

A FIG. 11 representa uma forma de realização de um sistema informático anfitrião (*host*) para incorporar e utilizar um ou mais aspetos da presente invenção;

A FIG. 12 representa um exemplo adicional de um sistema informático para incorporar e utilizar um ou mais aspetos da presente invenção;

A FIG. 13 representa um outro exemplo de um sistema informático compreendendo uma rede de computadores

para incorporar e utilizar um ou mais aspetos da presente invenção;

A FIG. 14 representa uma forma de realização de vários elementos de um sistema informático para incorporar e utilizar um ou mais aspetos da presente invenção;

A FIG. 15A representa uma forma de realização da unidade de execução do sistema informático da FIG. 14 para incorporar e utilizar um ou mais aspetos da presente invenção;

A FIG. 15B representa uma forma de realização da unidade de ramificação do sistema informático da FIG. 14 para incorporar e utilizar um ou mais aspetos da presente invenção;

A FIG. 15C representa uma forma de realização da unidade de carga/armazenamento do sistema informático da FIG. 14 para incorporar e utilizar um ou mais aspetos da presente invenção; e

A FIG. 16 representa uma forma de realização de um sistema informático *host* emulado para incorporar e utilizar um ou mais aspetos da presente invenção.

**DESCRIÇÃO DETALHADA**

De acordo com um aspeto da presente invenção, é

fornecida uma capacidade para traduzir endereços num ambiente informático. Num exemplo, os endereços a serem traduzidos são endereços fornecidos por um adaptador (aqui referido como endereços de entrada/saída (Input/Output - I/O), que irão ser traduzidos para endereços utilizáveis no acesso à memória de sistema. Para realizar a tradução de endereços, um número de níveis de tradução de endereços é utilizado, e o número de níveis baseia-se, por exemplo, no tamanho do espaço de endereçamento de memória atribuído ao adaptador, um tamanho de uma ou mais tabelas de tradução de endereços utilizadas na tradução, e/ou um tamanho da página (ou outra unidade de memória), a ser acedida.

O endereço a ser traduzido inclui uma pluralidade de *bits*, e numa forma de realização, apenas uma porção dos referidos *bits* é utilizada para indexar nas tabelas de tradução de endereços para obter o endereço traduzido. Os outros *bits* são ignorados para tradução. Por exemplo, o endereço inclui *bits* de ordem superior e *bits* de ordem inferior (de acordo com o tamanho do espaço de endereçamento atribuído). Neste exemplo, os *bits* de ordem inferior são utilizados para indexar nas tabelas de tradução de endereços, incluindo a tabela de páginas, e para indexar na própria página. Os *bits* de ordem superior (independentemente do valor, ou seja, zero ou diferente de zero) são ignorados para tradução e não são utilizados para indexar nas tabelas de tradução de endereços. Isto reduz o número de níveis de tabelas de tradução utilizados para traduzir os endereços. (A indicação de *bits* de ordem

inferior e de ordem superior é independente da forma como os *bits* são numerados).

Num exemplo, a tradução é realizada, proporcionando ao mesmo tempo proteção adequada para um servidor da classe *enterprise*, tal como um servidor *System z®*. Como exemplo, um endereço completo (por exemplo, a totalidade do endereço de 64 *bits*) é utilizado no acesso à memória, no entanto, apenas uma porção do endereço é utilizada para tradução. Uma outra porção do endereço, que é ignorada para tradução, é utilizada para validação (por exemplo, uma verificação de intervalo), juntamente com pelo menos uma porção do endereço utilizada para tradução. Ao ser utilizada apenas uma porção do endereço para tradução, as pesquisas de tradução de endereços são minimizadas. Isto é conseguido através da realização de um número de níveis de tradução, com base por exemplo no tamanho do espaço de endereçamento DMA registado para acesso pelo adaptador, em vez do tamanho do próprio endereço.

Uma forma de realização de um ambiente informático para incorporar e utilizar um ou mais aspetos da presente invenção é descrito com referência à FIG. 1. Num exemplo, um ambiente informático 100 é um servidor *System z®* proposto pela *International Business Machines Corporation*. O *System z®* é baseado na *z/Architecture®* proposta pela *International Business Machines Corporation*. Detalhes sobre a *z/Architecture®* são descritos numa publicação da IBM® intitulada, "*z/Architecture Principles*

*of Operation*" Publicação IBM No. SA22-7832-07, fevereiro de 2009. IBM®, *System z*® e *z/Architecture*® são marcas comerciais registadas da *International Business Machines Corporation*, Armonk, Nova Iorque. Outros nomes aqui utilizados podem ser marcas registadas, marcas comerciais ou nomes de produtos da *International Business Machines Corporation* ou de outras empresas.

Num exemplo, o ambiente informático 100 inclui uma ou mais unidades centrais de processamento (CPUs) 102 acopladas a uma memória de sistema 104 (também conhecida como memória principal) por meio de um controlador de memória 106. Para aceder à memória de sistema 104, uma unidade central de processamento 102 emite um pedido de leitura ou de escrita que inclui um endereço utilizado para aceder à memória de sistema. O endereço incluído no pedido normalmente não é utilizável diretamente para aceder à memória de sistema, e portanto é traduzido para um endereço que pode ser utilizado diretamente no acesso à memória de sistema. O endereço é traduzido através de um mecanismo de tradução (XLATE) 108. Por exemplo, o endereço é traduzido a partir de um endereço virtual para um endereço real ou absoluta utilizando, por exemplo, tradução de endereço dinâmica (*Dynamic Address Translation* – DAT).

O pedido, incluindo o endereço traduzido, é recebido pelo controlador de memória 106. Num exemplo, o controlador de memória 106 é constituído por *hardware* e é utilizado para arbitrar o acesso à memória de sistema e

para manter a consistência da memória. Esta arbitragem é realizada para pedidos recebidos a partir dos CPUs 102, bem como para pedidos recebidos a partir de um ou mais adaptadores 110. Tal como as unidades centrais de processamento, os adaptadores emitem pedidos para a memória de sistema 104 para obter acesso à memória de sistema.

Num exemplo, o adaptador 110 é um adaptador de Interconexão de Componentes Periféricos (*Peripheral Component Interconnect* – PCI) ou *PCI Express* (PCIe), que inclui uma ou mais funções PCI. Uma função PCI emite um pedido que requer acesso à memória de sistema. O pedido é encaminhado para um concentrador (*hub*) de entrada/saída 112 (por exemplo, um *hub* PCI) através de um ou mais comutadores (*switches*) (por exemplo, *switches* PCIe) 114. Num exemplo, o *hub* de entrada/saída é constituído por *hardware*, incluindo uma ou mais máquinas de estado.

Tal como aqui utilizado, o termo adaptador inclui qualquer tipo de adaptador (por exemplo, um adaptador de armazenamento, adaptador de rede, adaptador de processamento, adaptador PCI, adaptador criptográfico, outro tipo de adaptadores de entrada/saída, etc.). Numa forma de realização, um adaptador inclui uma função de adaptador. No entanto, em outras formas de realização, um adaptador pode incluir uma pluralidade de funções de adaptador. Um ou mais aspetos da presente invenção são aplicáveis se um adaptador incluir uma função de adaptador ou uma pluralidade de funções de adaptador. Além disso, nos

exemplos aqui apresentados, o adaptador é utilizado de forma intercambiável com função de adaptador (por exemplo, função PCI), a menos que seja indicado de forma diferente.

O *hub* de entrada/saída inclui, por exemplo, um complexo de raíz 116 que recebe o pedido de um comutador (*switch*). O pedido inclui um endereço de entrada/saída a ser traduzido e, por conseguinte, o complexo de raíz fornece o endereço para uma unidade de tradução de endereços e de proteção 118. Esta unidade é, por exemplo, uma unidade de *hardware* que traduz o endereço de I/O para um endereço diretamente utilizável para aceder à memória de sistema 104, tal como descrito em maior detalhe abaixo.

O pedido iniciado a partir do adaptador, incluindo o endereço traduzido, é fornecido ao controlador de memória 106 através de, por exemplo, um *bus* de I/O-para-memória 120. O controlador de memória realiza a sua arbitragem e encaminha o pedido com o endereço traduzido para a memória de sistema no momento apropriado.

Detalhes adicionais relativos à memória de sistema e ao *hub* de entrada/saída são descritos com referência à FIG. 2. Nesta forma de realização, o controlador de memória não é mostrado. No entanto, o *hub* de I/O pode ser acoplado à memória de sistema diretamente ou através de um controlador de memória. Num exemplo, a memória de sistema 104 inclui um ou mais espaços de endereçamento 200. Um espaço de endereçamento é uma porção

particular de memória de sistema que foi atribuída a um determinado componente do ambiente informático, tal como um adaptador específico. Num exemplo, o espaço de endereçamento é acessível por acesso direto à memória (*Direct Memory Access* – DMA), iniciado pelo adaptador, e portanto, o espaço de endereçamento é referido nos exemplos aqui como um espaço de endereçamento DMA. No entanto, noutros exemplos, o acesso direto à memória não é utilizado para aceder ao espaço de endereçamento.

Além disso, num exemplo, a memória de sistema 104 inclui tabelas de tradução de endereços 202 utilizadas para traduzir um endereço a partir de um que não é diretamente utilizável para aceder à memória de sistema para um endereço que é diretamente utilizável. Numa forma de realização, existe uma ou mais tabelas de tradução de endereços atribuídas a um espaço de endereçamento DMA, e essas uma ou mais tabelas de tradução de endereços são configuradas com base, por exemplo, no tamanho do espaço de endereçamento para o qual estão atribuídas, no tamanho das próprias tabelas de endereços de tradução, e/ou no tamanho da página (ou outra unidade de memória).

Num exemplo, existe uma hierarquia de tabelas de tradução de endereços. Por exemplo, tal como mostrado na FIG. 2, existe uma tabela de primeiro nível 202a (por exemplo, uma tabela de segmentos) apontada por um apontador IOAT 218 (descrito abaixo), e uma segunda tabela de nível inferior 202b (por exemplo, uma tabela de páginas) apontada

por uma entrada 206a da tabela de primeiro nível. Um ou mais *bits* de um endereço recebido 204 são utilizados para indexar na tabela 202a para localizar uma determinada entrada 206a, o que indica uma determinada tabela de nível inferior 202b. Em seguida, um ou mais *bits* do endereço 204 são utilizados para localizar uma determinada entrada 206b nessa tabela. Neste exemplo, essa entrada fornece o endereço utilizado para localizar a página correta e *bits* adicionais no endereço 204 são utilizados para localizar uma determinada localização 208 na página para realizar uma transferência de dados. Ou seja, o endereço na entrada 206b e os *bits* selecionados do endereço PCI 204 recebido são utilizados para fornecer o endereço diretamente utilizável para aceder à memória de sistema. Por exemplo, o endereço diretamente utilizável é formado a partir de uma concatenação de *bits* de ordem superior do endereço na entrada 206b (por exemplo, os *bits* 63:12, num exemplo de página com 4k) e os *bits* de ordem inferior selecionados a partir do endereço PCI recebido (por exemplo, *bits* 11:0 para uma página com 4k).

Num exemplo, é um sistema operativo que atribui um espaço de endereçamento DMA a um determinado adaptador. Esta atribuição é realizada através de um processo de registo, o que provoca uma inicialização (através de, por exemplo, *software* de confiança), de uma entrada de tabela 210 de um dispositivo para esse adaptador. A entrada de tabela do dispositivo está localizada numa tabela de dispositivo 211 localizada no *hub* de I/O 112. Por exemplo,

a tabela de dispositivo 211 está localizada dentro da unidade de tradução de endereços e de proteção do *hub* de I/O.

Num exemplo, a entrada de tabela do dispositivo 210 inclui um número de campos, tais como os seguintes:

Formato 212: Este campo inclui uma pluralidade de *bits* para indicar várias informações, incluindo, por exemplo, o formato de tradução de endereços (incluindo o nível) de uma tabela de nível superior das tabelas de tradução de endereços (por exemplo, no exemplo acima, a tabela de primeiro nível);

Tamanho da Página 213: Este campo indica o tamanho de uma página (ou outra unidade de memória) a ser acedida;

Endereço base PCI 214 e limite PCI 216: Estes valores fornecem um intervalo utilizado para definir um espaço de endereçamento DMA e verificar se um endereço recebido (por exemplo, um endereço PCI) é válido; e

Apontador IOAT (*Input/Output Address Translation* – Tradução de Endereços de Entrada/Saída) 218: Este campo inclui um apontador para o nível mais elevado da tabela de tradução de endereços utilizada para o espaço de endereçamento DMA.

Noutras formas de realização, a DTE pode incluir mais, menos ou diferente informação.

Numa forma de realização, a entrada de tabela de dispositivo a ser utilizada numa tradução específica está localizada utilizando um identificador solicitante (*Requestor Identifier* – RID), localizado num pedido emitido por uma função PCI 220 associada com um adaptador (e/ou por uma porção do endereço). O *id* do solicitante (por exemplo, um valor de 16 *bits* especificando, por exemplo, um número de *bus*, número de dispositivo e número de função) está incluído no pedido, bem como o endereço de I/O (por exemplo, um endereço PCIe de 64 *bits*) a ser utilizado para aceder à memória de sistema. O pedido, incluindo o RID e o endereço de I/O, são fornecidos para, por exemplo, uma memória de conteúdo endereçável (*Contents Addressable Memory* – CAM) 230 através de, por exemplo, um comutador (*switch*) 114, que é utilizado para proporcionar um valor de índice. Por exemplo, a CAM inclui múltiplas entradas, com cada entrada correspondente a um índice de tabela de dispositivo. Cada entrada CAM inclui o valor de um RID. Se, por exemplo, o RID recebido corresponde ao valor contido numa entrada na CAM, o índice da tabela de dispositivo correspondente é utilizado para localizar a entrada de tabela de dispositivo. Isto é, a saída da CAM é utilizada para indexar na tabela de dispositivo 211 para localizar a entrada de tabela de dispositivo 210. Se não existe correspondência, o pacote recebido é descartado, sem realização de acesso à memória de sistema. (Noutras formas

de realização, uma CAM ou outra pesquisa não é necessária e o RID é utilizado como índice.)

Subsequentemente, os campos dentro da entrada de tabela de dispositivo são utilizados para garantir a validade do endereço e a configuração das tabelas de tradução de endereços. Por exemplo, o endereço de entrada no pedido é verificado pelo *hardware* do *hub* de I/O (por exemplo, a unidade de tradução de endereços e de proteção) para garantir que está dentro dos limites definidos pelo endereço base PCI 214 e pelo limite PCI 216 armazenados na entrada de tabela de dispositivo localizados utilizando o RID do pedido que forneceu o endereço. Isto assegura que o endereço está dentro do intervalo previamente registado e para o qual as tabelas de tradução de endereços estão validamente configuradas.

De acordo com um aspeto da presente invenção, numa forma de realização, para traduzir um endereço de I/O (isto é, um endereço fornecido por um adaptador ou outro componente de um subsistema de I/O) para um endereço de memória de sistema (isto é, um endereço diretamente utilizável para aceder à memória de sistema), inicialmente, um processo de registo é realizado. Este processo de registo regista um determinado espaço de endereçamento com um solicitante específico, tal como um adaptador ou função de adaptador específico. Um exemplo de uma vista geral deste processo de registo é descrito com referência à FIG. 3A.

Inicialmente, um sistema operativo em execução dentro de uma das unidades centrais de processamento acopladas à memória de sistema determina o tamanho e localização do espaço de endereçamento a que o adaptador deve aceder, ETAPA 300. Num exemplo, o tamanho do espaço de endereçamento é determinado pelo endereço base PCI e limite PCI definidos pelo sistema operativo. O sistema operativo determina a base e o limite utilizando um ou mais critérios. Por exemplo, se o sistema operativo deseja que os endereços PCI mapeiem diretamente para endereços virtuais de CPU, então a base e o limite são definidos como tal. Num outro exemplo, se é desejado um isolamento adicional entre adaptadores e/ou imagens do sistema operativo, então os endereços a serem utilizados são selecionados para fornecer espaços de endereçamento não sobrepostos e disjuntos. A localização também é especificada pelo sistema operativo, e baseia-se, por exemplo, nas características do adaptador.

A partir daí, uma ou mais tabelas de tradução de endereços são criadas para cobrir esse espaço de endereçamento DMA, ETAPA 302. Como exemplos, as tabelas podem ser compatíveis com as tabelas de tradução de endereços de CPU ou um formato único pode ser fornecido que é suportado pelo *hub* de entrada/saída. Num exemplo, a criação inclui a construção das tabelas e a colocação dos endereços apropriados dentro das entradas das tabelas. Como exemplo, uma das tabelas de tradução é uma tabela de página com 4k tendo 512 entradas de 64 *bits*, e cada entrada inclui

um endereço de página de 4k compatível com o espaço de endereçamento atribuído.

Em seguida, o espaço de endereçamento DMA é registado para o adaptador, ETAPA 304, tal como descrito em maior detalhe com referência à FIG. 3B. Neste exemplo, é assumido que existe uma função PCI por adaptador, e portanto, um ID de solicitante por adaptador. Esta lógica é realizada, por exemplo, por uma unidade central de processamento ligada à memória de sistema, que responde a um pedido do sistema operativo.

Inicialmente, numa forma de realização, uma entrada de tabela de dispositivo disponível é selecionada por forma a corresponder ao ID do solicitante do adaptador, ETAPA 310. Isto é, o ID do solicitante irá ser utilizado para localizar uma entrada de tabela de dispositivo.

Além disso, o endereço base PCI e o limite PCI são armazenados na entrada de tabela de dispositivo, ETAPA 312. Além disso, o formato da tabela de tradução de endereços de nível mais elevado é também armazenado na entrada de tabela de dispositivo (por exemplo, o campo de formato), ETAPA 314, bem como o apontador de tradução de endereços de entrada/saída (*Input/Output Address Translation* – IOAT) utilizado para apontar para a tabela de tradução de endereços de nível mais elevado, ETAPA 316. Isto conclui o processo de registo.

Responsivo para realizar o registo, um espaço de endereçamento DMA e as tabelas de tradução de endereços correspondentes estão prontos para utilização, assim como uma entrada de tabela de dispositivo. Os detalhes sobre o processamento de um pedido emitido por um solicitante, tal como um adaptador, para aceder à memória de sistema são descritos com referência à FIG. 4. O processamento descrito abaixo é realizado pelo *hub* de I/O. Num exemplo, é a unidade de tradução de endereços e de proteção que realiza a lógica. Numa forma de realização, inicialmente, um pedido de DMA é recebido no *hub* de entrada/saída, ETAPA 400. Por exemplo, uma função PCI emite um pedido que é encaminhado para o *hub* PCI via, por exemplo, um *switch* PCI. Utilizando o ID solicitante no pedido, a entrada de tabela de dispositivo apropriada é localizada, ETAPA 402. Depois disso, é feita uma determinação sobre se a entrada de tabela de dispositivo é válida, PESQUISA 404. Num exemplo, a validade é determinada pela verificação de um *bit* de validade na própria entrada. Este *bit* é definido, por exemplo, em resposta à execução de um pedido de função de ativação pelo sistema operativo. Se ativado, o *bit* é definido como, por exemplo, um (ou seja, válido); caso contrário, ele permanece como zero (ou seja, inválido). Num outro exemplo, o *bit* pode ser definido quando o processo de registo está completo. Se a entrada na tabela de dispositivo é inválida, é apresentado um erro, ETAPA 405. Caso contrário, uma determinação adicional é feita sobre se o endereço PCI fornecido no pedido é menor do que o endereço base PCI armazenado na entrada de tabela de

dispositivo, PESQUISA 406. Se for, então o endereço está fora de um intervalo válido e é fornecido um erro, ETAPA 407. No entanto, se o endereço PCI é maior ou igual do que o endereço base, então uma é feita outra determinação sobre se o endereço PCI é maior do que o valor limite PCI na entrada de tabela de dispositivo, PESQUISA 408. Se o endereço PCI é maior do que o limite, então mais uma vez, é apresentado um erro uma vez que o endereço está fora do intervalo válido, ETAPA 409. No entanto, se o endereço está dentro de um intervalo válido, então o processamento continua.

Num exemplo, o formato fornecido na entrada da tabela do dispositivo é utilizado para determinar os *bits* de endereço PCI no endereço a ser utilizado para a tradução de endereços, ETAPA 410. Por exemplo, se o formato indica que a tabela de nível superior é uma tabela de primeiro nível com páginas de 4k, então os *bits* 29:21 do endereço são utilizados para indexar na tabela de primeiro nível; os *bits* 20:12 são utilizados para indexar na tabela de página; e os *bits* 11:0 são utilizados para indexar na página de 4k. Os *bits* utilizados dependem de quantos *bits* são necessários para indexar na página ou tabela de tamanho determinado. Por exemplo, para uma página de 4k com endereçamento de nível de *byte*, 12 *bits* são utilizados para endereçar 4096 *bytes*; e para uma tabela de página de 4k com 512 entradas, de 8 *bytes* cada, 9 *bits* são utilizados para endereçar 512 entradas, etc.

Em seguida, o *hub* PCI pesquisa a entrada da tabela de tradução de endereços apropriada, ETAPA 412. Por exemplo, inicialmente, a tabela de tradução de nível mais elevado é localizada utilizando o apontador IOAT da entrada de tabela de dispositivo. Então, os *bits* do endereço (aqueles após os *bits* de ordem superior utilizados para validade e não tradução; por exemplo, os *bits* 29:21 no exemplo acima) são utilizados para localizar a entrada específica dentro dessa tabela.

Uma determinação é então feita com base, por exemplo, no formato fornecido na entrada de tabela de dispositivo, sobre se a entrada de tradução de endereços localizada tem um formato correto, PESQUISA 414. Por exemplo, o formato na entrada de tabela de dispositivo é comparado com um formato indicado na entrada de tradução de endereços. Se for igual, então o formato na entrada de tabela de dispositivo é válido. Se não, é fornecido um erro, ETAPA 415; caso contrário, o processamento continua com a determinação sobre se esta é a última tabela a ser processada, PESQUISA 416. Ou seja, uma determinação é feita sobre se existem outras tabelas de tradução de endereços necessárias para obter o endereço real ou absoluto, ou se a entrada da tabela de nível mais baixo foi localizada. Esta determinação é feita com base no formato e tamanho fornecidos das tabelas que já foram processadas. Se não for a última tabela, então o processamento continua com a ETAPA 412. Caso contrário, o *hub* de I/O continua o processamento para permitir uma pesquisa ou armazenamento dos dados no

endereço traduzido, ETAPA 418. Num exemplo, o *hub* de I/O encaminha o endereço traduzido para o controlador de memória, o qual utiliza o endereço para pesquisar ou armazenar dados na localização DMA designada pelo endereço traduzido.

Conforme descrito acima, de acordo com um aspeto da presente invenção, o número de níveis de tradução e, portanto, o número de pesquisas necessárias para realizar a tradução são reduzidos. Isto é conseguido, por exemplo, ignorando os *bits* de ordem superior de um endereço durante a tradução e utilizando apenas os *bits* de ordem inferior para atravessar as tabelas de tradução, que se baseiam, por exemplo, no tamanho do espaço de endereçamento DMA atribuído ao adaptador. A utilização de um endereço parcial *versus* o endereço completo é mostrada adicionalmente nos exemplos seguintes.

Com referência inicialmente à FIG. 5A, é representado um exemplo em que o endereço inteiro é utilizado na tradução de endereços/acesso à memória. Com esta técnica anterior, são necessários seis níveis de tabelas de tradução, incluindo a tabela de página. O início da tabela de nível mais elevado (por exemplo, a tabela de nível 5, neste exemplo) é apontado por um apontador IOAT, e então os *bits* do endereço PCI são utilizados para localizar uma entrada na tabela. Cada entrada da tabela de tradução aponta para o início de uma tabela de tradução de nível inferior ou para uma página (por exemplo, uma entrada na

tabela de nível 5 aponta para o início de uma tabela de nível 4, etc.).

Neste exemplo, o espaço de endereçamento DMA (DMAAS) tem 6M em tamanho, e cada tabela tem 4k *bytes* com um máximo de 512 entradas de 8 *bytes* (com exceção da tabela de nível 5, que tem 128 entradas baseadas no tamanho do endereço). O endereço tem, por exemplo, 64 *bits*: FFFF C000 0009 C600. O início da tabela de nível 5 é apontado pelo apontador IOAT e os *bits* 63:57 do endereço PCI são utilizados para indexar na tabela de nível 5 para localizar o início da tabela de nível 4; os *bits* 56:48 do endereço PCI são utilizados para indexar na tabela de nível 4 para localizar o início da tabela de nível 3; os *bits* 47:39 são utilizados para indexar na tabela de nível 3 para localizar o início da tabela de nível 2; os *bits* 38:30 são utilizados para indexar na tabela de nível 2 para localizar o início da tabela de nível 1; os *bits* 29:21 são utilizados para indexar na tabela de nível 1 para localizar o início da tabela de página; os *bits* 20:12 são utilizados para indexar na tabela de página para localizar o início da página; e os *bits* 11:0 são utilizados para localizar a entrada na página de 4k. Assim, neste exemplo, todos os *bits* de endereço são utilizados para tradução/acesso.

Isto está em contraste com o exemplo na FIG. 5B, na qual o espaço de endereçamento é do mesmo tamanho (por exemplo 6M) e o endereço é a mesmo, mas a técnica de tradução ignora alguns dos *bits* de endereço durante a

tradução. Neste exemplo, os *bits* 63:30 do endereço são ignorados para tradução. O apontador IOAT aponta para o início da tabela de nível 1 e os *bits* 29:21 do endereço PCI são utilizados para indexar na tabela de nível 1 para localizar o início da tabela de página; os *bits* 20:12 são utilizados para indexar na tabela de página apropriada para localizar o início da página; e os *bits* 11:0 são utilizados para indexar na página de 4k.

Tal como mostrado, a tabela de nível 1 500 inclui três entradas 502, cada uma fornecendo um endereço a uma das três tabelas de página 504. O número de tabelas de página necessárias, e portanto, o número de tabelas de outros níveis, depende, por exemplo, do tamanho do espaço de endereçamento DMA, do tamanho das tabelas de tradução, e/ou do tamanho das páginas. Neste exemplo, o espaço de endereçamento DMA tem 6M, e cada tabela de página tem 4k, tendo até 512 entradas. Portanto, cada tabela de página pode mapear até 2M de memória (4k x 512 entradas). Assim, três tabelas de página são necessárias para o espaço de endereçamento de 6M. A tabela de nível 1 é capaz de armazenar as três entradas, uma para cada tabela de página, e assim, não são necessários mais níveis de tabelas de tradução de endereços, neste exemplo.

Numa outra forma de realização, podem existir diferentes formatos de tabelas de tradução de endereços utilizados para tradução de endereços, e podem existir variações entre os formatos. Assim, existem variações nos

*bits* utilizados para indexar numa determinada tabela ou página. Alguns destes exemplos são descritos com referência às FIGs. 5C e 5D.

Por exemplo, um formato é um formato compatível com CPU DAT no qual as tabelas de tradução são compatíveis com as tabelas de tradução utilizadas para traduções CPU DAT. Podem existir vários formatos compatíveis com CPU DAT, exemplos dos quais são descritos com referência à FIG. 5C. Tal como mostrado, um formato compatível com CPU DAT é um formato compatível com CPU DAT com uma página de 4k 550, e outro é um formato compatível com CPU DAT com uma página de 1M 552, como exemplos. O número de *bits* mostrados são o número de *bits* de endereço utilizados para indexar nessa página ou tabela (ou de outro modo localizar uma entrada na referida página ou tabela). Por exemplo, 12 *bits* 554 de um endereço PCI são utilizados como um deslocamento de *byte* numa página de 4k 556, 8 *bits* 558 são utilizados como um índice para uma tabela de página 560, 11 *bits* 562 são utilizados como um índice para uma tabela de segmento 564, etc. Localizado sob a tabela de tradução de endereços designada está o tamanho máximo do espaço de endereçamento suportado por essa tabela de tradução de endereços. Por exemplo, a tabela de página 560 suporta um espaço de endereçamento DMA de 1M; a tabela de segmento 564 suporta um espaço de endereçamento DMA de 2G, etc. Nesta figura, bem como na FIG. 5D, K = *kilobytes*, M = *megabytes*, G = *gigabytes*, T = *terabytes*, P = *petabytes* e E = *exabytes*.

Tal como representado, à medida que o tamanho da página aumenta, o número de níveis das tabelas de tradução diminui.Por exemplo, para a página de 4k 556, é necessária uma tabela de página, mas não é necessária para a página de 1M. Outros exemplos e variações são possíveis.

Outro formato de tradução de endereços é um formato de tradução de endereços estendido de I/O em que as tabelas de tradução de endereços estendidas são utilizadas. Vários exemplos de formatos de tradução de endereços estendidos de I/O estão representados na FIG. 5D. Por exemplo, os seguintes formatos são mostrados: uma tabela de tradução de endereços de 4k com páginas de 4k 570; tabelas de tradução de endereços de 1M com páginas de 4k 572; e tabelas de tradução de endereços de 1M com páginas de 1M 574. Tal como acontece com os formatos compatíveis com CPU DAT, o número de *bits* listados são os *bits* utilizados para localizar uma entrada na tabela específica. Por exemplo, no número de referência 576, os 12 *bits* são um deslocamento na página de 4k. Do mesmo modo, no número de referência 578, os 9 *bits* são utilizados para indexar numa tabela de página de I/O. Esta tabela de página de I/O permite um espaço de endereçamento DMA que tenha um tamanho de 2M. Existem muitos outros exemplos.

Numa implementação específica, para realizar o registo de um espaço de endereçamento DMA para o adaptador, uma instrução referida como uma instrução Modificar Controlos de Função PCI (*Modify PCI Function Controls* –

MPFC), é utilizada. Por exemplo, o sistema operativo determina qual o formato de tradução de endereços que pretende utilizar, constrói as tabelas de tradução de endereços para esse formato, e em seguida emite a instrução MPFC com esse formato incluído como um operando da instrução. Num exemplo, o formato e outros operandos da instrução são incluídos num bloco de informação de função (descrito abaixo), que é um operando da instrução. O bloco de informação de função é depois utilizado para atualizar a DTE e, numa forma de realização, opcionalmente, uma entrada de tabela de função (*Function Table Entry* - FTE) que inclui parâmetros operacionais do adaptador.

Uma forma de realização dos detalhes relacionados com esta instrução, e em particular o processo de registo, é descrita com referência às FIGs. 6A-9. Com referência à FIG. 6A, uma instrução Modificar Controlos de Função PCI 600 inclui, por exemplo, um código de operação 602 indicando a instrução Modificar Controlos de Função PCI; um primeiro campo 604 especificando uma localização onde várias informações estão incluídas em relação à função de adaptador para a qual os parâmetros operacionais estão a ser estabelecidos; e um segundo campo 606 especificando uma localização a partir da qual um bloco de informação de função PCI (Function Information Block - FIB) é pesquisado. Os conteúdos das localizações designadas por Campos 1 e 2 encontram-se adicionalmente descritos abaixo.

Numa forma de realização, o Campo 1 designa um

registo geral que inclui uma variedade de informação. Tal como mostrado na FIG. 6B, os conteúdos do registo incluem, por exemplo, um manípulo (*handle*) de função 610 que identifica a *handle* da função de adaptador em nome da qual a instrução de modificar está a ser realizada; um espaço de endereçamento 612 designando um espaço de endereçamento na memória de sistema associado com a função de adaptador designada pela *handle* da função; um controlo de operação 614 que especifica a operação a ser realizada para a função de adaptador; e estado 616 que fornece o estado relacionado com a instrução quando a instrução termina com um código predefinido.

Numa forma de realização, a *handle* da função inclui, por exemplo, um indicador de ativação indicando se a *handle* foi ativada, um número de função que identifica uma função de adaptador (este é um identificador estático e pode ser utilizado para indexar numa tabela de função); e um número de instância especificando a instância específica desta *handle* de função. Existe uma *handle* de função para cada função de adaptador, e é utilizada para localizar uma entrada de tabela de função (*Function Table Entry* - FTE) dentro da tabela de função. Cada entrada de tabela de função inclui parâmetros operacionais e/ou outras informações associadas com a sua função de adaptador. Como exemplo, uma entrada de tabela de função inclui:

Número de Instância: Este campo indica uma instância específica da *handle* de função de adaptador

associada com a entrada de tabela de função;

Índice 1...n da Entrada de Tabela de Dispositivo (*Device Table Entry* – DTE): Podem existir um ou mais índices de tabela de dispositivo, e cada índice é um índice para uma tabela de dispositivo para localizar uma entrada de tabela de dispositivo (DTE). Existem uma ou mais entradas de tabela de dispositivo por cada função de adaptador, e cada entrada inclui informação associada à sua função de adaptador, incluindo informação utilizada para processar os pedidos da função de adaptador (por exemplo, pedidos de DMA, pedidos de MSI) e informação referente a pedidos relacionados com a função de adaptador (por exemplo, instruções PCI). Cada entrada de tabela de dispositivo está associada com um espaço de endereçamento na memória de sistema atribuída à função de adaptador. Uma função de adaptador pode ter um ou mais espaços de endereçamento dentro da memória de sistema atribuída à função de adaptador.

Indicador de Ocupado (*Busy*): Este campo indica se a função de adaptador está ocupada;

Indicador de Estado de Erro Permanente: Este campo indica se a função de adaptador está em num estado de erro permanente;

Indicador de Recuperação Iniciada: Este campo indica se a recuperação foi iniciada para a função de

adaptador;

Indicador de Permissão: Este campo indica se o sistema operativo que tenta controlar a função de adaptador tem autorização para fazê-lo;

Indicador de Ativação: Este campo indica se a função de adaptador está ativada (por exemplo, 1 = ativada, 0 = desativada);

Identificador do Solicitante (Requestor Identifier - RID): Este é um identificador da função de adaptador, e inclui, por exemplo, um número de *bus*, um número de dispositivo e um número de função.

Num exemplo, este campo é utilizado para acessos de um espaço de configuração da função de adaptador. (Memória de um adaptador pode ser definida como espaços de endereçamento, incluindo, por exemplo, um espaço de configuração, um espaço de I/O e/ou um ou mais espaços de memória.). Num exemplo, o espaço de configuração pode ser acedido através da especificação do espaço de configuração numa instrução emitida pelo sistema operativo (ou outra configuração) para a função de adaptador. Especificado na instrução está um deslocamento no espaço de configuração e uma *handle* de função utilizada para localizar a entrada de tabela de função apropriada que inclui o RID. O *firmware* recebe a instrução e determina se é para um espaço de configuração. Portanto, utiliza o RID para gerar um pedido

para o *hub* de I/O, e o *hub* de I/O cria uma pedido para aceder ao adaptador. A localização da função de adaptador é baseada no RID, e o deslocamento especifica um deslocamento no espaço de configuração da função de adaptador.

Tal como aqui utilizado, o *firmware* inclui, por exemplo, o microcódigo, milicódigo e/ou macrocódigo do processador. Inclui, por exemplo, as instruções ao nível do *hardware* e/ou as estruturas de dados utilizadas na implementação de código de máquina de nível mais elevado. Numa forma de realização, inclui, por exemplo, código proprietário que normalmente é fornecido como microcódigo que inclui *software* de confiança ou microcódigo específico do *hardware* subjacente e controla o acesso do sistema operativo ao *hardware* do sistema.

Registo de Endereço Base (*Base Address Register* – BAR) (1 para n): Este campo inclui uma pluralidade de números inteiros sem sinal, designados como $BAR_0$ – $BAR_n$, que estão associados com a função de adaptador especificada originalmente, e cujos valores são também armazenados nos registos de endereço base associados com a função de adaptador. Cada BAR especifica o endereço inicial de um espaço de memória ou espaço de I/O dentro da função de adaptador, e indica também o tipo de espaço de endereçamento, ou seja, se se trata de um espaço de memória de 64 ou 32 *bits*, ou de um espaço de I/O de 32 *bits*, como exemplos;

Num exemplo, é utilizado para acessos ao espaço de memória e/ou ao espaço de I/O da função de adaptador. Por exemplo, um deslocamento fornecido numa instrução para aceder à função de adaptador é adicionado ao valor no registo de endereço base associado ao espaço de endereçamento designado na instrução para obter o endereço a ser utilizado para aceder à função de adaptador. O identificador de espaço de endereçamento fornecido na instrução identifica o espaço de endereçamento dentro da função de adaptador a ser acedido e o BAR correspondente a ser utilizado;

Tamanho 1...n: Este campo inclui uma pluralidade de números inteiros sem sinal, designados como $SIZE_0$ - $SIZE_n$. O valor de um campo SIZE (tamanho), quando diferente de zero, representa o tamanho de cada espaço de endereçamento com cada entrada correspondendo a um BAR previamente descrito.

Detalhes adicionais relacionados com BAR e SIZE estão descritos abaixo.

1. Quando um BAR não é implementado para uma função de adaptador, o campo BAR e o seu campo de tamanho correspondente são ambos armazenados como zeros.

2. Quando um campo BAR representa ou um espaço de endereçamento de I/O ou um espaço de endereçamento de memória de 32 *bits*, o campo de tamanho correspondente é

diferente de zero e representa o tamanho do espaço de endereçamento.

3. Quando um campo BAR representa um espaço de endereçamento de memória de 64 *bits*,

a. o campo $BAR_n$ representa os *bits* menos significativos do endereço.

b. O próximo campo $BAR_{n+1}$ consecutivo representa os *bits* mais significativos do endereço.

c. O campo $SIZE_n$ correspondente é diferente de zero e representa o tamanho do espaço de endereçamento.

d. O campo $SIZE_{n+1}$ correspondente não é significativo e é armazenado como zero.

Informação de Encaminhamento Interno: Esta informação é utilizada para realizar encaminhamento específico ao adaptador. Inclui, por exemplo, nó, *chip* do processador, e informação de endereçamento do *hub*, como exemplos.

Indicação de Estado: Proporciona uma indicação de, por exemplo, se as operações de carga/armazenamento estão bloqueadas ou se o adaptador está no estado de erro, bem como outras indicações.

Num exemplo, o indicador de ocupado, o indicador

de estado de erro permanente, e o indicador de recuperação iniciada são definidos com base em monitorização realizada pelo *firmware*. Além disso, o indicador de permissão é definido, por exemplo, com base na política; e a informação de BAR é baseada em informação de configuração descoberta durante um varrimento de *bus* pelo processador (por exemplo, *firmware* do processador). Outros campos podem ser definidos com base na configuração, inicialização e/ou eventos. Noutras formas de realização, a entrada de tabela de função pode incluir mais, menos ou diferente informação. A informação incluída pode depender das operações suportadas ou ativadas para a função de adaptador.

Com referência à FIG. 6C, num exemplo, o Campo 2 designa um endereço lógico 620 de um bloco de informação de função (*Function Information Block* – FIB) PCI, que inclui informação sobre a função de adaptador associada. O bloco de informação de função é utilizado para atualizar uma entrada de tabela de dispositivo e/ou entrada de tabela de função (ou outra localização) associada com a função de adaptador. A informação é armazenada no FIB durante a inicialização e/ou configuração do adaptador, e/ou responsiva a eventos específicos.

Detalhes adicionais sobre um bloco de informação de função (*Function Information Block* – FIB) são descritos com referência à FIG. 6D. Numa forma de realização, um bloco de informação de função 650 inclui os seguintes campos:

Formato 651: Este campo especifica o formato do FIB.

Controlo de Interceção 652: Este campo é utilizado para indicar se a execução *guest* de instruções específicas por um *guest* de modo paginável resulta em interceção da instrução;

Indicação de Erro 654: Este campo inclui a indicação de estado de erro para interrupções de acesso direto à memória e de adaptador. Quando o *bit* é definido (por exemplo, 1), um ou mais erros foram detetados durante a realização de interrupção de acesso direto à memória ou de adaptador para a função de adaptador;

Bloqueio de Carga/Armazenamento 656: Este campo indica se as operações de carga/armazenamento estão bloqueadas;

Função PCI Válida 658: Este campo inclui um controlo de ativação para a função de adaptador. Quando o *bit* está definido (por exemplo, 1), a função de adaptador é considerada ativada para operações de I/O;

Espaço de Endereçamento Registado 660: Este campo inclui um controlo de ativação de acesso direto à memória para uma função de adaptador. Quando o campo está definido (por exemplo, 1) o acesso direto à memória está ativado;

Tamanho de Página 661: Este campo indica o tamanho da página ou outra unidade de memória a ser acedida por um acesso à memória DMA;

Endereço Base PCI (*PCI Base Address* – PBA) 662: Este campo é um endereço base para um espaço de endereçamento na memória de sistema atribuído para a função de adaptador. Representa o menor endereço virtual que é permitido a uma função de adaptador utilizar para acesso direto à memória ao espaço de endereçamento DMA especificado;

Limite de Endereço PCI (*PCI Address Limit* – PAL) 664: Este campo representa o maior endereço virtual a que é permitido aceder uma função de adaptador dentro do espaço de endereçamento DMA especificado;

Apontador de Tradução de Endereço de Entrada/Saída (*Input/Output Address Translation Pointer* – IOAT) 666: O apontador de tradução de endereço de entrada/saída designa a primeira de quaisquer tabelas de tradução utilizadas por uma tradução de endereço virtual PCI, ou pode designar diretamente o endereço absoluto de uma estrutura de armazenamento que é o resultado da tradução;

Subclasse de Interrupção (*Interrupt Subclass* – ISC) 668: Este campo inclui a subclasse de interrupção utilizada para apresentar interrupções de adaptador para a

função de adaptador;

Número de Interrupções (*Number of Interruptions* – NOI) 670: Este campo indica o número de códigos de interrupção distintos aceites para uma função de adaptador. Este campo também define o tamanho, em *bits*, do vetor de *bit* de interrupção do adaptador designado por um endereço de vetor de *bit* de interrupção do adaptador e pelos campos de deslocamento do vetor de *bit* de interrupção do adaptador;

Endereço de Vetor de *Bit* de Interrupção do Adaptador (*Adapter Interruption Bit Vector Address* – AIBV) 672: Este campo especifica um endereço do vetor de *bit* de interrupção do adaptador para a função de adaptador. Este vetor é utilizado no processamento de interrupção;

Deslocamento de Vetor de *Bit* de Interrupção do Adaptador 674: Este campo especifica o deslocamento do primeiro *bit* do vetor de *bit* de interrupção do adaptador para a função de adaptador;

Endereço de *Bit* de Resumo de Interrupção do Adaptador (*Adapter Interruption Summary Bit Address* – AISB) 676: Este campo fornece um endereço que designa o *bit* de sumarização de interrupção do adaptador, que é opcionalmente utilizado no processamento de interrupção;

Deslocamento de *Bit* de Sumarização de Interrupção

do Adaptador 678: Este campo fornece o deslocamento no vetor de *bit* de resumo de interrupção do adaptador;

Endereço de Bloco de Medição de Função (Function *Measurement Block* – FMB) 680: Este campo fornece um endereço de um bloco de medição de função utilizado para recolher medições relacionadas com a função de adaptador;

Chave de Bloco de Medição de Função 682: Este campo inclui uma chave de acesso para aceder ao bloco de medição de função;

Controlo de Notificação de *Bit* de Resumo 684: Este campo indica se há um vetor de *bit* de resumo a ser utilizado;

*Token* de Autorização de Instrução 686: Este campo é utilizado para determinar se um modo de *guest* de armazenamento paginável está autorizado a executar instruções PCI sem intervenção do *host*.

Num exemplo, na *z/Architecture*®, um *guest* paginável é interpretativamente executado através da execução da instrução Iniciar Execução Interpretativa (*Start Interpretive Execution* – SIE), no nível 2 de interpretação. Por exemplo, o hipervisor de partição lógica (*Logical Partition Hypervisor* – LPAR) executa a instrução SIE para iniciar a partição lógica na memória física, fixa. Se *z/VM*® é o sistema operativo nessa partição lógica, emite

a instrução SIE para executar as suas máquinas clientes (virtuais) no seu armazenamento V = V (virtual). Portanto, o hipervisor LPAR utiliza SIE de nível 1, e o hipervisor *z/VM®* utiliza SIE de nível 2 SIE; e

Formato de Tradução de Endereço 687: Este campo indica um formato selecionado para tradução de endereço da tabela de tradução de nível mais elevado a ser utilizado na tradução (por exemplo, uma indicação de tabela de nível mais elevado (por exemplo, tabela de segmento, região terceira, etc.)).

O bloco de informação de função designado na instrução Modificar Controlos de Função PCI é utilizado para modificar uma entrada de tabela de dispositivo selecionada, uma entrada de tabela de função e/ou outros controlos de *firmware* associados com a função de adaptador designada na instrução. Ao modificar a entrada de tabela de dispositivo, a entrada de tabela de função e/ou outros controlos de *firmware*, determinados serviços são fornecidos ao adaptador. Estes serviços incluem, por exemplo, interrupções de adaptador; traduções de endereços; reinicialização de estado de erro; reinicialização de bloqueio de carga/armazenamento; definição de parâmetros de medição de função; e definição de controlo de interceção.

Uma forma de realização da lógica associada com a instrução Modificar Controlos de Função PCI é descrita com referência à FIG. 7. Num exemplo, a instrução é emitida por

um sistema operativo (ou outra configuração) e executada pelo processador (por exemplo, *firmware*) que executa o sistema operativo. Nos exemplos aqui contidos, a instrução e funções de adaptador são baseadas em PCI. Contudo, noutros exemplos, pode ser utilizada uma arquitetura de adaptador e instruções correspondentes diferente.

Num exemplo, o sistema operativo fornece os seguintes operandos à instrução (por exemplo, em um ou mais registos designados pela instrução): a *handle* da função PCI, o identificador de espaço de endereçamento DMA, um controlo de operação; e um endereço do bloco de informação de função.

Com referência à FIG. 7, numa primeira fase, é feita uma determinação sobre se a componente permitindo uma instrução Modificar Controlos de Função PCI está instalada, PESQUISA 700. Esta determinação é feita, por exemplo, através da verificação de um indicador armazenado, por exemplo, num bloco de controlo. Se a componente não estiver instalada, é fornecida uma condição de exceção, ETAPA 702. Caso contrário, é feita uma determinação sobre se a instrução foi emitida por um modo *guest* de armazenamento paginável (ou por outro *guest*), PESQUISA 704. Se sim, o sistema operativo *host* irá emular a operação para esse *guest*, ETAPA 706.

Caso contrário, é feita uma determinação sobre se um ou mais dos operandos estão alinhados, PESQUISA 708. Por

exemplo, é feita uma determinação sobre se o endereço do bloco de informação de função está num limite de palavra dupla. Num exemplo, isto é opcional. Se os operandos não estão alinhados, então é fornecida uma condição de exceção, ETAPA 710. Caso contrário, é feita uma determinação sobre se o bloco de informação de função está acessível, PESQUISA 712. Se não, então é fornecida uma condição de exceção, ETAPA 714. Caso contrário, é feita uma determinação sobre se a *handle* fornecida nos operandos da instrução Modificar Controlos de Função PCI está ativada, PESQUISA 716. Num exemplo, esta determinação é feita através da verificação de um indicador de ativação na *handle*. Se a *handle* não estiver ativada, então é fornecida uma condição de exceção, ETAPA 718.

Se a *handle* estiver ativada, a *handle* é utilizada para localizar uma entrada de tabela de função, ETAPA 720. Isto é, pelo menos uma porção da *handle* é utilizada como um índice para a tabela de função para localizar a entrada de tabela de função correspondente à função de adaptador para a qual os parâmetros operacionais devem ser estabelecidos.

É feita uma determinação sobre se a entrada de tabela de função foi encontrada, PESQUISA 722. Se não, então é fornecida uma condição de exceção, ETAPA 724. Caso contrário, se a configuração que emite a instrução é um *guest*, PESQUISA 726, então é fornecida uma condição de exceção (por exemplo, a interceção do *host*), ETAPA 728. Esta pesquisa pode ser ignorada se a configuração não é um

*guest* ou outras autorizações podem ser verificadas, se designado.

É então feita uma determinação sobre se a função está ativada, PESQUISA 730. Num exemplo, essa determinação é feita através da verificação de um indicador de ativação na entrada de tabela da função. Se não estiver ativada, é fornecida uma condição de exceção, ETAPA 732.

Se a função estiver ativada, então é feita uma determinação sobre se a recuperação está ativa, PESQUISA 734. Se a recuperação estiver ativa conforme determinado por um indicador de recuperação na entrada de tabela de função, então é fornecida uma condição de exceção, ETAPA 736. No entanto, se a recuperação não estiver ativa, é feita uma nova determinação sobre se a função está ocupada, PESQUISA 738. Esta determinação é feita através da verificação do indicador de ocupado na entrada de tabela de função. Se a função estiver ocupada, então é fornecida uma condição de ocupado, ETAPA 740. Com a condição de ocupado, a instrução pode ser repetida, em vez de descartada.

Se a função não estiver ocupada, então é feita uma nova determinação sobre se o formato de bloco de informação de função é válido, PESQUISA 742. Por exemplo, o campo de formato do FIB é verificado para determinar se este formato é suportado pelo sistema. Se for inválido, então é fornecida uma condição de exceção, ETAPA 744. Se o formato de bloco de informação de função for válido, então

é feita uma nova determinação sobre se o controlo de operação especificado nos operandos da instrução é válido, PESQUISA 746. Ou seja, é o controlo da operação um dos controlos de operação especificados para esta instrução. Se for inválido, então é fornecida uma condição de exceção, ETAPA 748. No entanto, se o controlo de operação é válido, então o processamento continua com o controlo de operação específico sendo especificado.

Um controlo de operação que pode ser especificado é uma operação de registo de parâmetros de tradução de endereços de I/O utilizada no controlo de traduções de endereços para um adaptador. Com esta operação, os parâmetros de função PCI relevantes para a tradução de endereços de I/O são definidos na DTE, FTE e/ou outra localização a partir dos parâmetros apropriados do FIB, que é um operando para a instrução. Estes parâmetros incluem, por exemplo, o endereço base PCI, o limite de endereço PCI (*aka*, limite PCI ou limite); o formato de tradução de endereços; o tamanho da página; e o apontador de tradução de endereços de I/O, que são operandos para esta operação. Existem também operandos implícitos, incluindo um endereço DMA inicial (*Starting DMA Address* – SDMA) e um endereço DMA final (*Ending DMA Address* – EDMA), que estão armazenados numa localização acessível ao processador que executa a instrução.

Uma forma de realização da lógica para estabelecer os parâmetros operacionais para tradução de

endereços de I/O é descrita com referência à FIG. 8. Inicialmente, é feita uma determinação sobre se o endereço base PCI no FIB é maior do que o limite PCI no FIB, PESQUISA 800. Se a comparação do endereço base e o limite indicam que o endereço base é maior do que o limite, então uma condição de exceção é reconhecida, ETAPA 802. No entanto, se o endereço base é menor ou igual ao limite, então uma nova determinação é feita sobre se o formato de tradução de endereços e o tamanho de página são válidos, PESQUISA 804. Se forem inválidos, então é fornecida uma condição de exceção, ETAPA 806. No entanto, se forem válidos, então uma nova determinação é feita sobre se o tamanho do espaço de endereçamento (com base no endereço base e no limite) excede a capacidade de tradução, PESQUISA 808. Num exemplo, o tamanho do espaço de endereçamento é comparado com a capacidade máxima possível de tradução de endereços com base no formato da tabela de nível superior. Por exemplo, se a tabela de nível superior é uma tabela de segmento compatível com DAT, a capacidade máxima de tradução é de 2 Gbytes.

Se o tamanho do espaço de endereçamento exceder a capacidade de tradução, então é fornecida uma condição de exceção, ETAPA 810. Caso contrário, uma nova determinação é feita sobre se o endereço base é menor do que o endereço DMA inicial, PESQUISA 812. Se assim for, então é fornecida uma condição de exceção, ETAPA 814. Caso contrário, uma outra determinação é feita sobre se o limite de endereços é superior ao último endereço DMA, PESQUISA 816. Se o for,

então é fornecida uma condição de exceção, ETAPA 818. Num exemplo, o endereço DMA inicial e o endereço DMA final são baseados numa política global do sistema.

A partir daí, é feita uma determinação sobre se estão disponíveis recursos suficientes, se forem necessários, para realizar uma tradução de endereços de I/O, PESQUISA 820. Se não, então é fornecida uma condição de exceção, ETAPA 822. Caso contrário, uma nova determinação é feita sobre se os parâmetros de tradução de endereços de I/O já foram registados na FTE e DTE, PESQUISA 824. Isto é determinado através da verificação dos valores dos parâmetros na FTE/DTE. Por exemplo, se os valores da FTE/DTE são zero ou outro valor definido, então o registo não foi realizado. Para localizar a FTE, o identificador fornecido na instrução é utilizado e, para localizar a DTE, um índice dispositivo na FTE é utilizada.

Se a função de adaptador já foi registada para a tradução de endereços, então é fornecida uma condição de exceção, ETAPA 826. Se não, então é feita uma determinação sobre se o espaço de endereçamento DMA que foi especificado é válido (ou seja, se é um espaço de endereçamento para o qual uma DTE foi ativada), PESQUISA 828. Se não, então é fornecida uma condição de exceção, ETAPA 830. Se todas as verificações forem bem sucedidas, os parâmetros de tradução são colocados na entrada de tabela de dispositivo, e opcionalmente na entrada de tabela de função correspondente (ou outra localização designada), ETAPA 832. Por exemplo,

os parâmetros de função PCI relevantes para a tradução de endereços de I/O são copiados a partir do bloco de informação de função e colocados na DTE/FTE. Estes parâmetros incluem, por exemplo, o endereço base PCI, o limite de endereço PCI, o formato de tradução, o tamanho da página, e o apontador de tradução de endereços de I/O. Esta operação permite acessos DMA ao espaço de endereçamento DMA especificado. Permite tradução de endereços de I/O para a função de adaptador.

Outro controlo de operação que pode ser especificado pela instrução Modificar Controlos de Função PCI é uma operação de cancelar registo de parâmetros de tradução de endereços de I/O, a exemplo do que é descrito com referência à FIG. 9. Com esta operação, os parâmetros de função relevantes para tradução de endereços de I/O são reinicializados para zeros. Esta operação desativa os acessos DMA ao espaço de endereçamento DMA especificado e provoca uma limpeza das entradas de *buffer* de tradução (*translation lookaside buffer* – TLB) de I/O para esse espaço de endereçamento DMA. Desativa a tradução de endereços.

Com referência à FIG. 9, numa forma de realização, é feita uma determinação sobre se os parâmetros de tradução de endereços de I/O não são registados, PESQUISA 900. Num exemplo, esta determinação é feita através da verificação dos valores dos parâmetros apropriados na FTE ou DTE. Se esses campos são zero ou

algum valor especificado, não são registados. Portanto, é fornecida uma condição de exceção, ETAPA 902. Se eles são registados, então é feita uma determinação sobre se o espaço de endereçamento DMA é válido, PESQUISA 904. Se for inválido, então é fornecida uma condição de exceção, ETAPA 906. Se o espaço de endereçamento DMA é válido, então os parâmetros de tradução na entrada de tabela de dispositivo, e opcionalmente na entrada de tabela de função correspondente são apagados, ETAPA 908.

Descrito em detalhe acima está um mecanismo eficiente para traduzir um endereço de entrada/saída fornecido por um adaptador para um endereço de memória de sistema. Num exemplo, o endereço PCI completo (por exemplo, a totalidade do endereço de 64 *bits*) é utilizado no acesso à memória, no entanto, as pesquisas de tradução de endereços são minimizadas utilizando apenas uma porção do endereço para tradução. Embora o uso do endereço completo forneça proteção adicional, a utilização de, por exemplo, apenas os *bits* de ordem mais baixa para tradução permite uma tradução mais eficiente por meio de menos níveis de pesquisa de tradução. Isto permite uma maior flexibilidade do sistema operativo na utilização de endereços que podem coexistir ou serem os mesmos que os endereços virtuais do sistema operativo. Além disso, permite que as tabelas de tradução de endereços de CPU possam ser partilhadas por adaptadores, permitindo ao mesmo tempo que o I/O reduza o número de tabelas que necessitam de ser digitalizadas. Além disso, é fornecida proteção adicional ao ser permitido que

diferentes adaptadores e/ou sistemas operativos utilizem intervalos de espaço de endereçamento disjuntos. Nas formas de realização aqui descritas, os adaptadores são adaptadores PCI. PCI, tal como é aqui utilizado, refere-se a todos os adaptadores implementados de acordo com uma especificação baseada em PCI, tal como definido pelo Grupo de Interesse Especial em Conexão Componente Periférica (*Peripheral Component Interconnect Special Interest Group* – PCI-SIG), incluindo, mas não estando limitado a, PCI ou PCIe. Num exemplo particular, a Conexão Componente Periférica Expresso (*Peripheral Component Interconnect Express* – PCIe) é uma norma de interconexão do nível componente que define um protocolo de comunicação bidirecional para transações entre adaptadores e sistemas de *host* de I/O. As comunicações PCIe são encapsuladas em pacotes de acordo com a norma PCIe para a transmissão num *bus* PCIe. As transações provenientes de adaptadores de I/O e terminando em sistemas *host* são referidas como transações *upbound*. As transações provenientes de sistemas *host* e terminando em adaptadores de I/O são chamadas de transações *downbound*. A topologia PCIe baseia-se em ligações unidirecionais ponto-a-ponto que são emparelhadas (por exemplo, uma ligação *upbound*, uma ligação *downbound*) para formar o *bus* PCIe. A norma PCIe é mantida e publicada pelo PCI-SIG.

Como será valorizado por um perito na técnica, os aspetos da presente invenção podem ser incorporados como um sistema, método ou produto de programa de computador.

Assim, aspetos da presente invenção podem assumir o contorno de uma forma de realização totalmente de *hardware*, uma forma de realização totalmente de *software* (incluindo *firmware*, *software* residente, microcódigo, etc) ou uma forma de realização combinando aspetos de *software* e de *hardware* que podem todos ser geralmente aqui referidos como um "circuito", "módulo" ou "sistema". Além disso, os aspetos da presente invenção podem assumir o contorno de um produto de programa de computador incorporado num ou mais suportes legíveis por computador, tendo código de programa legível por computador incorporado nele.

Qualquer combinação de um ou mais suportes legíveis por computador pode ser utilizada. O suporte legível por computador pode ser um suporte de armazenamento legível por computador. Um suporte de armazenamento legível por computador pode ser, por exemplo, mas não estando limitado, a um sistema eletrónico, magnético, ótico, eletromagnético, de infravermelhos ou de semicondutores, aparelho, ou dispositivo, ou qualquer combinação adequada destes materiais. Exemplos mais específicos (uma lista não exaustiva) do suporte de armazenamento legível por computador incluem o seguinte: uma conexão elétrica com um ou mais fios, uma disquete de computador portátil, um disco rígido, uma memória de acesso aleatório (*Random Access Memory* - RAM), uma memória só de leitura (*Read-Only Memory* - ROM), uma memória só de leitura programável apagável (*Erasable Programmable Read-Only Memory* - EPROM ou memória *Flash*), uma fibra ótica, uma memória só de leitura em disco

compacto portátil (*Compact Disc Read-Only Memory* – CD-ROM), um dispositivo de armazenamento ótico, um dispositivo de armazenamento magnético, ou qualquer combinação adequada destes materiais. No contexto deste documento, um suporte de armazenamento legível por computador pode ser um qualquer suporte material que possa conter ou armazenar um programa para utilização ou ligação com um sistema, aparelho ou dispositivo de execução de instruções.

Com referência agora à FIG. 10, num exemplo, um produto de programa de computador 1000 inclui, por exemplo, um ou mais suportes de armazenamento legíveis por computador 1002 para armazenar suportes ou lógica de código de programa legível por computador 1004 sobre os mesmos para proporcionar e facilitar um ou mais aspetos da presente invenção.

O código do programa incorporado num suporte legível por computador pode ser transmitido através de um suporte adequado, incluindo, mas não estando limitado a um suporte sem fios, com fios, um cabo de fibra ótica, RF, etc, ou qualquer combinação adequada destes materiais.

O código de programa de computador para a realização de operações para aspetos da presente invenção pode ser escrito em qualquer combinação de uma ou mais linguagens de programação, incluindo uma linguagem de programação orientada a objetos, como Java, Smalltalk, C++ ou semelhante, e linguagens de programação processuais

convencionais, tais como a linguagem de programação "C", *assembler* ou linguagens de programação similares. O código de programa pode executar-se inteiramente no computador do utilizador, em parte no computador do utilizador, como um pacote de *software* independente, parcialmente no computador do utilizador e parcialmente num computador remoto ou inteiramente no computador ou servidor remoto. Neste último cenário, o computador remoto pode ser conectado ao computador do utilizador através de qualquer tipo de rede, incluindo uma rede de área localização (*Local Area Network* – LAN), uma rede de área ampla (*Wide Area Network* – WAN), ou ou a conexão pode ser feita para um computador externo (para exemplo, através da Internet utilizando um Provedor de Serviços de Internet).

Os aspetos da presente invenção são aqui descritos com referência ilustrações de fluxogramas e/ou diagramas de blocos de métodos, aparelhos (sistemas) e produtos de programas de computador de acordo com formas de realização da invenção. Será compreendido que cada bloco das ilustrações de fluxogramas e/ou diagramas de blocos, e combinações de blocos nas ilustrações de fluxogramas e/ou diagramas de blocos, podem ser implementados por instruções de programa de computador. Estas instruções de programa de computador podem ser fornecidas a um processador de um computador de uso geral, um computador com um objetivo específico, ou qualquer outro aparelho de processamento de dados programável para produzir uma máquina, de tal modo que as instruções que se executam por meio do processador

do computador ou outro aparelho de processamento de dados programável, criar suportes para implementar as funções/ações especificadas no bloco ou blocos de fluxograma e/ou bloco de diagrama.

Estas instruções de programa de computador podem também ser armazenadas num suporte legível por computador que pode controlar um computador, outro aparelho de processamento de dados programável, ou outros dispositivos para funcionar de uma forma específica, de tal modo que as instruções armazenadas no suporte legível pelo computador produzam um artigo de fabrico incluindo instruções que implementem a função/ação especificada no blocos ou blocos de fluxograma e/ou bloco de diagrama.

As instruções do programa de computador podem também ser carregadas num computador, outro aparelho de processamento de dados programável, ou outros dispositivos para provocar uma série de etapas operacionais a serem executadas no computador, outro aparelho programável ou outros dispositivos para produzir um processo implementado por computador de tal modo que as instruções que se executam no computador ou outro aparelho programável proporcionam processos para implementação das funções/ações especificados no blocos ou blocos de fluxograma e/ou bloco de diagrama.

Os fluxogramas e diagramas de blocos nas figuras ilustram a arquitetura, funcionalidade e operação de

possíveis implementações de sistemas, métodos e produtos de programas de computador de acordo com várias formas de realização da presente invenção. A este respeito, cada bloco nos fluxogramas ou diagramas de blocos pode representar um módulo, segmento, ou porção de código, o qual compreende uma ou mais instruções executáveis para a implementação da(s) função(ões) lógica(s) especificada(s). Também deve ser notado que, em algumas implementações alternativas, as funções destacadas no bloco podem ocorrer fora da ordem destacada nas figuras. Por exemplo, dois blocos mostrados em sucessão podem, na verdade, ser executados substancialmente de forma concorrente, ou os blocos podem por vezes ser executados na ordem inversa, dependendo da funcionalidade envolvida. Também deve ser notado que cada bloco dos diagramas de blocos e/ou ilustração de fluxograma, e combinações de blocos nos diagramas de blocos e/ou ilustração de fluxograma, podem ser implementados por sistemas baseados em *hardware* para fins específicos que desempenham as funções ou ações especificadas, ou combinações de *hardware* e instruções de computador para fins específicos.

Para além do referido, um ou mais aspetos da presente invenção pode ser fornecido, oferecido, implementado, gerido, mantido, etc. por um provedor de serviços que proponha gestão de ambientes de cliente. Por exemplo, o provedor de serviços pode criar, manter, suportar, etc. código de computador e/ou uma infraestrutura informática que realiza um ou mais aspetos da presente

invenção para um ou mais clientes. Em comutação, o provedor de serviços pode receber o pagamento do cliente no âmbito de um contrato de subscrição e/ou taxação, como exemplos. Adicionalmente ou alternativamente, o provedor de serviços pode receber o pagamento com a venda de conteúdos publicitários a uma ou mais entidades terceiras.

Num aspeto da presente invenção, uma aplicação pode ser implementada para a realização de um ou mais aspetos da presente invenção. Como um exemplo, a implementação de uma aplicação compreende o fornecimento de uma infraestrutura informática operável para realizar um ou mais aspetos da presente invenção.

Como um aspeto adicional da presente invenção, uma infraestrutura informática pode ser implementada compreendendo a integração de código legível por computador num sistema informático, em que o código em combinação com o sistema informático é capaz de realizar um ou mais aspetos da presente invenção.

Ainda como um aspeto adicional da presente invenção, pode ser fornecido um processo para a integração de infraestrutura informática compreendendo a integração de código legível por computador num sistema informático. O sistema informático compreende um suporte legível por computador, em que o suporte de computador compreende um ou mais aspetos da presente invenção. O código em combinação com o sistema informático é capaz de realizar um ou mais

aspetos da presente invenção.

Embora diversas formas de realização sejam acima descritas, são apenas exemplos. Por exemplo, os ambientes informáticos de outras arquiteturas podem incorporar e utilizar um ou mais aspetos da presente invenção. Como exemplos, servidores diferentes dos servidores do Sistema z®, tais como servidores *Power System* ou outros servidores propostos pela *International Business Machines Corporation*, ou servidores de outras companhias podem incluir, utilizar e/ou beneficiar de um ou mais aspetos da presente invenção. Além disso, embora no presente exemplo os adaptadores e o *hub* PCI sejam considerados uma parte do servidor, noutras formas de realização não têm de ser necessariamente considerados uma parte do servidor, mas podem simplesmente ser considerados como estando acoplados à memória de sistema e/ou a outros componentes de um ambiente informático. O ambiente informático não precisa de ser um servidor. Além disso, apesar de serem descritas tabelas de tradução, qualquer estrutura de dados pode ser utilizada e o termo tabela deve incluir todas essas estruturas de dados. Mais ainda, embora os adaptadores sejam baseados em PCI, um ou mais aspetos da presente invenção são utilizáveis com outros adaptadores ou outros componentes de I/O. O adaptador e o adaptador PCI são apenas alguns exemplos. Além disso, espaços de endereçamento, tabelas de endereços e/ou páginas de outros tamanhos podem ser utilizados sem existir afastamento da presente invenção. Além disso, a DTE pode incluir mais, menos ou informação

diferente. Mais ainda, outros tipos de endereços podem ser traduzidos utilizando um ou mais aspetos da presente invenção. Muitas outras variações são possíveis.

Além disso, outros tipos de ambientes informáticos podem beneficiar de um ou mais aspetos da presente invenção. Como um exemplo, um sistema de processamento de dados apropriado para armazenar e/ou executar código de programa é utilizável o qual inclui pelo menos dois processadores acoplados direta ou indiretamente a elementos de memória através de um *bus* de sistema. Os elementos de memória incluem, por exemplo, memória localização empregue durante a execução efetiva do código de programa, armazenamento de grande capacidade, e memória *cache*, que permitem o armazenamento temporário de pelo menos algum código de programa, a fim de reduzir o número de vezes que o código tem de ser obtido do armazenamento de grande capacidade durante a execução.

Dispositivos de Entrada/Saída (*Input/Output* ou I/O) (incluindo, mas não estando limitados a teclados, monitores, dispositivos apontadores, DASD, *tape*, CDs, DVDs, *pen drives* e outros suportes de memória, etc) podem ser acoplados ao sistema diretamente ou através de controladores de I/O intervenientes. Os adaptadores de rede podem também ser acoplados ao sistema para permitir que o sistema de processamento de dados fique acoplado a outros sistemas de processamento de dados remotos ou impressoras ou dispositivos de armazenamento através de redes públicas

ou privadas intervenientes. *Modems*, *modems* de cabo (*cable modems*) e placas *Ethernet* são apenas alguns dos tipos de adaptadores de rede disponíveis.

Com referência à FIG. 11, os componentes representativos de um sistema *Host Computer* 5000 para implementar um ou mais aspetos da presente invenção são representados. O computador *host* 5000 representativo compreende um ou mais CPUs 5001 em comunicação com a memória de computador (isto é, armazenamento central) 5002, bem como interfaces I/O para dispositivos de suporte de armazenamento 5011 e redes 5010 para comunicação com outros computadores ou SANs e similares. O CPU 5001 é compatível com uma arquitetura tendo um conjunto de instruções arquitetado e funcionalidade arquitetada. O CPU 5001 pode ter tradução de endereços dinâmica (Dynamic Address Translation - DAT) 5003 para transformação de endereços de programa (endereços virtuais) em endereços reais de memória. Uma DAT geralmente inclui um *buffer* de tradução (*Translation Lookaside Buffer* - TLB) 5007 para fazer *cache* de traduções para que acessos posteriores ao bloco de memória de computador 5002 não necessitem do atraso de tradução de endereços. Normalmente, uma *cache* 5009 é empregue entre a memória de computador 5002 e o processador 5001. A *cache* 5009 pode ser hierárquica tendo uma *cache* de grande capacidade disponível para mais do que um CPU e *caches* mais pequenas, mais rápidas (nível inferior) entre a *cache* de grande capacidade e cada CPU. Em algumas implementações, as *caches* de nível inferior são divididas

para fornecer *caches* separadas de nível inferior para pesquisa de instruções e acessos de dados. Numa forma de realização, uma instrução é pesquisada a partir da memória 5002 por uma unidade de pesquisa de instruções 5004 através de uma *cache* 5009. A instrução é descodificada numa unidade de descodificação de instruções 5006 e enviada (com outras instruções em algumas formas de realização) para a unidade de execução da instruções. Normalmente, várias unidades de execução 5008 são empregues, por exemplo, uma unidade de execução aritmética, uma unidade de execução de ponto flutuante e uma unidade de execução de instruções de ramificação. A instrução é executada pela unidade de execução, acedendo a operandos a partir de registos de instruções especificadas ou da memória, conforme necessário. Se um operando deve ser acedido (carregado ou armazenado) a partir da memória 5002, uma unidade de carga/armazenamento 5005 normalmente manipula o acesso sob controlo da instrução em execução. As instruções podem ser executadas em circuitos de *hardware* ou em microcódigo interno (*firmware*) ou por uma combinação de ambos.

Como notado, um sistema informático inclui informação em armazenamento localização (ou principal), bem como endereçamento, proteção e de gravação de referências e de alterações. Alguns aspetos de endereçamento incluem o formato dos endereços, o conceito de espaço de endereçamento, os vários tipos de endereços, e a maneira pela qual um tipo de endereço é traduzido para um outro tipo de endereço. Algum armazenamento principal inclui

localizações de armazenamento atribuídas de forma permanente. O armazenamento principal fornece ao sistema armazenamento de dados diretamente endereçável de acesso rápido. Tanto os dados como os programas deverão ser carregados no armazenamento principal (a partir de dispositivos de entrada), antes de poderem ser processados.

O armazenamento principal pode incluir um ou mais armazenamentos de *buffer* mais pequenos e de acesso mais rápido, por vezes chamados de *caches*. Uma *cache* é tipicamente associada de uma forma física com um CPU ou um processador de I/O. Os efeitos, exceto no desempenho, da construção física e utilização de suportes de armazenamento distintos geralmente não são observáveis pelo programa. *Caches* separadas podem ser mantidas para instruções e operandos de dados. A informação dentro de uma *cache* é mantida em *bytes* contíguos num limite integrante chamado bloco de *cache* ou linha de *cache* (ou linha, para abreviar). Um modelo pode proporcionar uma instrução EXTRAIR ATRIBUTO DA CACHE que retorna o tamanho de uma linha de *cache* em *bytes*. Um modelo também pode fornecer as instruções DADOS PRÉVIOS DE PESQUISA e PESQUISA PRÉVIA DE DADOS RELATIVAMENTE LONGOS que efetuam a pesquisa prévia de armazenamento na *cache* de dados ou instruções ou a libertação de dados a partir da *cache*.

O armazenamento é visto como uma longa sequência horizontal de *bits*. Na maioria das operações, os acessos ao armazenamento prosseguem numa sequência da esquerda para a

direita. A sequência de *bits* é subdividida em unidades de oito *bits*. Uma unidade de oito *bits* é chamada de um *byte*, que é o bloco de construção básico de todos os formatos de informação. Cada localização de *byte* em armazenamento é identificada por um número inteiro não negativo único, que é o endereço dessa localização de *byte* ou simplesmente o endereço de *byte*. As localizações de *byte* adjacentes têm endereços consecutivos, começando com 0 na esquerda e prosseguindo numa sequência da esquerda para a direita. Os endereços são inteiros binários sem sinal e têm 24, 31 ou 64 *bits*.

A informação é transmitida entre o armazenamento e um CPU ou um subsistema de canal de um *byte*, ou um grupo de *bytes*, de cada vez. Salvo indicação em contrário, por exemplo, na *z/Architecture®*, um grupo de *bytes* em armazenamento é endereçado pelo *byte* mais à esquerda do grupo. O número de *bytes* no grupo está implícita ou explicitamente especificado pela operação a ser realizada. Quando utilizados numa operação do CPU, um grupo de *bytes* é chamado de um campo. Dentro de cada grupo de *bytes*, por exemplo, na *z/Architecture®*, os *bits* são numerados numa sequência da esquerda para a direita. Na *z/Architecture®*, os *bits* mais à esquerda são por vezes referidos como os *bits* “de ordem superior” e os *bits* mais à direita como os *bits* “de ordem inferior”. Os números de *bits* não são endereços de armazenamento, no entanto. Apenas *bytes* podem ser endereçados. Para operar em *bits* individuais de um *byte* em armazenamento, o *byte* completo é acedido. Os *bits* num

*byte* são numerados de 0 a 7, da esquerda para a direita (por exemplo, na *z/Architecture*®). Os *bits* num endereço podem ser numerados 8-31 ou 40-63 para endereços de 24 *bits*, ou 1-31 ou 33-63 para endereços de 31 *bits*; são numerados 0-63 para endereços de 64 *bits*. Dentro de qualquer outro formato de comprimento fixo de vários *bytes*, os *bits* que compõem o formato são numerados consecutivamente a partir de 0. Para fins de deteção de erros, e de preferência, para a sua correção, um ou mais *bits* de verificação podem ser transmitidos com cada *byte* ou com um grupo de *bytes*. Tais *bits* de verificação são gerados automaticamente pela máquin e não podem ser controlados diretamente pelo programa. As capacidades de armazenamento são expressas em número de *bytes*. Quando o comprimento de um campo de operando de armazenamento está implícito pelo código de operação de uma instrução, é dito que o campo tem um comprimento fixo, que pode ser de um, dois, quatro, oito ou dezasseis *bytes*. Campos maiores podem estar implícitos para algumas instruções. Quando o comprimento de um campo de operando de armazenamento não está implícito mas é expressamente referido, é dito que o campo tem um comprimento variável. Os operandos de comprimento variável podem variar em comprimento em incrementos de um *byte* (ou com algumas instruções, em múltiplos de dois *bytes* ou outros múltiplos). Quando a informação é colocada em armazenamento, os conteúdos de apenas aquelas localizações de *byte* são substituídos que estão incluídos no campo designado, embora a largura do caminho físico para o armazenamento possa ser maior do que o comprimento do campo

a ser armazenado.

Certas unidades de informação devem ficar num limite integrante em armazenamento. Um limite é chamado integrante para uma unidade de informação quando o seu endereço de armazenamento é um múltiplo do comprimento da unidade em *bytes*. Nomes especiais são dados aos campos de 2, 4, 8 e 16 *bytes* num limite integrante. Uma meia palavra (*halfword*) é um grupo de dois *bytes* consecutivos num limite de dois *bytes* e é o bloco de construção básico de instruções. Uma palavra é um grupo de quatro *bytes* consecutivos num limite de quatro *bytes*. Uma palavra dupla (*doubleword*) é um grupo de oito *bytes* consecutivos num limite de oito *bytes*. Uma palavra quádrupla (*quadword*) é um grupo de 16 *bytes* consecutivos num limite de 16 *bytes*. Quando o armazenamento designa meias palavras, palavras, palavras duplas e palavras quádruplas, a representação binária do endereço contém um, dois, três ou quatro ou mais *bits* com zero mais à direita, respetivamente. As instruções devem estar em limites integrantes de dois *bytes*. Os operandos de armazenamento da maioria das instruções não têm requisitos de alinhamento de limite.

Em dispositivos que implementam *caches* separadas para instruções e operandos de dados, um atraso significativo pode ser experimentado se um programa armazena numa linha de *cache* a partir da qual as instruções são posteriormente pesquisadas, independentemente do armazenamento alterar as instruções que são posteriormente

pesquisadas.

Numa forma de realização, a invenção pode ser praticada por *software* (por vezes referido como código interno licenciado, *firmware*, microcódigo, milicódigo, picocódigo e similares, qualquer um dos quais seria consistente com a presente invenção). Com referência à FIG. 11, o código de programa de *software* que incorpora a presente invenção é normalmente acedido pelo processador 5001 do sistema *host* 5000 a partir de dispositivos de suporte de armazenamento de longa duração 5011, tais como uma unidade de CD-ROM, unidade de *tape* ou um disco rígido. O código de programa de *software* pode ser incorporado em qualquer um de uma variedade de suportes conhecidos para utilização com um sistema de processamento de dados, tais como uma disquete, um disco rígido ou um CD-ROM. O código pode ser distribuído em tais suportes, ou pode ser distribuído aos utilizadores a partir da memória de computador 5002 ou armazenamento de um sistema informático através de uma rede 5010 para outros sistemas informáticos para utilização por utilizadores de tais outros sistemas.

O código de programa de *software* inclui um sistema operativo que controla a função e interação dos vários componentes do computador e um ou mais programas aplicativos. O código do programa é normalmente paginado a partir do dispositivo de suporte de armazenamento 5011 para o armazenamento de computador relativamente mais rápido 5002 onde se encontra disponível para o processamento pelo

processador 5001. As técnicas e os métodos para incorporar o código de programa de *software* na memória, em suportes físicos e/ou para distribuir código de *software* através de redes são bem conhecidos e não irão ser mais discutidos aqui. O código de programa, quando criado e armazenado num suporte material (incluindo, mas não estando limitado a módulos de memória eletrónica (RAM), memória *flash*, Discos Compactos (CDs), DVDs, Fitas (*Tapes*) Magnéticas e similares é muitas vezes referido como um "produto de programa de computador".

O suporte de produto de programa de computador é tipicamente legível por um circuito de processamento, de preferência num sistema informático para execução pelo circuito de processamento.

A FIG. 12 ilustra um sistema de *hardware* de estação de trabalho ou servidor representativos no qual a presente invenção pode ser praticada. O sistema 5020 da FIG. 12 compreende um sistema informático base representativo 5021, tal como um computador pessoal, uma estação de trabalho ou um servidor, incluindo dispositivos periféricos opcionais. O sistema informático base 5021 inclui um ou mais processadores 5026 e um *bus* utilizado para conectar e permitir a comunicação entre o(s) processador(es) 5026 e os outros componentes do sistema 5021 de acordo com técnicas conhecidas. O *bus* conecta o processador 5026 à memória 5025 e ao armazenamento de longa duração 5027 que pode incluir um disco rígido (incluindo

qualquer suporte magnético, CD, DVD e Memória *Flash*, por exemplo) ou uma unidade de *tape*, por exemplo. O sistema 5021 pode incluir também um adaptador de interface de utilizador, que conecta o microprocessador 5026 através do *bus* para um ou mais dispositivos de interface, tais como um teclado 5024, um rato 5023, uma impressora/*scanner* 5030 e/ou outros dispositivos de interface, que podem ser qualquer dispositivo de interface de utilizador, tal como uma tela sensível ao toque, um *pad* de entrada digitalizada, etc. O *bus* também conecta um dispositivo de visualização 5022, tal como uma tela ou monitor LCD, para o microprocessador 5026 através de um adaptador de visualização.

O sistema 5021 pode comunicar com outros computadores ou redes de computadores através de um adaptador de rede capaz de comunicar 5028 com uma rede 5029. Adaptadores de rede exemplificativos são canais de comunicação, *Token Ring*, *Ethernet* ou *modems*. Alternativamente, o sistema 5021 pode comunicar através de uma interface sem fios, tal como um cartão CDPD (dados de pacote digital celular - *cellular digital packet data*). O sistema 5021 pode ser associado com tais outros computadores numa Rede de Área Local (*Local Area Network* - LAN) ou numa Rede de Área Ampla (*Wide Area Network* - WAN), ou o sistema 5021 pode ser um cliente numa disposição cliente/servidor com outro computador, etc. Todas estas configurações, bem como o *hardware* e o *software* de comunicações apropriado, são conhecidos na técnica.

A FIG. 13 ilustra uma rede de processamento de dados 5040 na qual a presente invenção pode ser praticada. A rede de processamento de dados 5040 pode incluir uma pluralidade de redes individuais, tais como uma rede sem fios e uma rede com fios, cada uma das quais pode incluir uma pluralidade de estações de trabalho individuais, 5041, 5042, 5043, 5044. Além disso, conforme os peritos na técnica apreciarão, uma ou mais LANs podem ser incluídas, onde uma LAN pode compreender uma pluralidade de estações de trabalho inteligentes acopladas a um processador host.

Ainda com referência à FIG. 13, as redes também podem incluir computadores ou servidores *mainframe*, tais como um computador *gateway* (cliente servidor 5046) ou servidor de aplicação (servidor remoto 5048 que pode aceder a um repositório de dados e também pode ser acedido iretamente a partir de uma estação de trabalho 5045). Um computador *gateway* 5046 serve como um ponto de entrada em cada rede individual. Uma *gateway* é necessária ao conectar um protocolo de rede a outro. A *gateway* 5046 pode ser preferencialmente acoplada a outra rede (a Internet 5047, por exemplo) por meio de uma ligação de comunicações. A *gateway* 5046 também pode ser diretamente acoplada a uma ou mais estações de trabalho 5041, 5042, 5043, 5044 utilizando uma ligação de comunicações. O computador *gateway* pode ser implementado utilizando um servidor IBM eServer™ System z® disponível a partir da *International Business Machines Corporation*.

Com referência simultaneamente à FIG. 12 e FIG. 13, o código de programação de *software*, que pode incorporar a presente invenção pode ser acedido pelo processador 5026 do sistema 5020 a partir do suporte de armazenamento de longa duração 5027, tal como uma unidade de CD-ROM ou disco rígido. O código de *software* de programação pode ser incorporado em qualquer um de uma variedade de suportes conhecidos para utilização com um sistema de processamento de dados, tal como uma disquete, um disco rígido ou um CD-ROM. O código pode ser distribuído em tais suportes, ou pode ser distribuído aos utilizadores 5050, 5051 a partir da memória ou armazenamento de um sistema informático através de uma rede para outros sistemas informáticos para utilização por utilizadores de tais outros sistemas.

Em alternativa, o código de programação pode ser incorporado na memória 5025, e acedido pelo processador 5026 utilizando o *bus* do processador. Tal código de programação inclui um sistema operativo que controla a função e interação dos vários componentes do computador e um ou mais programas aplicativos 5032. O código de programa é normalmente paginado a partir do suporte de armazenamento 5027 para a memória de alta velocidade 5025, onde está disponível para processamento pelo processador 5026. As técnicas e métodos de programação para incorporar o código de programação de *software* na memória, sobre suporte físico e/ou distribuir código de *software* através de redes são bem conhecidos e não irão ser mais discutidos aqui. O código de

programa, quando criado e armazenado num suporte material (incluindo, mas não estando limitado a módulos de memória eletrónica (RAM), memória *flash*, Discos Compactos (CDs), DVDs, *Tapes* Magnéticas e similares é muitas vezes referido como um "produto de programa de computador". O suporte de produto de programa de computador é tipicamente legível por um circuito de processamento, de preferência num sistema informático para execução pelo circuito de processamento.

A *cache* que está mais prontamente disponível para o processador (normalmente mais rápida e mais pequena do que outras *caches* do processador) é a *cache* mais baixa (L1 ou nível 1) e o armazenamento principal (memória principal) é a *cache* de nível mais elevado (L3 se existirem 3 níveis). A *cache* de nível mais baixo é frequentemente dividida numa *cache* de instruções (*I-Cache*) mantendo instruções de máquina a serem executadas e uma *cache* de dados (*D-Cache*) mantendo operandos de dados.

Com referência à FIG. 14, uma forma de realização de um processador exemplar é representada para o processador 5026. Tipicamente um ou mais níveis da *cache* 5053 são utilizados para colocar em *buffer* blocos de memória, a fim de melhorar o desempenho do processador. A *cache* 5053 é um *buffer* de alta velocidade mantendo linhas de *cache* de dados de memória que são suscetíveis de ser utilizadas. As linhas de *cache* típicas têm 64, 128 ou 256 *bytes* de dados de memória. *Caches* separadas são mais frequentemente utilizadas para armazenar em *cache* as

instruções do que armazenar em *cache* os dados. A coerência da *cache* (sincronização de cópias de linhas na memória e as *caches*) é muitas vezes fornecida por vários algoritmos "*snoop*" bem conhecidos na técnica. O armazenamento de memória principal 5025 de um sistema de processador é muitas vezes referido como uma *cache*. Num sistema de processador tendo 4 níveis de *cache* 5053, o armazenamento principal 5025 é por vezes referido como a *cache* de nível 5 (L5), uma vez que é geralmente mais rápida e apenas mantém uma porção de armazenamento não-volátil (DASD, *tape*, etc) que está disponível a um sistema informático. O armazenamento principal 5025 armazena em *cache* páginas de dados paginados dentro e fora do armazenamento principal 5025 pelo sistema operativo.

Um contador de programa (contador de instruções) 5061 mantém o controlo do endereço da instrução atual a ser executada. Um contador de programa num processador de *z/Arquitecture*® tem 64 *bits* e pode ser truncado para 31 ou 24 *bits* para suportar limites de endereçamento anteriores. Um contador de programa é normalmente incorporado numa PSW (palavra de estado do programa - *program status word*) de um computador, que persiste durante a comutação de contexto. Assim, um programa em curso, tendo um valor de contador do programa, pode ser interrompido, por exemplo, pelo sistema operativo (comutação de contexto do ambiente do programa para o ambiente do sistema operativo). A PSW do programa mantém o valor de contador do programa enquanto o programa não estiver ativo, e o contador de programa (na PSW) do

sistema operativo é utilizado enquanto o sistema operativo está em execução. Tipicamente, o contador de programa é incrementado por um valor igual ao número de *bytes* da instrução atual. As instruções RISC (Computação de Conjunto Reduzido de Instruções – *Reduced Instruction Set Computing*) são normalmente de comprimento fixo, enquanto as instruções CISC (Computação de Conjunto Complexo de Instruções – *Complex Instruction Set Computing*) são normalmente de tamanho variável. As instruções da IBM *z/Architecture*® são instruções CISC com um comprimento de 2, 4 ou 6 *bytes*. O contador de programa 5061 é modificado ou por uma operação de comutação de contexto ou por uma operação de retirada de ramificação de uma instrução de ramificação, por exemplo. Numa operação de comutação de contexto, o valor do contador de programa atual é gravado na palavra de estado do programa, juntamente com outras informações de estado sobre o programa que está a ser executado (tais como códigos de condição), e um novo valor de contador do programa é carregado apontando para uma instrução de um novo módulo de programa a ser executado. Uma operação de retirada de ramificação é realizada a fim de permitir que o programa tome decisões ou entre em ciclo (*loop*) dentro do programa através do carregamento do resultado da instrução de ramificação no contador de programa 5061.

Normalmente, uma unidade de pesquisa de instruções 5055 é utilizada para pesquisar instruções em nome do processador 5026. A unidade de pesquisa ou vai pesquisar “instruções sequenciais seguintes”, instruções

alvo de instruções retiradas de ramificação, ou primeiras instruções de um programa depois de uma comutação de contexto. As unidades de pesquisa Instruções Modernas muitas vezes empregam técnicas de pesquisa prévia para pesquisar previamente de forma especulativa instruções baseadas na probabilidade das instruções pesquisadas previamente poderem ser utilizadas. Por exemplo, uma unidade de pesquisa pode pesquisar 16 *bytes* de instrução que inclui a próxima instrução sequencial e *bytes* adicionais de instruções sequenciais adicionais.

As instruções pesquisadas são então executadas pelo processador 5026. Numa forma de realização, a(s) instrução(ões) pesquisada(s) são passadas a uma unidade de envio 5056 da unidade de pesquisa. A unidade de envio descodifica a(s) instrução(ões) e encaminha a informação sobre a(s) instrução(ões) descodificada(s) para unidades apropriadas 5057, 5058, 5060. Uma unidade de execução 5057 irá receber tipicamente informações sobre instruções aritméticas descodificadas a partir da unidade de pesquisa de instruções 5055 e irá realizar operações aritméticas em operandos de acordo com o código de operação da instrução. Os operandos são fornecidos à unidade de execução 5057, de preferência ou a partir da memória 5025, registos arquitetados 5059 ou a partir de um campo imediato da instrução a ser executada. Os resultados da execução, quando armazenados, são armazenados na memória 5025, registos 5059 ou em outro *hardware* da máquina (tal como registos de controlo, registos PSW e afins).

Um processador 5026 tem tipicamente uma ou mais unidades 5057, 5058, 5060 para executar a função da instrução. Com referência à FIG. 15A, uma unidade de execução 5057 pode comunicar com os registos gerais arquitetados 5059, uma unidade de descodificação/envio 5056, uma unidade de armazenamento de carga 5060, e outras unidades de processador 5065 por meio de uma lógica de interface 5071. Uma unidade de execução 5057 pode empregar vários circuitos de registo 5067, 5068, 5069 para manter informação de que a unidade aritmética lógica (*Arithmetic Logic Unit* - ALU) 5066 irá operar. A ALU realiza operações aritméticas tais como somar, subtrair, multiplicar e dividir assim como funções lógicas tais como AND, OR e ou-exclusivo (XOR), rodar e deslocar. De preferência, a ALU suporta operações especializadas que são dependentes do desenho. Outros circuitos podem fornecer outras condições arquitetadas 5072, incluindo códigos de condição e lógica de suporte à recuperação, por exemplo. Tipicamente, o resultado de uma operação ALU é mantido num circuito de registo de saída 5070 que pode encaminhar o resultado para uma variedade de outras funções de processamento. Existem muitas disposições de unidades de processador, a presente descrição destina-se apenas a fornecer uma compreensão representativa de uma forma de realização.

Uma instrução ADD (Adicionar), por exemplo, seria executada numa unidade de execução 5057 tendo funcionalidade aritmética e lógica enquanto uma instrução de ponto flutuante, por exemplo, seria executada numa

execução de ponto flutuante tendo capacidade especializada de ponto flutuante. De preferência, uma unidade de execução opera em operandos identificados por uma instrução através da realização de uma função definida pelo código de operação nos operandos. Por exemplo, uma instrução ADD pode ser executada por uma unidade de execução 5057 em operandos encontrados em dois registos 5059 identificados por campos do registo da instrução.

A unidade de execução 5057 realiza a adição aritmética sobre dois operandos e armazena o resultado num terceiro operando em que o terceiro operando pode ser um terceiro registo ou um dos dois registos de origem. A unidade de execução de preferência utiliza uma Unidade Aritmética Lógica (*Arithmetic Logic Unit* – ALU) 5066 que é capaz de realizar uma variedade de funções lógicas tais como o Deslocamento (Shift), Rotação (Rotate), E (And), Ou (Or) e Ou-exclusivo (eXclusive-OR – XOR), bem como uma variedade de funções algébricas incluindo qualquer uma das funções de somar, subtrair, multiplicar, dividir. Algumas ALUs 5066 são desenhadas para operações escalares e algumas para ponto flutuante. Os dados podem ser *Big Endian* (onde o *byte* menos significativo está no endereço de *byte* mais elevado) ou *Little Endian* (onde o *byte* menos significativo está no endereço de *byte* mais baixo), dependendo da arquitetura. A IBM *z/Architecture*® é *Big Endian*. Os campos com sinal podem ser sinal e magnitude, complemento de 1's ou complemento de 2's dependendo da arquitetura. Um número de complemento de 2's é vantajoso pelo facto de a ALU não

necessitar de uma capacidade de subtrair uma vez que tanto um valor negativo como um valor positivo em complemento de 2's requer apenas uma adição dentro da ALU. Os números são normalmente descritos na forma abreviada, onde um campo de 12 *bits* define o endereço de um bloco de 4096 *bytes* e é descrito geralmente como um bloco de 4 Kbytes (Kilobyte), por exemplo.

Com referência à FIG. 15B, a informação de instrução de ramificação para a execução de uma instrução de ramificação é normalmente enviada para uma unidade de ramificação 5058 que muitas vezes emprega um algoritmo de previsão de ramificações, tal como uma tabela de histórico de ramificação 5082 para prever o resultado da ramificação antes de outras operações condicionais estarem completas. O alvo da instrução de ramificação atual será pesquisado e especulativamente executado antes das operações condicionais estarem concluídas. Quando as operações condicionais estão concluídas as instruções de ramificação especulativamente executadas são ou concluídas ou descartadas com base nas condições da operação condicional e no resultado especulado. Uma instrução de ramificação típica pode testar códigos de condição e ramificar para um endereço alvo, se os códigos de condição cumprirem a exigência de ramificação da instrução de ramificação, um endereço alvo pode ser calculado com base em vários números, incluindo os encontrados nos campos de registos ou um campo imediato da instrução por exemplo. A unidade de ramificação 5058 pode empregar uma ALU 5074 tendo uma

pluralidade de circuitos de registo de entrada 5075, 5076, 5077 e um circuito de registo de saída 5080. A unidade de ramificação 5058 pode comunicar com os registos gerais 5059, unidade de envio de descodificação 5056 ou outros circuitos 5073, por exemplo.

A execução de um conjunto de instruções pode ser interrompida por uma variedade de razões incluindo uma comutação de contexto iniciada por um sistema operativo, um erro ou exceção de programa provocando uma comutação de contexto, um sinal de interrupção de I/O que provoca uma comutação de contexto ou uma atividade multi-tarefa de uma pluralidade de programas (num ambiente multi-tarefa), por exemplo. De preferência, uma ação de comutação de contexto grava informação de estado sobre um programa atualmente em execução e em seguida carrega informação de estado sobre outro programa que está a ser invocado. A informação de estado pode ser gravada em registos de *hardware* ou na memória, por exemplo. A informação de estado preferencialmente compreende um valor de contador do programa apontando para a próxima instrução a ser executada, códigos de condição, informação de tradução em memória e conteúdos de registos arquitetados. Uma atividade de comutação de contexto pode ser exercida por circuitos de *hardware*, programas aplicativos, programas de sistemas operativos ou código de *firmware* (microcódigo, picocódigo ou código interno licenciado (LIC)) isoladamente ou em combinação.

Um processador acede a operandos de acordo com os métodos definidos na instrução. A instrução pode fornecer um operando imediato utilizando o valor de uma porção da instrução, pode fornecer um ou mais campos de registo que apontam ou para registos de propósito geral ou registos de propósito específico (registos de ponto flutuante, por exemplo). A instrução pode utilizar registos implícitos identificados por um campo *opcode* como operandos. A instrução pode utilizar localizações de memória para operandos. Uma localização de memória de um operando pode ser fornecida por um registo, um campo imediato, ou uma combinação dos registos e campo imediato tal como exemplificado pelas condições longas de deslocamento da *z/Architecture*® em que a instrução define um registo base, um registo de índice e um campo imediato (campo de deslocamento), que são adicionados em conjunto para fornecer o endereço do operando na memória, por exemplo. A localização aqui normalmente implica uma localização na memória principal (armazenamento principal), salvo indicação em contrário.

Com referência à FIG. 15C, um processador acede ao armazenamento utilizando uma unidade de carga/armazenamento 5060. A unidade de carga/armazenamento 5060 pode realizar uma operação de carga através da obtenção do endereço do operando alvo na memória 5053 e carregando o operando num registo 5059 ou noutra localização da memória 5053, ou pode realizar uma operação de armazenamento mediante a obtenção do endereço do

operando alvo na memória 5053 e armazenando os dados obtidos a partir de um registo 5059 ou outra localização da memória 5053 na localização do operando alvo na memória 5053. A unidade de carga/armazenamento 5060 pode ser especulativa e pode aceder à memória numa sequência que está fora de ordem em relação à sequência de instruções, no entanto a unidade de carga/armazenamento 5060 deve manter a aparência aos programas que as instruções foram executadas em ordem. Uma unidade de carga/armazenamento 5060 pode comunicar com os registos gerais 5059, a unidade de descodificação/envio 5056, a interface *cache*/memória 5053 ou outros elementos 5083 e compreende vários circuitos de registo, ALUs 5085 e lógica de controlo 5090 para calcular os endereços de armazenamento e para fornecer sequenciação encadeada para manter as operações em ordem. Algumas operações podem estar fora de ordem, mas a unidade de carga/armazenamento fornece funcionalidade para fazer com que as operações fora de ordem apareçam no programa como tendo sido realizadas por ordem, como é bem conhecido na técnica.

Preferivelmente os endereços que um programa aplicativo “vê” são muitas vezes referidos como endereços virtuais. Os endereços virtuais são muitas vezes referidos como “endereços lógicos” e “endereços efetivos”. Estes endereços virtuais são virtuais no sentido em que eles são redirecionados para uma localização de memória física por uma de uma variedade de tecnologias de tradução de endereços dinâmica (*Dynamic Address Translation* – DAT),

incluindo, mas não estando limitada a simplesmente colocar um prefixo num endereço virtual com um valor de deslocamento, traduzir o endereço virtual através de uma ou mais tabelas de tradução, as tabelas de tradução compreendendo preferencialmente pelo menos uma tabela de segmento e uma tabela de página isoladamente ou em combinação, preferencialmente a tabela de segmento tendo uma entrada apontando para a tabela de página. Na *z/Architecture®*, uma hierarquia de tradução é fornecida incluindo uma primeira tabela de região, uma segunda tabela de região, uma terceira tabela de região, uma tabela de segmento e uma tabela de página opcional. O desempenho da tradução de endereços é muitas vezes melhorado através da utilização de um *buffer* de tradução (*Translation Lookaside Buffer* - TLB) que compreende entradas mapeando um endereço virtual para uma localização de memória física associada. As entradas são criadas quando a DAT traduz um endereço virtual utilizando as tabelas de tradução. Uma utilização posterior do endereço virtual pode então utilizar a entrada da TLB rápida em vez dos acessos sequenciais lentos à tabela de tradução. O conteúdo do TLB pode ser gerido por uma variedade de algoritmos de substituição, incluindo LRU (Utilizado Há Mais Tempo - *Least Recently Used*).

No caso em que o processador é um processador de um sistema multi-processador, cada processador tem a responsabilidade de manter recursos partilhados, tais como I/O, *caches*, TLBs e memória, interligados para coerência. Normalmente, tecnologias “snoop” serão utilizadas na

manutenção da coerência de *cache*. Num ambiente *snoop*, cada linha de *cache* pode ser marcada como estando em qualquer um dos estados partilhado, exclusivo, alterado, inválido e similar, a fim de facilitar a partilha.

As unidades de I/O 5054 (FIG. 14) fornecem ao processador meios para acoplagem a dispositivos periféricos, incluindo *tape*, disco, impressoras, monitores e redes, por exemplo. As unidades de I/O são frequentemente apresentadas ao programa de computador por *drivers* de *software*. Nos computadores centrais (*mainframes*), tais como o *System z*® da IBM®, os adaptadores de canal e os adaptadores de sistema aberto são unidades de I/O do *mainframe* que fornecem as comunicações entre o sistema operativo e os dispositivos periféricos.

Além disso, outros tipos de ambientes informáticos podem beneficiar de um ou mais aspetos da presente invenção. Como um exemplo, um ambiente pode incluir um emulador (por exemplo, *software* ou outros mecanismos de emulação), em que uma arquitetura específica (incluindo, por exemplo, execução de instruções, funções arquitetadas, tal como tradução de endereços, e registos arquitetados) ou um subconjunto dela é emulada (por exemplo, num sistema informático nativo tendo um processador e memória). Num tal ambiente, uma ou mais funções de emulação do emulador podem implementar um ou mais aspetos da presente invenção, muito embora um computador executando o emulador possa ter uma arquitetura

diferente das capacidades a serem emuladas. Como um exemplo, num modo de emulação, a instrução ou operação específica a ser emulada é descodificada, e uma função de emulação apropriada é construída para implementar a instrução ou operação individual.

Num ambiente de emulação, um computador *host* inclui, por exemplo, uma memória para armazenar instruções e dados; uma unidade de pesquisa de instruções para pesquisar instruções a partir da memória e para opcionalmente fornecer *buffering* local para a instrução pesquisada; uma unidade de descodificação de instruções para receber as instruções pesquisadas e para determinar o tipo de instruções que foram pesquisadas; e uma unidade de execução de instruções para executar as instruções. A execução pode incluir o carregamento de dados num registo a partir da memória; o armazenamento de dados de volta para a memória a partir de um registo; ou a realização de algum tipo de operação aritmética ou lógica, tal como determinado pela unidade de descodificação. Num exemplo, cada unidade é implementada em *software*. Por exemplo, as operações sendo realizadas pelas unidades são implementadas como uma ou mais subrotinas no *software* de emulação.

Mais particularmente, num *mainframe*, as instruções de máquina arquitetadas são utilizadas por programadores, normalmente hoje em dia programadores de "C", muitas vezes por meio de um aplicativo compilador. Estas instruções armazenadas no suporte de armazenamento

podem ser executadas nativamente num servidor IBM® *z/Architecture®*, ou alternativamente em máquinas que executam outras arquiteturas. Elas podem ser emuladas nos servidores IBM® *mainframe* existentes e em futuros, e também em outras máquinas da IBM® (por exemplo, servidores *Power Systems* e Servidores System x®). Podem ser executadas em máquinas a correr Linux numa ampla variedade de máquinas que utilizam *hardware* fabricado pela IBM®, Intel®, AMD™, e outros. Para além da execução nesse *hardware* sob uma arquitetura *z/Arquitecture®*, também pode ser utilizado Linux, bem como máquinas que utilizam emulação por *TurboHercules* (www.turbohercules.com/), *Hercules* (www.hercules-390.org/) ou FSI (*Fundamental Software, Inc*) (www.funsoft.com/), onde geralmente a execução está num modo de emulação. No modo de emulação, o *software* de emulação é executado por um processador nativo para emular a arquitetura de um processador emulado.

O processador nativo normalmente executa *software* de emulação compreendendo ou *firmware* ou um sistema operativo nativo para realizar a emulação do processador emulado. O *software* de emulação é responsável por pesquisar e executar instruções da arquitetura do processador emulado. O *software* de emulação mantém um contador de programa emulado para manter o controlo dos limites das instruções. O *software* de emulação pode pesquisar uma ou mais instruções de máquina emulada de cada vez e converter uma ou mais dessas instruções de máquina emulada para um grupo correspondente de instruções de máquina nativas para

execução pelo processador nativo. Estas instruções convertidas podem ser colocadas em *cache* de modo a que possa ser conseguida uma tradução mais rápida. Apesar disso, o *software* de emulação deve manter as regras de arquitetura da arquitetura do processador emulado de forma a assegurar os sistemas operativos e aplicações escritos para o processador emulado funcionar corretamente. Além disso, o *software* de emulação deve fornecer recursos identificados pela arquitetura do processador emulado, incluindo, mas não estando limitados a registos de controlo, registos de propósito geral, registos de ponto flutuante, função de tradução de endereços dinâmica incluindo tabelas de segmentos e tabelas de páginas por exemplo, mecanismos de interrupção, mecanismos de comutação de contexto, relógios de Hora do Dia (*Time of Day* – TOD) e interfaces arquitetadas para subsistemas de I/O tal que um sistema operativo ou um programa aplicativo desenvolvido para correr no processador emulado, possa ser executado no processador nativo que tem o *software* de emulação.

Uma instrução específica que esteja a ser emulada é descodificada, e uma subrotina é chamada a realizar a função da instrução individual. Uma função de *software* de emulação que emula uma função de um processador emulado é implementada, por exemplo, numa subrotina "C" ou *driver*, ou qualquer outro método de fornecimento de um *driver* para o *hardware* específico pois estará dentro da habilidade dos peritos na técnica, após compreenderem a descrição da forma de realização preferida. Várias patentes de emulação de

*software* e *hardware*, incluindo, mas não estando limitadas à Carta-Patente dos EUA N° 5 551 013, intitulada "*Multiprocessor for Hardware Emulation*", por Beausoleil et al.; e Carta-Patente dos EUA N° 6 009 261, intitulada "*Preprocessing of Stored Target Routines for Emulating Incompatible Instructions on a Target Processor*", por Scalzi et al; e a Carta-Patente dos EUA N° 5 574 873, intitulada "*Decoding Guest Instruction to Directly Access Emulation Routines that Emulate the Guest Instructions*", por Davidian et al; e a Carta-Patente dos EUA N° 6 308 255, intitulada "*Symmetrical Multiprocessing Bus and Chipset Used for Coprocessor Support Allowing Non-Native Code to Run in a System*", por Gorishek et al; e a Carta-Patente dos EUA N° 6 463 582, intitulada "*Dynamic Optimizing Object Code Translator for Arquitecture Emulation and Dynamic Optimizing Object Code Translation Method*", por Lethin et al; e a Carta-Patente dos EUA N° 5 790 825, intitulada "Method for Emulating Guest Instructions on a Host Computer Through Dynamic Recompilation of Host Instructions", por Eric Traut; e muitas outras, ilustram uma variedade de formas conhecidas para conseguir uma emulação de um formato de instrução arquitetado para uma máquina diferente para uma máquina alvo disponível para os peritos na técnica.

Na FIG. 16, um exemplo de um sistema informático *host* emulado 5092 é fornecido o qual emula um sistema informático *host* 5000' de uma arquitetura *host*. No sistema informático *host* emulado 5092, o processador *host* (CPU) 5091 é um processador *host* emulado (ou processador *host*

virtual), e compreende um processador de emulação 5093 tendo uma arquitetura de conjunto de instruções nativo diferente do que a do processador 5091 do computador *host* 5000'. O sistema informático *host* emulado 5092 tem a memória 5094 acessível ao processador de emulação 5093. Na forma de realização exemplificativa, a memória 5094 é particionada numa porção de memória de um computador *host* 5096 e numa porção de rotinas de emulação 5097. A memória de computador *host* 5096 está disponível para os programas do computador *host* emulado 5092 de acordo com a arquitetura do computador *host*. O processador de emulação 5093 executa instruções nativas de um conjunto de instruções arquitetado de uma arquitetura diferente da do processador emulado 5091, as instruções nativas obtidas a partir da memória de rotinas de emulação 5097, e pode aceder a uma instrução de *host* para execução a partir de um programa na memória de computador *host* 5096, empregando uma ou mais instruções obtidas numa rotina de sequência & acesso/descodificação que pode descodificar as instruções do *host* acedidas para determinar uma rotina de execução de instruções nativas para emular a função da instrução de *host* acedida. Outras condições que são definidas para a arquitetura do sistema informático *host* 5000' podem ser emuladas por rotinas de condições arquitetadas, incluindo condições tais como registos de propósito geral, registos de controlo, tradução de endereços dinâmica e suporte do subsistema de I/O e *cache* do processador, por exemplo. As rotinas de emulação também podem tirar proveito de funções disponíveis no processador de emulação 5093 (tais como registos gerais e

tradução dinâmica de endereços virtuais) para melhorar o desempenho das rotinas de emulação. Pode também ser fornecido *hardware* especial e motores de libertação de carga (*off-load engines*) para auxiliar o processador 5093 na emulação da função do computador host 5000'.

A terminologia aqui utilizada existe para o propósito de descrever apenas formas de realização específicas e não tem a intenção de ser uma limitação da invenção. Tal como aqui utilizadas, as formas singulares "um", "uma", "a" ou "o" pretendem também incluir as formas plurais, a menos que o contexto indique claramente o contrário. Será adicionalmente compreendido que os termos "compreende" e/ou "compreendendo", quando utilizados na presente especificação, especificam a presença de características, números inteiros, etapas, operações, elementos e/ou componentes indicados, mas não impedem a presença ou a adição de uma ou mais outras características, números inteiros, etapas, operações, elementos, componentes e/ou grupos da mesma.

Lisboa, 27 de Setembro de 2013

# REIVINDICAÇÕES

1. Um método de tradução de endereços num ambiente informático, o referido método compreendendo:

obter um endereço a partir de um adaptador (110) a ser traduzido para um endereço de memória diretamente utilizável no acesso à memória de sistema do ambiente informático, o endereço compreendendo uma pluralidade de *bits*, a pluralidade de *bits* compreendendo uma primeira porção de *bits* e uma segunda porção de *bits*;

receber um valor de intervalo de endereços que indica um intervalo de endereços permitidos, em que o intervalo é definido por um endereço base (214) e um limite (216) localizado numa entrada de tabela de dispositivo associada com o adaptador, a entrada de tabela de dispositivo (210) localizada por um identificador de solicitante localizado num pedido emitido pelo adaptador;

validar o endereço obtido a partir do adaptador utilizando pelo menos a primeira porção de *bits* e o intervalo de endereços recebido; e

converter o endereço obtido a partir do adaptador para o endereço de memória diretamente utilizável no acesso à memória de sistema do ambiente informático, o método sendo caracterizado pela conversão ignorar a

primeira porção de bits e utilizar a segunda porção de bits para obter informação de endereços a partir de um ou mais níveis de tabelas de tradução de endereços para realizar a conversão.

2. O método da reivindicação 1, em que a primeira porção de *bits* compreende *bits* de ordem superior do endereço e a segunda porção de *bits* compreende *bits* de ordem inferior do endereço, os *bits* de ordem inferior determinados com base num tamanho de um espaço de endereçamento atribuído que inclui o endereço de memória.

3. O método da reivindicação 1, em que um número de níveis de tabelas de tradução de endereços é baseado em pelo menos um tamanho dos tamanhos de um espaço de endereçamento atribuído que inclui o endereço de memória, um tamanho de uma ou mais tabelas de tradução de endereços a serem utilizadas na conversão, e um tamanho da unidade de memória acedida pelo endereço de memória.

4. O método da reivindicação 1, em que a conversão compreende selecionar uma tabela de tradução de endereços para ser utilizada para converter o endereço, a seleção utilizando um apontador numa entrada de tabela de dispositivo (210) utilizada na tradução.

5. O método da reivindicação 4, em que o método compreende ainda a localização da entrada de tabela de dispositivo (210), a localização utilizando pelo menos um identificador do solicitante do adaptador que emite um

pedido que inclui o endereço a ser traduzido ou uma porção do endereço.

6. O método da reivindicação 4, em que o método compreende ainda a determinação de um formato da tabela de tradução de endereços selecionada, a determinação utilizando um campo de formato da entrada de tabela de dispositivo (210).

7. Um produto de programa de computador para traduzir endereços num ambiente informático, o produto de programa de computador compreendendo:

um suporte de armazenamento legível por computador legível por um circuito de processamento e instruções de armazenamento para execução pelo circuito de processamento para a realização de um método de acordo com qualquer uma das reivindicações 1 a 6.

8. Um sistema compreendendo meios adaptados para levar a cabo todas as etapas do método de acordo com qualquer reivindicação do método precedente.

Lisboa, 27 de Setembro de 2013

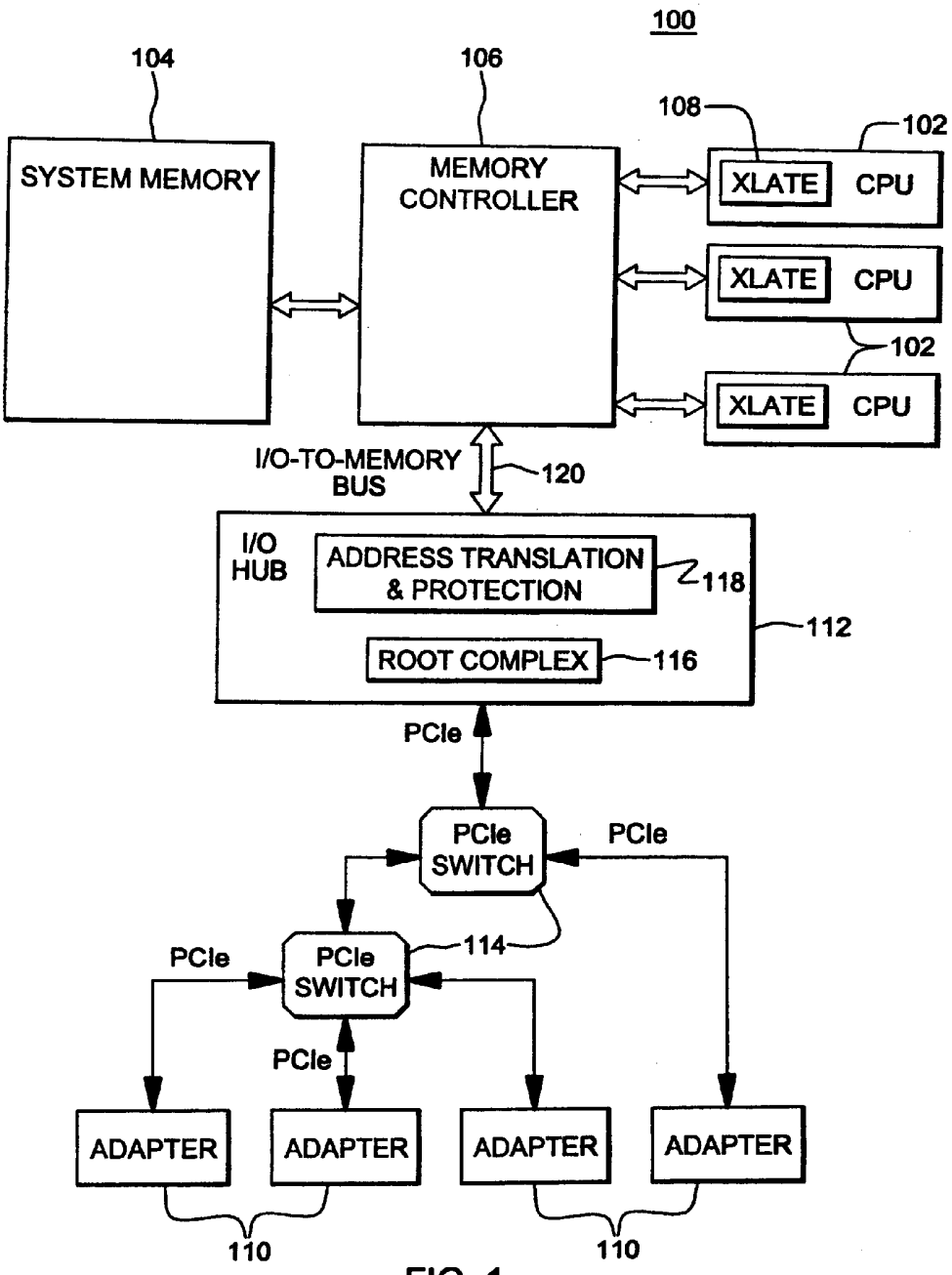

100
104
106
108
102
SYSTEM MEMORY
MEMORY CONTROLLER
XLATE
CPU
XLATE
CPU
102
XLATE
CPU
I/O-TO-MEMORY BUS
120
I/O HUB
ADDRESS TRANSLATION & PROTECTION
118
112
ROOT COMPLEX
116
PCIe
PCIe SWITCH
PCIe
PCIe
PCIe SWITCH
114
PCIe
ADAPTER
ADAPTER
ADAPTER
ADAPTER
110
110

FIG. 1

Legendas da FIG.1

104 – MEMÓRIA DE SISTEMA

106 – CONTROLADOR DE MEMÓRIA

120 – *BUS* DE I/O PARA A MEMÓRIA

112 – *HUB* DE I/O

116 – COMPLEXO DE RAÍZ

118 – TRADUÇÃO & PROTEÇÃO DE ENDEREÇOS

114 – COMUTADOR PCIe

110 – ADAPTADOR

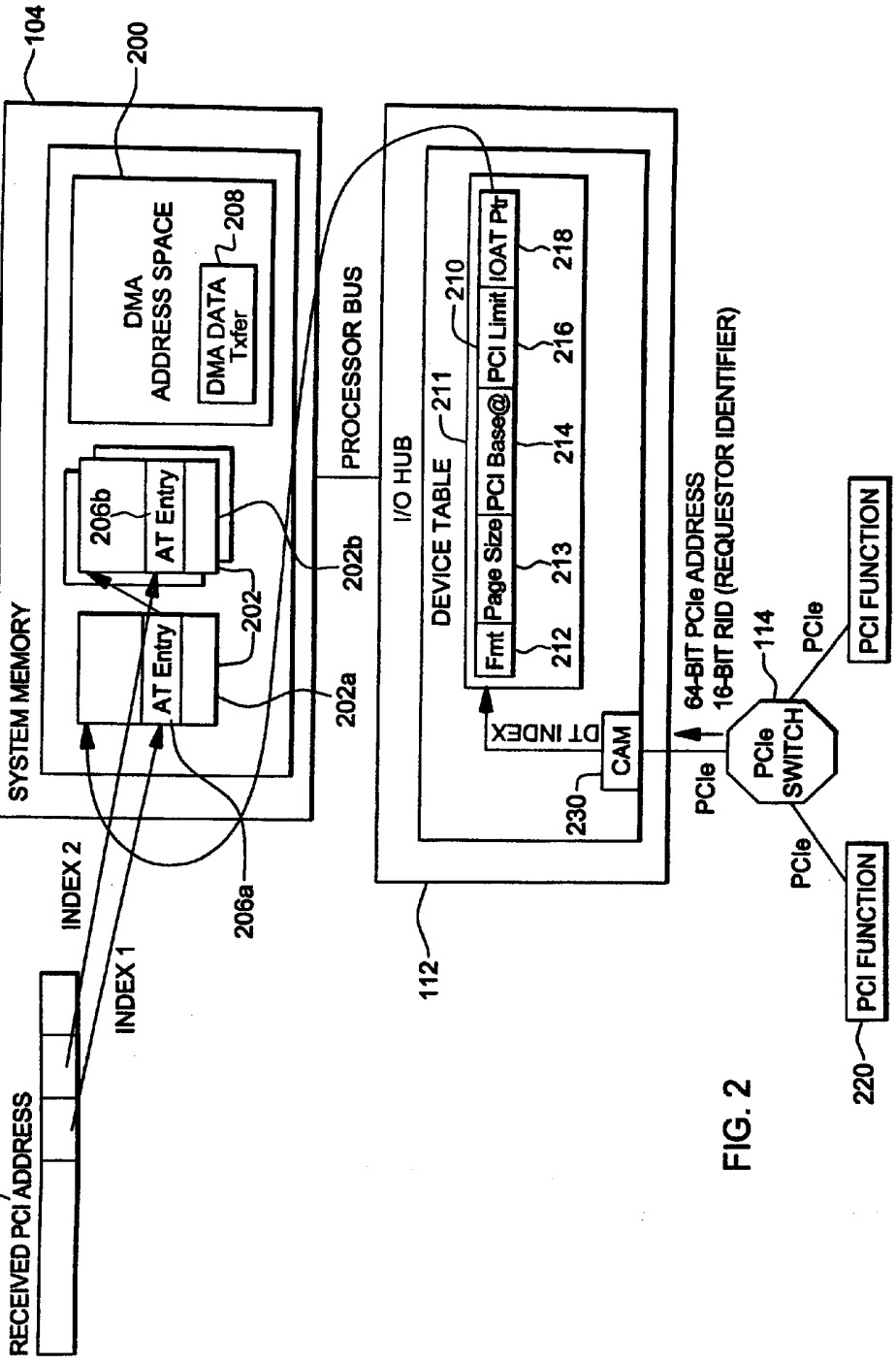

104
200
SYSTEM MEMORY
DMA ADDRESS SPACE
208
DMA DATA Txfer
206b
AT Entry
202
AT Entry
202a
206a
INDEX 2
INDEX 1
204
RECEIVED PCI ADDRESS
PROCESSOR BUS
202b
I/O HUB
DEVICE TABLE
211
210
IOAT Ptr
PCI Limit
PCI Base@
Page Size
Fmt
212
213
214
216
218
DT INDEX
CAM
230
112
64-BIT PCIe ADDRESS
16-BIT RID (REQUESTOR IDENTIFIER)
PCIe
PCIe SWITCH
114
PCIe
PCIe
PCI FUNCTION
PCI FUNCTION
220

FIG. 2

Legenda da FIG.2

204 - ENDEREÇO PCI RECEBIDO

INDEX - ÍNDICE

SYSTEM MEMORY - MEMÓRIA DE SISTEMA

AT ENTRY - ENTRADA AT

200 - ESPAÇO DE ENDEREÇAMENTO DMA

208 - TRANSF. DADOS DMA

PROCESSOR BUS - *BUS* DE PROCESSADOR

112 - *HUB* DE I/O

DEVICE TABLE - TABELA DE DISPOSITIVO

220 - FUNÇÃO PCI

114 - COMUTADOR PCIe

64-BIT PCIe ADDRESS - ENDEREÇO PCIe DE 64 *BITS*

16-BIT RID (REQUESTOR IDENTIFIER) - RID DE 16 *BITS* (IDENTIFICADOR DE SOLICITANTE)

DT INDEX - ÍNDICE DT

PAGE SIZE - TAMANHO DA PÁGINA

PCI BASE@ - END. BASE PCI

PCI LIMIT - LIMITE PCI

IOAT PTR - APONTADOR IOAT

FMT - FORMATO

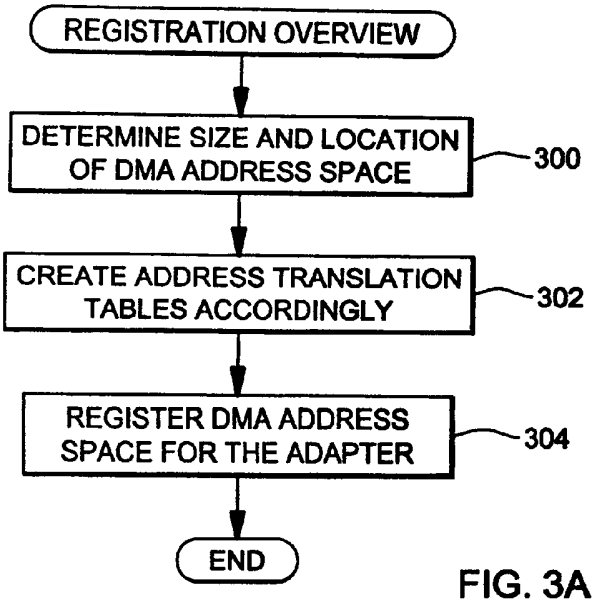

REGISTRATION OVERVIEW
DETERMINE SIZE AND LOCATION OF DMA ADDRESS SPACE
300
CREATE ADDRESS TRANSLATION TABLES ACCORDINGLY
302
REGISTER DMA ADDRESS SPACE FOR THE ADAPTER
304
END

FIG. 3A

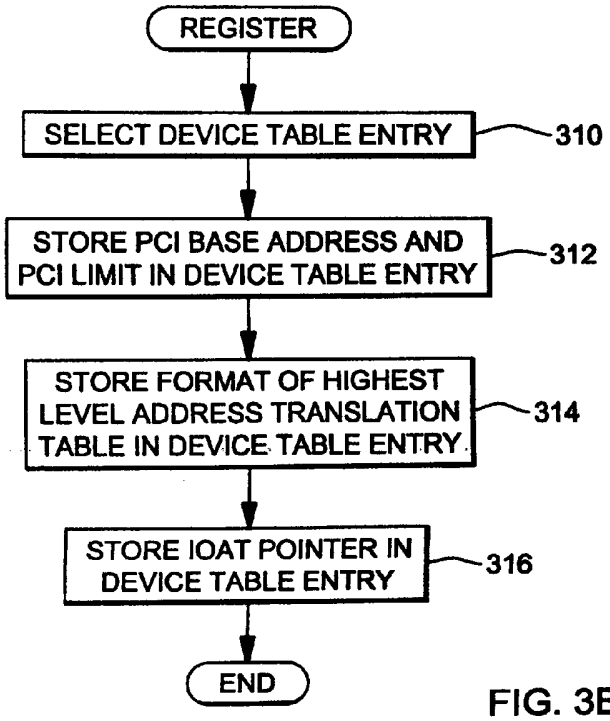

REGISTER
SELECT DEVICE TABLE ENTRY
310
STORE PCI BASE ADDRESS AND PCI LIMIT IN DEVICE TABLE ENTRY
312
STORE FORMAT OF HIGHEST LEVEL ADDRESS TRANSLATION TABLE IN DEVICE TABLE ENTRY
314
STORE IOAT POINTER IN DEVICE TABLE ENTRY
316
END

FIG. 3B

Legenda da FIG.3A

REGISTRATION OVERVIEW – VISTA GERAL DO REGISTO

300 – DETERMINAR TAMANHO E LOCALIZAÇÃO DO ESPAÇO DE ENDEREÇAMENTO DMA

302 – CRIAR TABELAS DE TRADUÇÃO DE ENDEREÇOS EM CONFORMIDADE

304 – REGISTAR ESPAÇO DE ENDEREÇAMENTO DMA PARA O ADAPTADOR

END – FIM

Legenda da FIG.3B

REGISTER – REGISTAR

310 – SELECIONAR ENTRADA DE TABELA DE DISPOSITIVO

312 – ARMAZENAR ENDEREÇO BASE PCI E LIMITE PCI NA ENTRADA DE TABELAS DE DISPOSITIVOS

314 – ARMAZENAR FORMATO DE TABELA DE TRADUÇÃO DE ENDEREÇOS DE NÍVEL MAIS ELEVADO NA ENTRADA DE TABELA DE DISPOSITIVO

316 – ARMAZENAR APONTADOR IOAT NA ENTRADA DE TABELA DE DISPOSITIVO

END – FIM

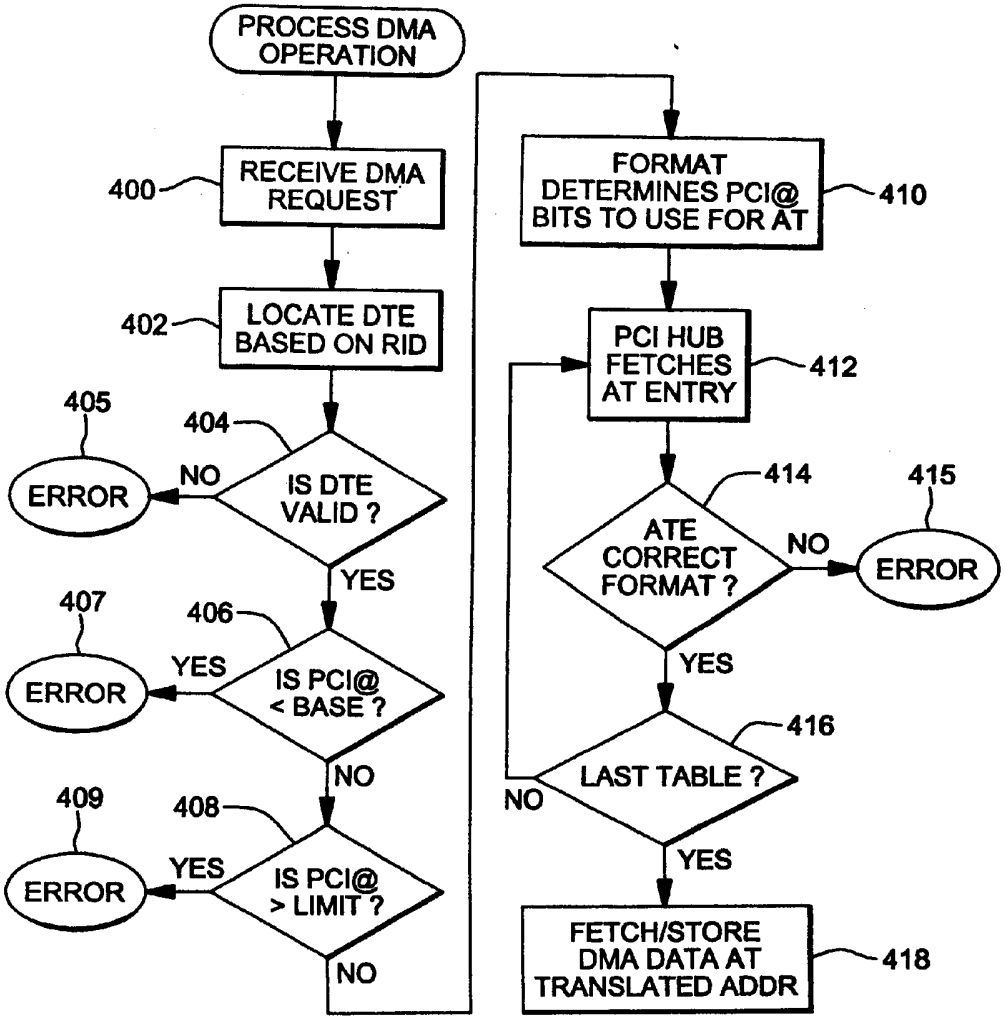

PROCESS DMA OPERATION
400
RECEIVE DMA REQUEST
402
LOCATE DTE BASED ON RID
404
IS DTE VALID ?
NO
405
ERROR
YES
406
IS PCI@ < BASE ?
YES
407
ERROR
NO
408
IS PCI@ > LIMIT ?
YES
409
ERROR
NO
410
FORMAT DETERMINES PCI@ BITS TO USE FOR AT
412
PCI HUB FETCHES AT ENTRY
414
ATE CORRECT FORMAT ?
NO
415
ERROR
YES
416
LAST TABLE ?
NO
YES
418
FETCH/STORE DMA DATA AT TRANSLATED ADDR

FIG. 4

Legenda da FIG.4

PROCESS DMA OPERATION – PROCESSAR OPERAÇÃO DMA

400 – RECEBER PEDIDO DMA

402 – LOCALIZAR DTE COM BASE EM RID

404 – DTE É VÁLIDO?

NO – NÃO

YES – SIM

ERROR – ERRO

406 – O ENDEREÇO PCI < BASE?

408 – O ENDEREÇO PCI > LIMITE?

410 – FORMATO DETERMINA QUE BITS DE ENDEREÇO PCI UTILIZAR PARA AT

412 – *HUB* PCI PESQUISA ENTRADA DE AT

414 – FORMATO CORRETO DE ATE?

416 – ÚLTIMA TABELA?

418 – PESQUISAR/ARMAZENAR DADOS DMA NO ENDEREÇO TRADUZIDO

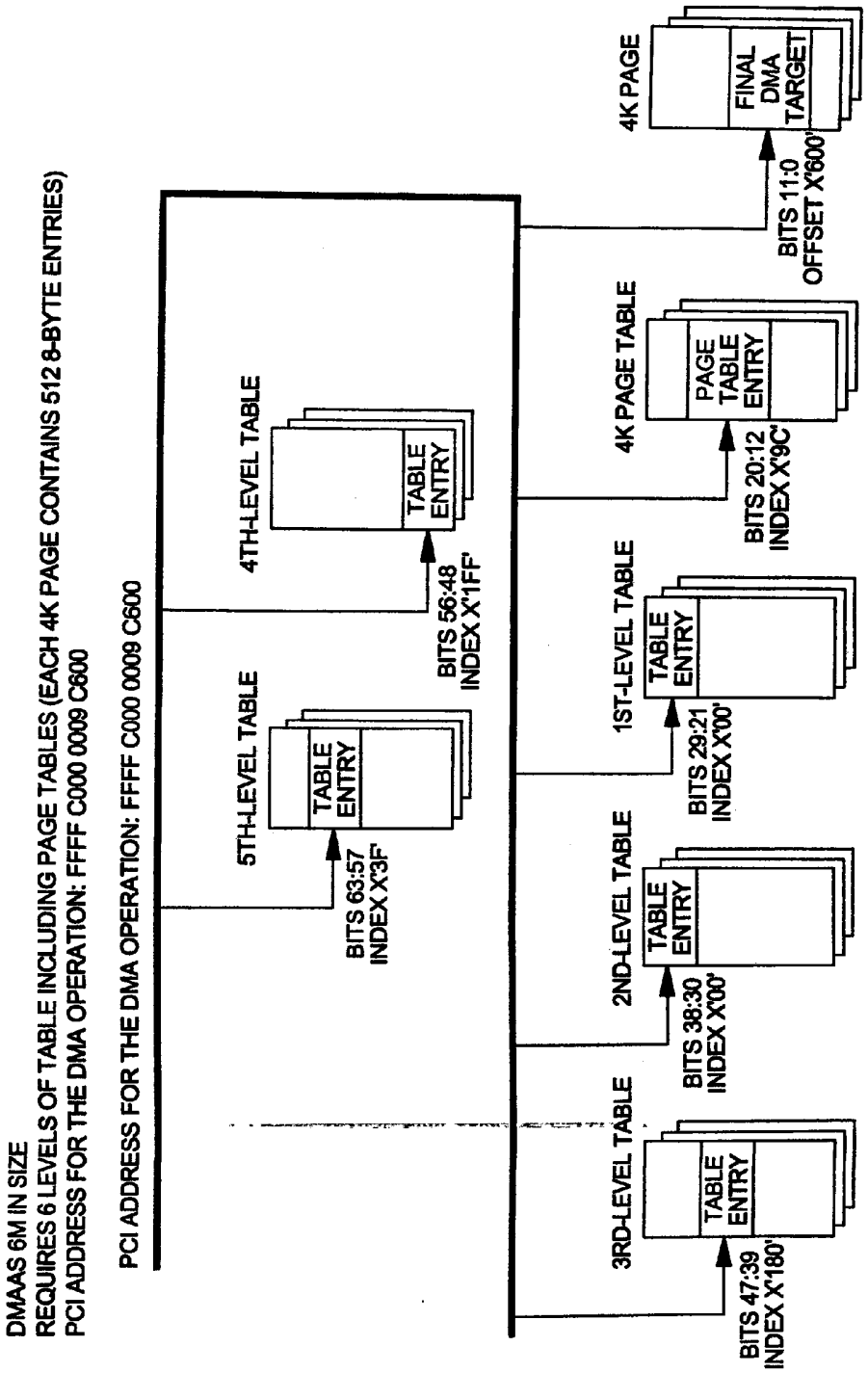

DMAAS 6M IN SIZE
REQUIRES 6 LEVELS OF TABLE INCLUDING PAGE TABLES (EACH 4K PAGE CONTAINS 512 8-BYTE ENTRIES)
PCI ADDRESS FOR THE DMA OPERATION: FFFF C000 0009 C600
PCI ADDRESS FOR THE DMA OPERATION: FFFF C000 0009 C600
5TH-LEVEL TABLE
TABLE ENTRY
BITS 63:57 INDEX X'3F'
4TH-LEVEL TABLE
TABLE ENTRY
BITS 56:48 INDEX X'1FF'
3RD-LEVEL TABLE
TABLE ENTRY
BITS 47:39 INDEX X'180'
2ND-LEVEL TABLE
TABLE ENTRY
BITS 38:30 INDEX X'00'
1ST-LEVEL TABLE
TABLE ENTRY
BITS 29:21 INDEX X'00'
4K PAGE TABLE
PAGE TABLE ENTRY
BITS 20:12 INDEX X'9C'
4K PAGE
FINAL DMA TARGET
BITS 11:0 OFFSET X'600'

FIG. 5A

Legenda da FIG.5A

DMAAS 6M EM TAMANHO

REQUER 6 NÍVEIS DE TABELA INCLUINDO TABELAS DE PÁGINA (CADA PÁGINA DE 4K CONTÉM 512 ENTRADAS DE 8 *BYTES*)

ENDEREÇO PCI PARA A OPERAÇÃO DMA: FFFF C000 0009 C600

INDEX – ÍNDICE

5TH LEVEL TABLE – TABELA DE NÍVEL 5

TABLE ENTRY – ENTRADA DE TABELA

4TH LEVEL TABLE – TABELA DE NÍVEL 4

3RD LEVEL TABLE – TABELA DE NÍVEL 3

2ND LEVEL TABLE – TABELA DE NÍVEL 2

1ST LEVEL TABLE – TABELA DE NÍVEL 1

4K PAGE TABLE – TABELA DE PÁGINAS DE 4K

4K PAGE – PÁGINA DE 4K

FINAL DMA TARGET – ALVO DMA FINAL

OFFSET – DESLOCAMENTO

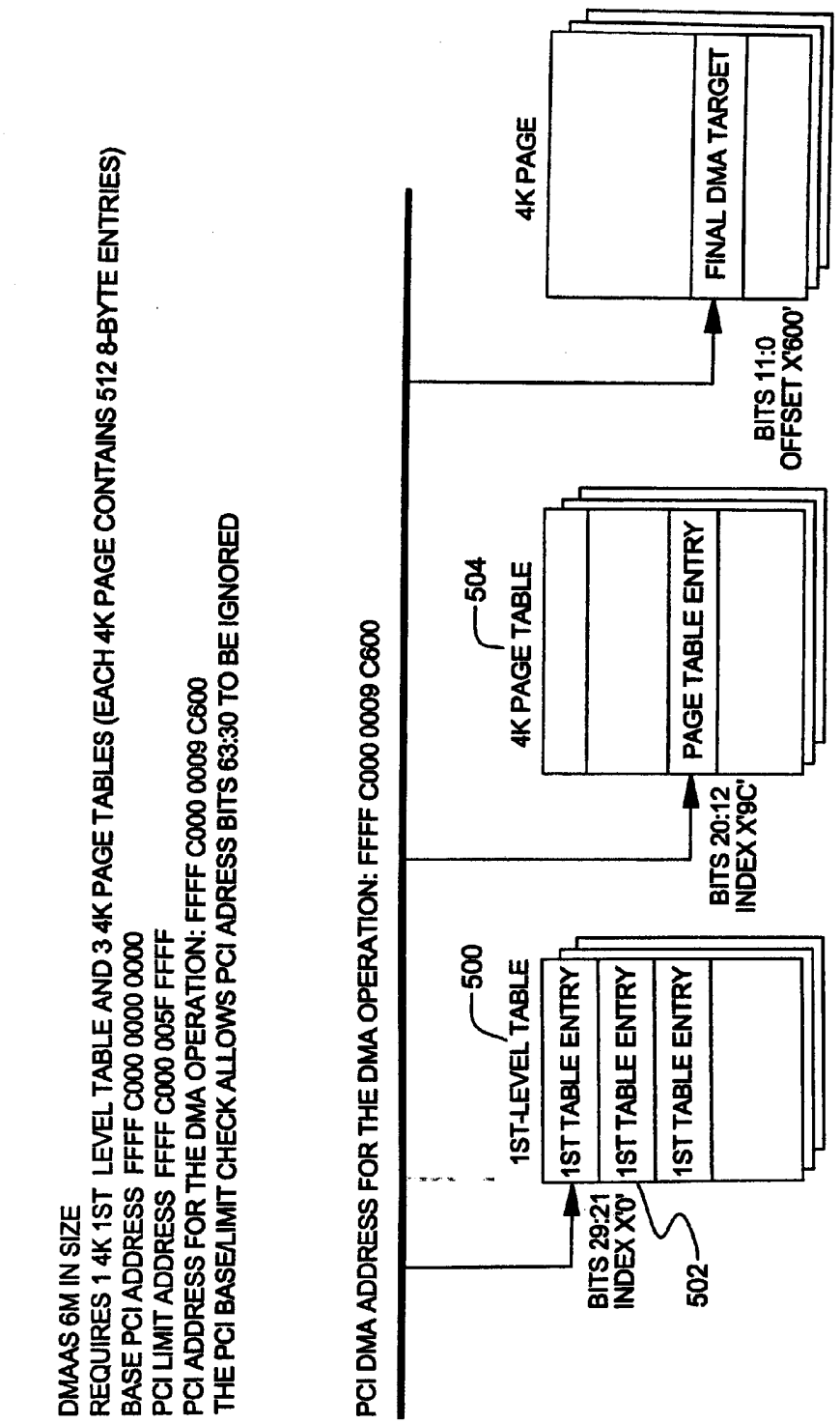

DMAAS 6M IN SIZE
REQUIRES 1 4K 1ST LEVEL TABLE AND 3 4K PAGE TABLES (EACH 4K PAGE CONTAINS 512 8-BYTE ENTRIES)
BASE PCI ADDRESS FFFF C000 0000 0000
PCI LIMIT ADDRESS FFFF C000 005F FFFF
PCI ADDRESS FOR THE DMA OPERATION: FFFF C000 0009 C600
THE PCI BASE/LIMIT CHECK ALLOWS PCI ADRESS BITS 63:30 TO BE IGNORED
PCI DMA ADDRESS FOR THE DMA OPERATION: FFFF C000 0009 C600
500
1ST-LEVEL TABLE
1ST TABLE ENTRY
1ST TABLE ENTRY
1ST TABLE ENTRY
BITS 29:21
INDEX X'0'
502
504
4K PAGE TABLE
PAGE TABLE ENTRY
BITS 20:12
INDEX X'9C'
4K PAGE
FINAL DMA TARGET
BITS 11:0
OFFSET X'600'

FIG. 5B

Legenda da FIG.5B

DMAAS 6M EM TAMANHO

REQUER 1 TABELA DE NÍVEL 1 DE 4K E 3 TABELAS DE PÁGINAS DE 4K (CADA PÁGINA DE 4K CONTÉM 512 ENTRADAS DE 8 *BYTES*)

ENDEREÇO BASE PCI: FFFF C000 0000 0000

ENDEREÇO LIMITE PCI: FFFF C000 005F FFFF

ENDEREÇO PCI PARA A OPERAÇÃO DMA: FFFF C000 0009 C600

A VERIFICAÇÃO BASE/LIMITE PCI PERMITE QUE OS *BITS* DE ENDEREÇO PCI 63:30 SEJAM IGNORADOS

1ST TABLE ENTRY – ENTRADA NA 1ª TABELA

4K PAGE TABLE – TABELA DE PÁGINAS DE 4K

4K PAGE – PÁGINA DE 4K

FINAL DMA TARGET – ALVO DMA FINAL

OFFSET – DESLOCAMENTO

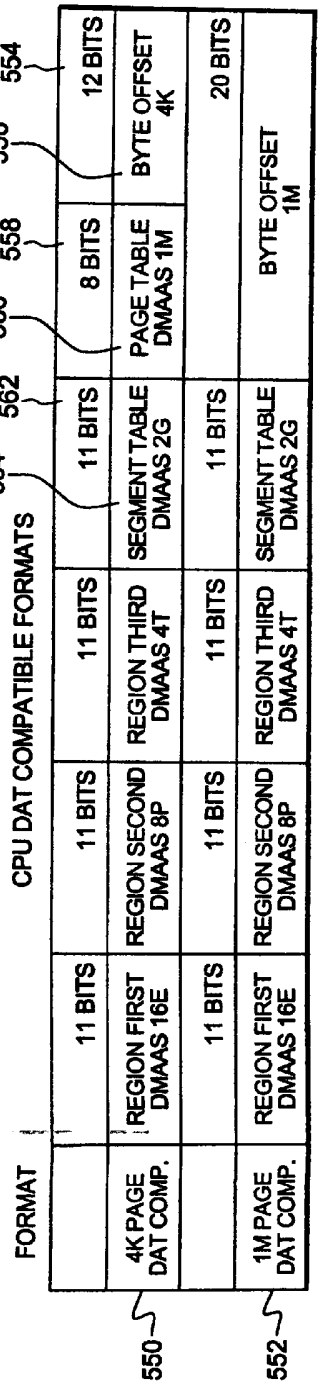

CPU DAT COMPATIBLE FORMATS
FORMAT
11 BITS
11 BITS
11 BITS
11 BITS
8 BITS
12 BITS
4K PAGE DAT COMP.
REGION FIRST DMAAS 16E
REGION SECOND DMAAS 8P
REGION THIRD DMAAS 4T
SEGMENT TABLE DMAAS 2G
PAGE TABLE DMAAS 1M
BYTE OFFSET 4K
11 BITS
11 BITS
11 BITS
11 BITS
20 BITS
1M PAGE DAT COMP.
REGION FIRST DMAAS 16E
REGION SECOND DMAAS 8P
REGION THIRD DMAAS 4T
SEGMENT TABLE DMAAS 2G
BYTE OFFSET 1M
550
552
554
556
558
560
562
564

FIG. 5C

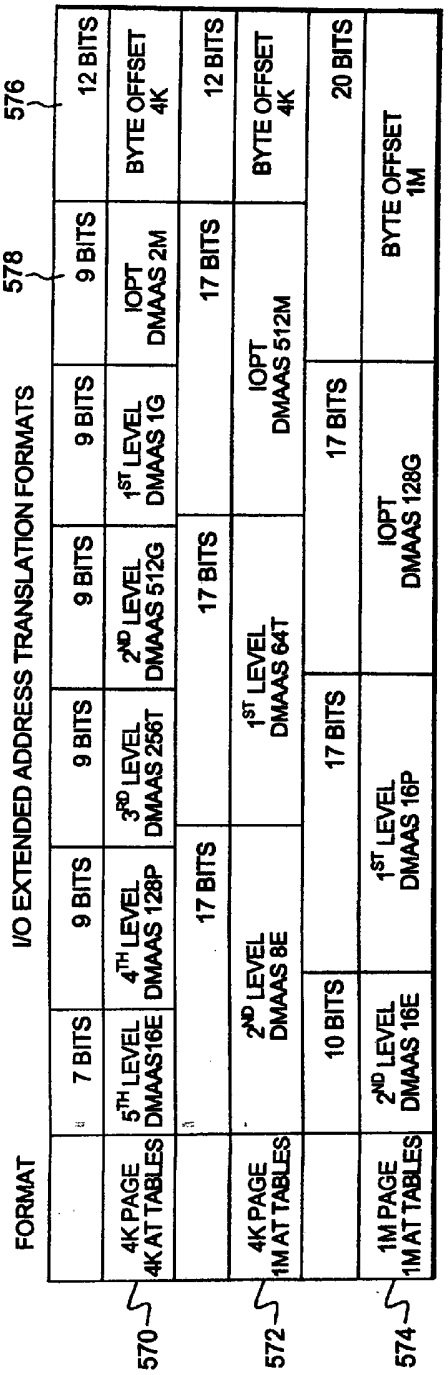

I/O EXTENDED ADDRESS TRANSLATION FORMATS
FORMAT
7 BITS
9 BITS
9 BITS
9 BITS
9 BITS
12 BITS
4K PAGE 4K AT TABLES
5TH LEVEL DMAAS16E
4TH LEVEL DMAAS 128P
3RD LEVEL DMAAS 256T
2ND LEVEL DMAAS 512G
1ST LEVEL DMAAS 1G
IOPT DMAAS 2M
BYTE OFFSET 4K
17 BITS
17 BITS
12 BITS
4K PAGE 1M AT TABLES
2ND LEVEL DMAAS 8E
1ST LEVEL DMAAS 64T
IOPT DMAAS 512M
BYTE OFFSET 4K
10 BITS
17 BITS
17 BITS
20 BITS
1M PAGE 1M AT TABLES
2ND LEVEL DMAAS 16E
1ST LEVEL DMAAS 16P
IOPT DMAAS 128G
BYTE OFFSET 1M
570
572
574
576
578

FIG. 5D

Legenda da FIG.5C

FORMAT – FORMATO

CPU DAT COMPATIBLE FORMATS – FORMATOS COMPATÍVEIS COM CPU DAT

REGION FIRST – PRIMEIRA REGIÃO

REGION SECOND – SEGUNDA REGIÃO

REGION THIRD – TERCEIRA REGIÃO

SEGMENT TABLE – TABELA DE SEGMENTO

PAGE TABLE – TABELA DE PÁGINA

BYTE OFFSET – DESLOCAMENTO DE *BYTE*

1M PAGE DAT COMP. – PÁGINA DE 1M COMPATÍVEL COM DAT

Legenda da FIG.5D

I/O EXTENDED ADDRESS TRANSLATION FORMATS – FORMATOS DE TRADUÇÃO DE ENDEREÇOS ESTENDIDOS DE I/O

FORMAT – FORMATO

4K PAGE 4K AT TABLES – TABELAS AT DE 4K, PÁGINAS DE 4K

4K PAGE 1M AT TABLES – TABELAS AT DE 1M, PÁGINAS DE 4K

1M PAGE 1M AT TABLES – TABELAS AT DE 1M, PÁGINAS DE 1M

1ST LEVEL – NÍVEL 1

2ND LEVEL – NÍVEL 2

3RD LEVEL – NÍVEL 3

4TH LEVEL – NÍVEL 4

BYTE OFFSET – DESLOCAMENTO DE *BYTE*

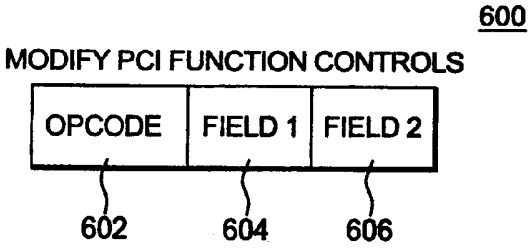



FIG. 6A

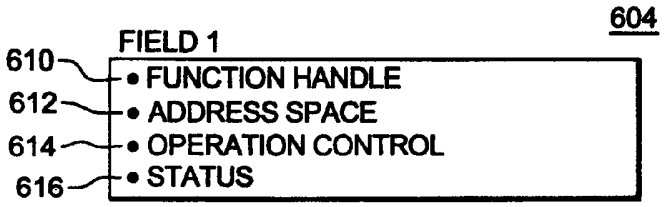



FIG. 6B

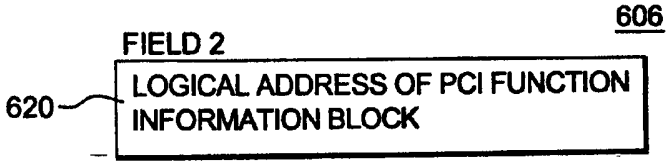



FIG. 6C

Legenda da FIG. 6A

MODIFY PCI FUNCTION CONTROLS - MODIFICAR CONTROLOS DE FUNÇÕES PCI

OPCODE - CÓDIGO OPERACIONAL

FIELD - CAMPO

Legenda da FIG.6B

610 - *HANDLE* DE FUNÇÃO

612 - ESPAÇO DE ENDEREÇAMENTO

614 - CONTROLO OPERACIONAL

616 - ESTADO

Legenda daFIG.6C

FIELD - CAMPO

620 - ENDEREÇO LÓGICO DE BLOCO DE INFORMAÇÃO DE FUNÇÃO PCI

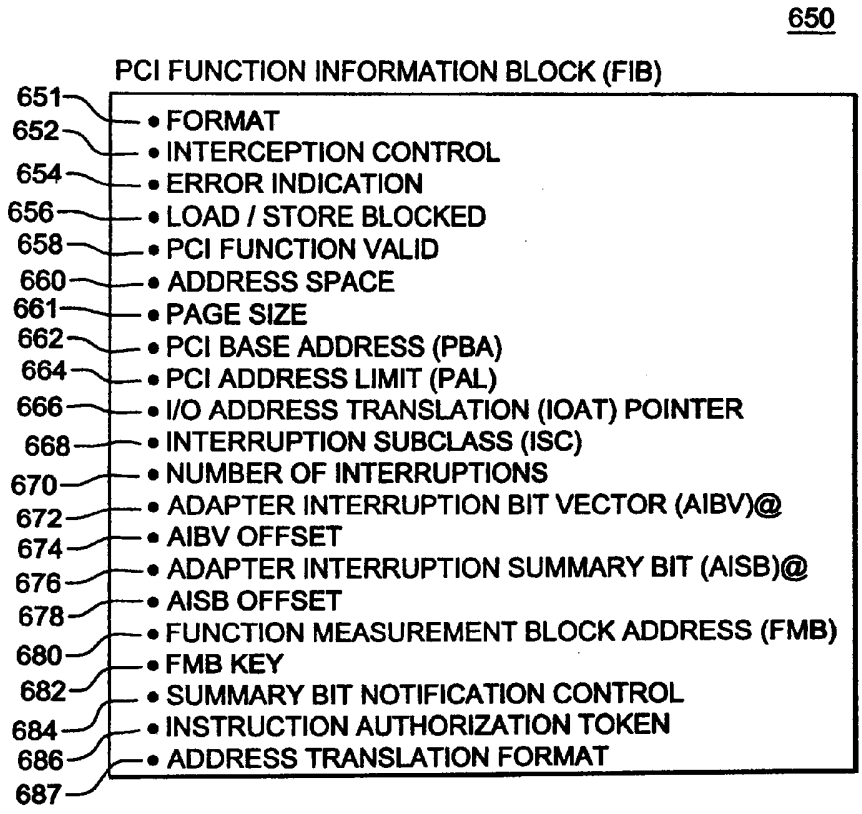

650
PCI FUNCTION INFORMATION BLOCK (FIB)
651
652
654
656
658
660
661
662
664
666
668
670
672
674
676
678
680
682
684
686
687
• FORMAT
• INTERCEPTION CONTROL
• ERROR INDICATION
• LOAD / STORE BLOCKED
• PCI FUNCTION VALID
• ADDRESS SPACE
• PAGE SIZE
• PCI BASE ADDRESS (PBA)
• PCI ADDRESS LIMIT (PAL)
• I/O ADDRESS TRANSLATION (IOAT) POINTER
• INTERRUPTION SUBCLASS (ISC)
• NUMBER OF INTERRUPTIONS
• ADAPTER INTERRUPTION BIT VECTOR (AIBV)@
• AIBV OFFSET
• ADAPTER INTERRUPTION SUMMARY BIT (AISB)@
• AISB OFFSET
• FUNCTION MEASUREMENT BLOCK ADDRESS (FMB)
• FMB KEY
• SUMMARY BIT NOTIFICATION CONTROL
• INSTRUCTION AUTHORIZATION TOKEN
• ADDRESS TRANSLATION FORMAT

FIG. 6D

Legendas da FIG.6D

PCI FUNCTION INFORMATION BLOCK (FIB) - BLOCO DE INFORMAÇÃO DE FUNÇÃO PCI (FIB)

651 - FORMATO

652 - CONTROLO DE INTERCEÇÃO

654 - INDICAÇÃO DE ERRO

656 - CARREGAMENTO/ARMAZENAMENTO BLOQUEADO

658 - FUNÇÃO PCI VÁLIDA

660 - ESPAÇO DE ENDEREÇAMENTO

661 - TAMANHO DE PÁGINA

662 - ENDEREÇO BASE PCI (PBA)

664 - LIMITE DE ENDEREÇO PCI (PAL)

666 - APONTADOR DE TRADUÇÃO DE ENDEREÇOS DE I/O (IOAT)

668 - SUBCLASSE DE INTERRUPÇÃO (ISC)

670 - NÚMERO DE INTERRUPÇÕES

672 - VETOR DE *BITS* DE INTERRUPÇÃO DE ADAPTADOR (AIBV)@

674 - DESLOCAMENTO DE AIBV

676 - *BIT* DE SUMARIZAÇÃO DE INTERRUPÇÃO DO ADAPTADOR (AISB)@

678 - DESLOCAMENTO DE AISB

680 - ENDEREÇO DE BLOCO DE MEDIÇÃO DE FUNÇÃO (FMB)

682 - CHAVE DE FMB

684 - CONTROLO DE NOTIFICAÇÃO DE *BIT* DE SUMARIZAÇÃO

686 - *TOKEN* DE AUTORIZAÇÃO DE INSTRUÇÕES

687 - FORMATO DE TRADUÇÃO DE ENDEREÇOS

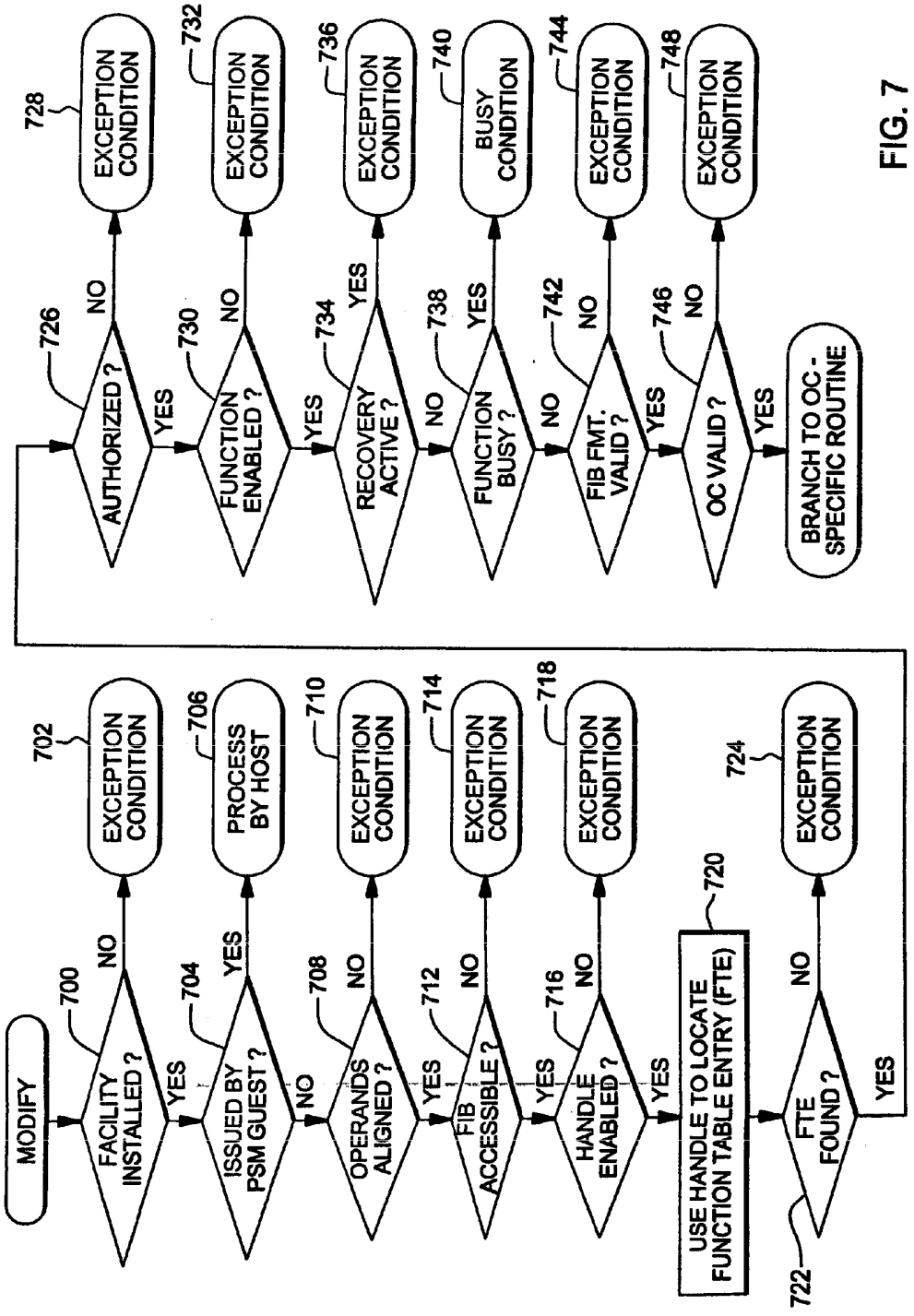

MODIFY
700
FACILITY INSTALLED ?
NO
YES
702
EXCEPTION CONDITION
704
ISSUED BY PSM GUEST ?
YES
NO
706
PROCESS BY HOST
708
OPERANDS ALIGNED ?
NO
YES
710
EXCEPTION CONDITION
712
FIB ACCESSIBLE ?
NO
YES
714
EXCEPTION CONDITION
716
HANDLE ENABLED ?
NO
YES
718
EXCEPTION CONDITION
720
USE HANDLE TO LOCATE FUNCTION TABLE ENTRY (FTE)
722
FTE FOUND ?
NO
YES
724
EXCEPTION CONDITION
726
AUTHORIZED ?
NO
YES
728
EXCEPTION CONDITION
730
FUNCTION ENABLED ?
NO
YES
732
EXCEPTION CONDITION
734
RECOVERY ACTIVE ?
YES
NO
736
EXCEPTION CONDITION
738
FUNCTION BUSY ?
YES
NO
740
BUSY CONDITION
742
FIB FMT. VALID ?
NO
YES
744
EXCEPTION CONDITION
746
OC VALID ?
NO
YES
748
EXCEPTION CONDITION
BRANCH TO OC - SPECIFIC ROUTINE

FIG. 7

Legenda da FIG.7

MODIFY – MODIFICAR

700 – CONDIÇÃO INSTALADA?

NO – NÃO

YES – SIM

702 – CONDIÇÃO DE EXCEÇÃO

704 – EMITIDO POR *GUEST* PSM?

706 – PROCESSAR POR *HOST*

708 – OPERANDOS ALINHADOS?

712 – FIB ACESSÍVEL?

716 – *HANDLE* ATIVADA?

720 – UTILIZAR HANDLE PARA LOCALIZAR ENTRADA DE TABELA DE FUNÇÃO (FTE)

722 – FTE ENCONTRADA?

726 – AUTORIZADA?

730 – FUNÇÃO ATIVADA?

734 – RECUPERAÇÃO ATIVA?

738 – FUNÇÃO OCUPADA?

742 – FORMATO FIB VÁLIDO?

746 – OC VÁLIDO?

BRANCH TO OC-SPECIFIC ROUTINE – RAMIFICAR PARA ROTINA ESPECÍFICA DE OC

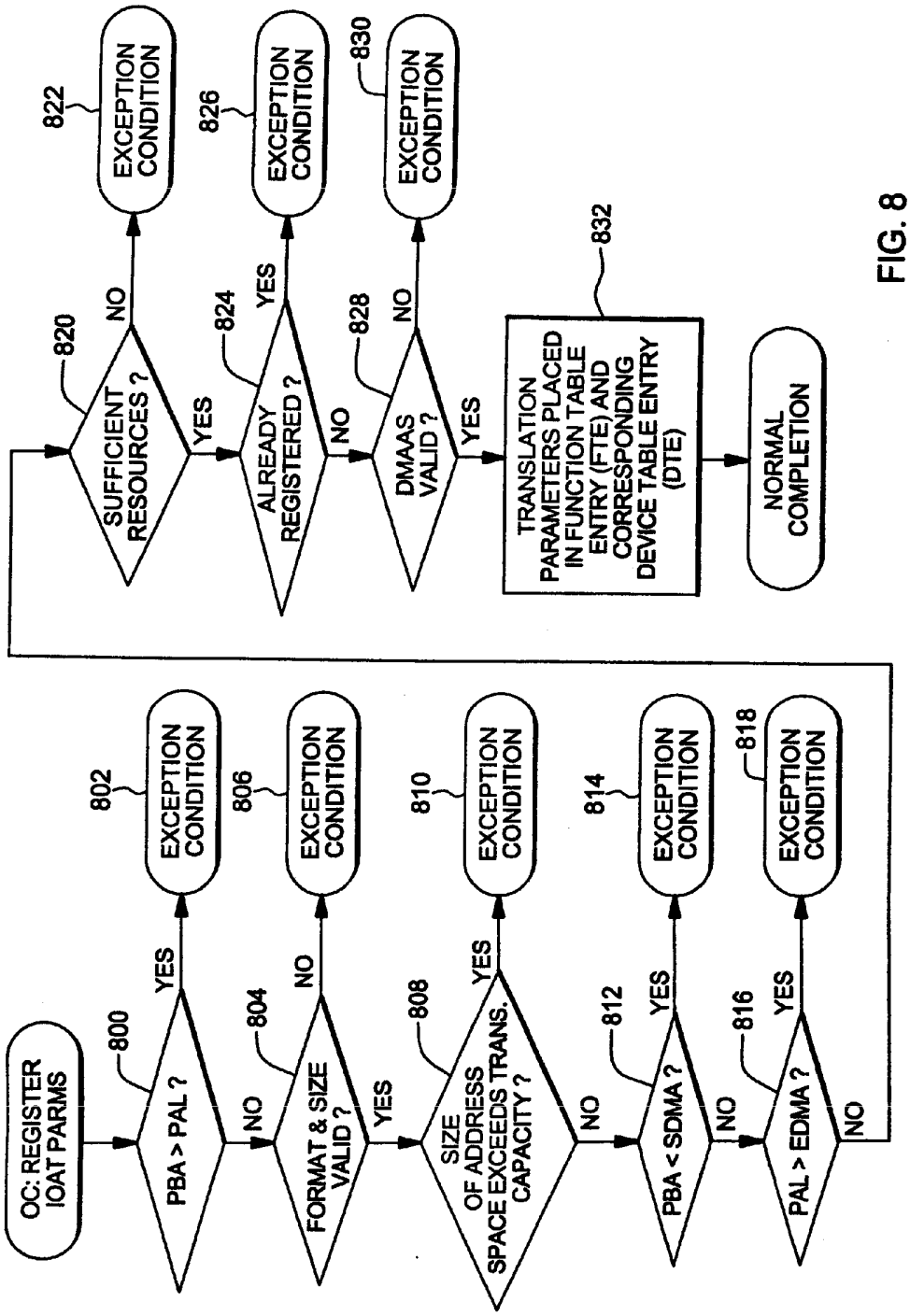

OC: REGISTER IOAT PARMS
800
PBA > PAL ?
YES
802
EXCEPTION CONDITION
NO
804
FORMAT & SIZE VALID ?
NO
806
EXCEPTION CONDITION
YES
808
SIZE OF ADDRESS SPACE EXCEEDS TRANS. CAPACITY ?
YES
810
EXCEPTION CONDITION
NO
812
PBA < SDMA ?
YES
814
EXCEPTION CONDITION
NO
816
PAL > EDMA ?
YES
818
EXCEPTION CONDITION
NO
820
SUFFICIENT RESOURCES ?
NO
822
EXCEPTION CONDITION
YES
824
ALREADY REGISTERED ?
YES
826
EXCEPTION CONDITION
NO
828
DMAAS VALID ?
NO
830
EXCEPTION CONDITION
YES
832
TRANSLATION PARAMETERS PLACED IN FUNCTION TABLE ENTRY (FTE) AND CORRESPONDING DEVICE TABLE ENTRY (DTE)
NORMAL COMPLETION

FIG. 8

Legenda da FIG.8

OC:REGISTER IOAT PARAMS - OC:REGISTAR PARÂMETROS DE OC

802 - CONDIÇÃO DE EXCEÇÃO

YES - SIM

NO -NÃO

804 - FORMATO & TAMANHO VÁLIDOS?

808 - TAMANHO DO ESPAÇO DE ENDEREÇAMENTO EXCEDE CAPACIDADE DE TRADUÇÃO?

820 - RECURSOS SUFICIENTES?

824 - JÁ REGISTADO?

828 - DMAAS VÁLIDO?

832 - PARÂMETROS DE TRADUÇÃO COLOCADOS NA ENTRADA DE TABELA DE FUNÇÃO (FTE) E ENTRADA DE TABELA DE DISPOSITIVO (DTE) CORRESPONDENTE

NORMAL COMPLETION - CONCLUSÃO NORMAL

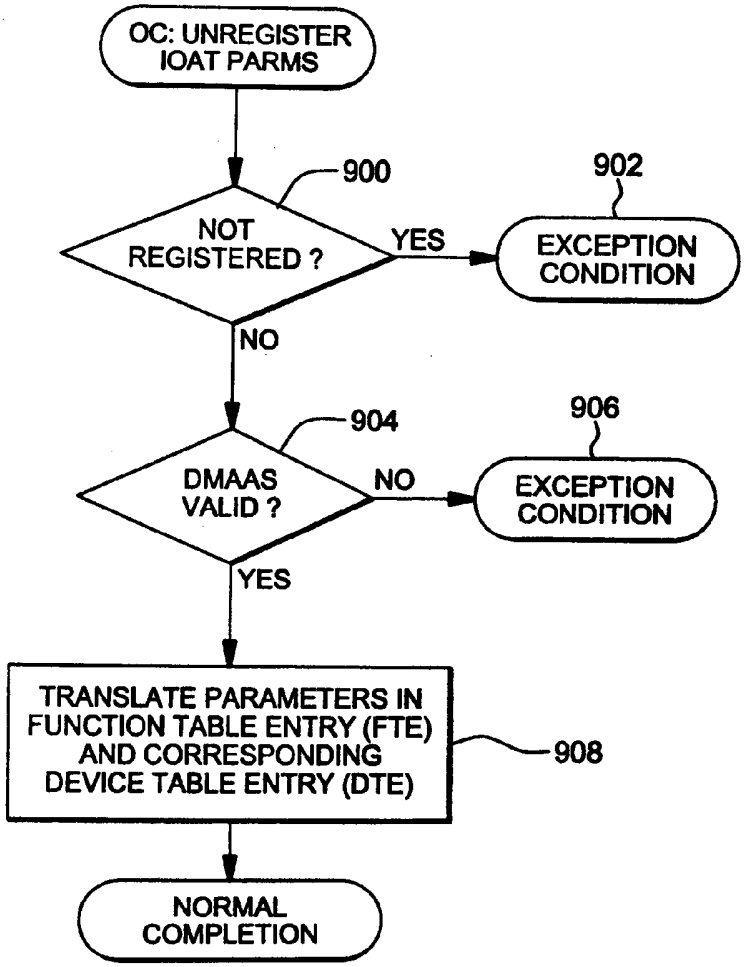

OC: UNREGISTER IOAT PARMS
900
NOT REGISTERED ?
YES
902
EXCEPTION CONDITION
NO
904
DMAAS VALID ?
NO
906
EXCEPTION CONDITION
YES
TRANSLATE PARAMETERS IN FUNCTION TABLE ENTRY (FTE) AND CORRESPONDING DEVICE TABLE ENTRY (DTE)
908
NORMAL COMPLETION

FIG. 9

Legenda da FIG.9

OC:UNREGISTER IOAT PARAMS - OC:CANCELAMENTO DE REGISTO DE PARÂMETROS IOAT

900 - NÃO REGISTADO?

YES - SIM

NO - NÃO

902 - CONDIÇÃO DE EXCEÇÃO

904 - DMAAS VÁLIDO?

908 - TRADUZIR PARÂMETROS NA ENTRADA DE TABELA DE FUNÇÃO (FTE) E ENTRADA DE TABELA DE DISPOSITIVO (DTE) CORRESPONDENTE

NORMAL COMPLETION - CONCLUSÃO NORMAL

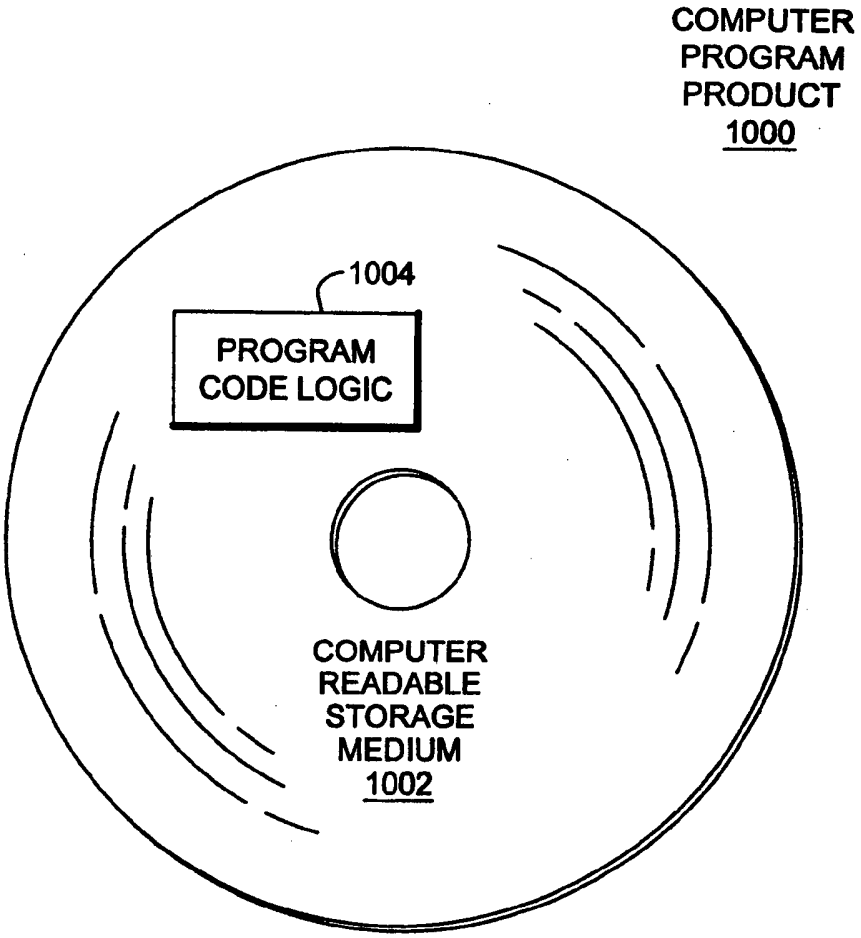



FIG. 10

Legenda da FIG.10

1000 - PRODUTO DE PROGRAMA DE COMPUTADOR

1004 - LÓGICA DE CÓDIGO DE PROGRAMA

1002 - SUPORTE DE ARMAZENAMENTO LEGÍVEL POR COMPUTADOR

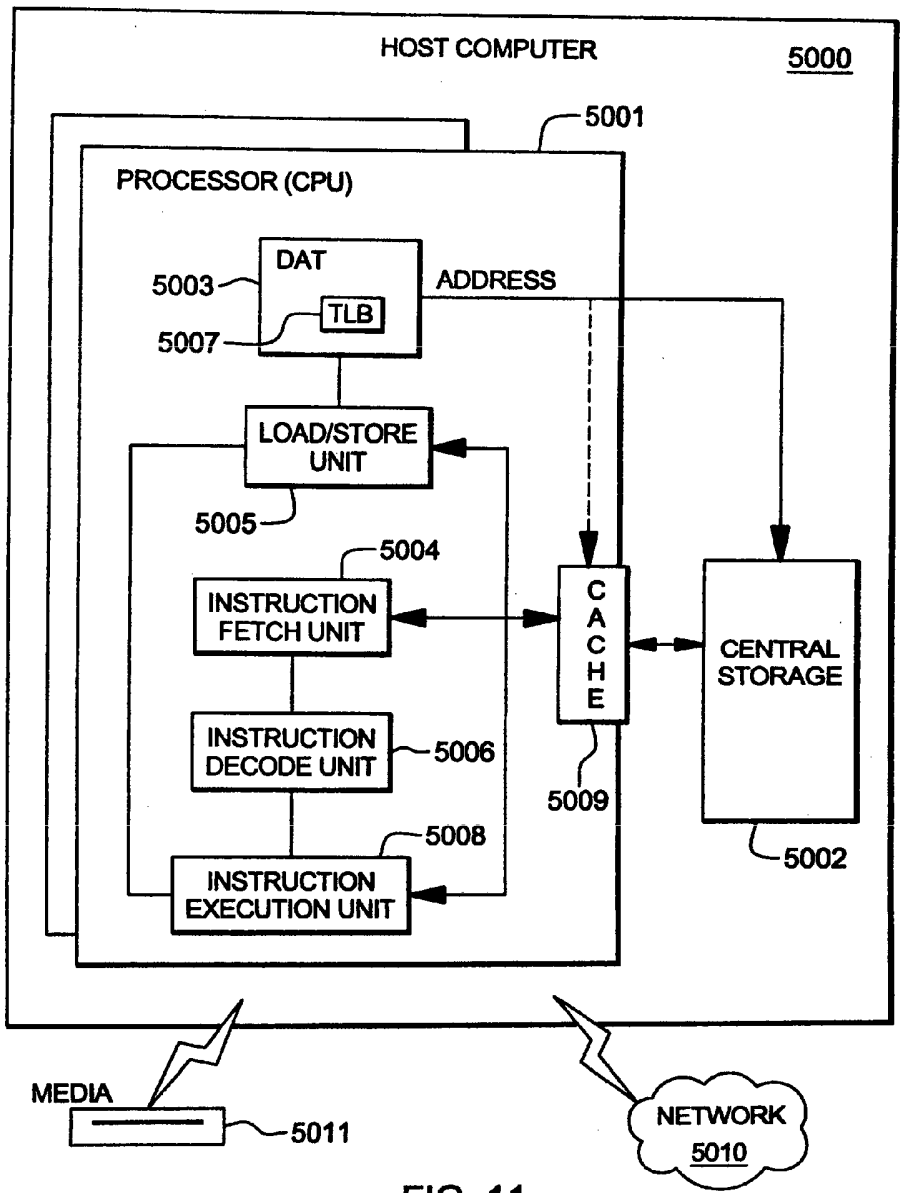

HOST COMPUTER
5000
5001
PROCESSOR (CPU)
DAT
5003
TLB
5007
ADDRESS
LOAD/STORE UNIT
5005
5004
INSTRUCTION FETCH UNIT
INSTRUCTION DECODE UNIT
5006
INSTRUCTION EXECUTION UNIT
5008
CACHE
5009
CENTRAL STORAGE
5002
MEDIA
5011
NETWORK
5010

FIG. 11

Legenda da FIG. 11

5000 – COMPUTADOR *HOST*

PROCESSOR (CPU) – PROCESSADOR (CPU)

ADDRESS – ENDEREÇO

5005 – UNIDADE DE CARGA/ARMAZENAMENTO

5004 – UNIDADE DE PESQUISA DE INSTRUÇÕES

5006 – UNIDADE DE DESCODIFICAÇÃO DE INSTRUÇÕES

5008 – UNIDADE DE EXECUÇÃO DE INSTRUÇÕES

5009 – *CACHE*

5002 – ARMAZENAMENTO CENTRAL

MEDIA – SUPORTE

5010 – REDE

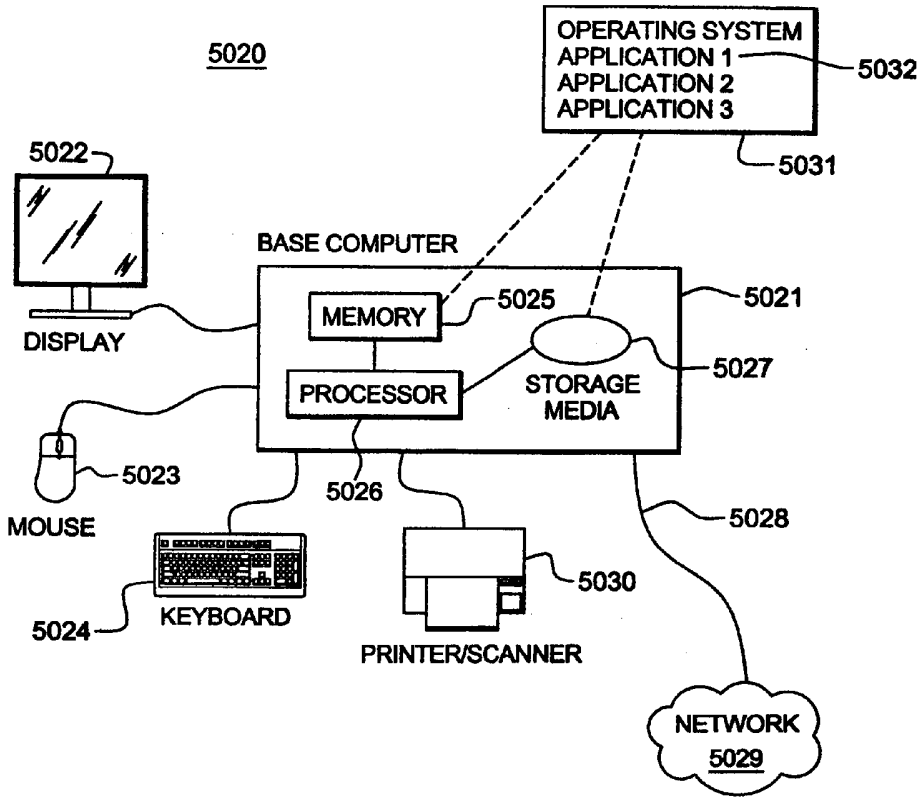



FIG. 12

Legenda da FIG.12

5022 - MONITOR

5023 - RATO

5024 - TECLADO

BASE COMPUTER - COMPUTADOR BASE

5025 - MEMÓRIA

5026 - PROCESSADOR

5027 - SUPORTE DE ARMAZENAMENTO

5030 - IMPRESSORA/*SCANNER*

5029 - REDE

OPERATING SYSTEM - SISTEMA OPERATIVO

APPLICATION - APLICAÇÃO

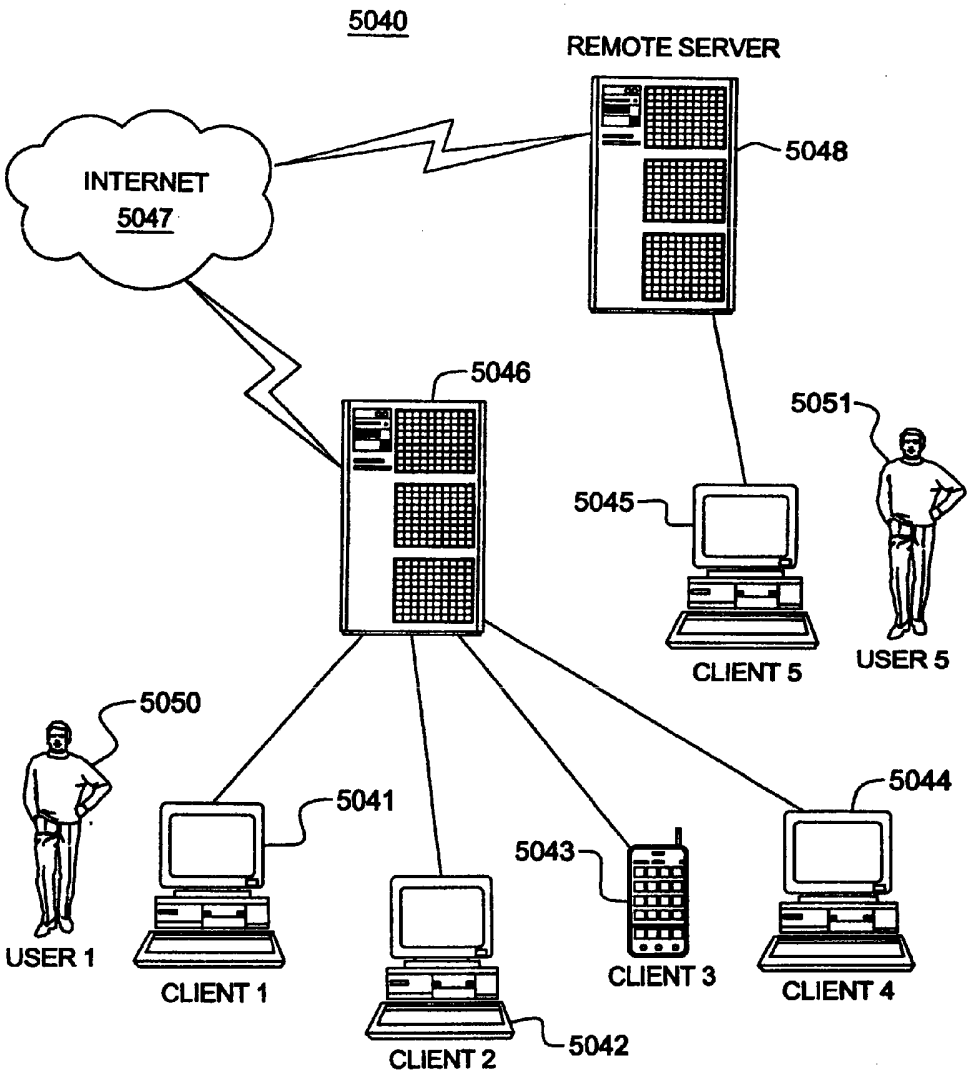



FIG. 13

Legenda da FIG.13

USER - UTILIZADOR

CLIENT - CLIENTE

REMOTE SERVER - SERVIDOR REMOTO

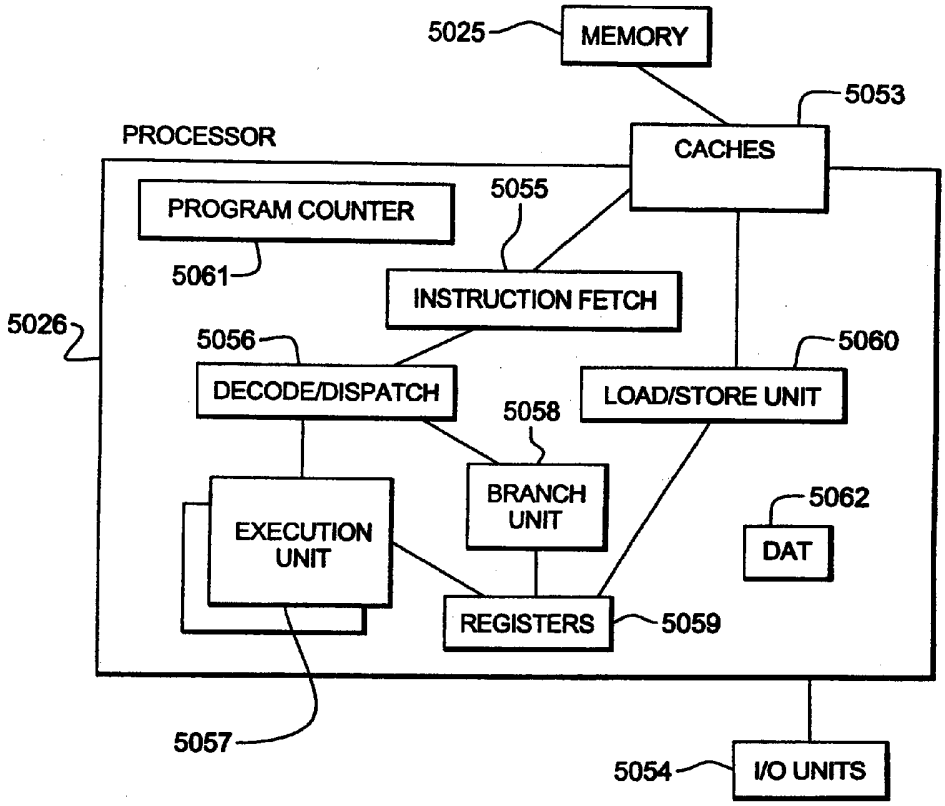



FIG. 14

Legenda da FIG.14

5025 - MEMÓRIA

PROCESSOR - PROCESSADOR

5061 - CONTADOR DE PROGRAMA

5056 - DESCODIFICAR/ENVIAR

5057 - UNIDADE DE EXECUÇÃO

5055 - PESQUISA DE INSTRUÇÕES

5058 - UNIDADE DE RAMIFICAÇÃO

5059 - REGISTOS

5060 - UNIDADE DE CARGA/ARMAZENAMENTO

5054 - UNIDADES DE I/O

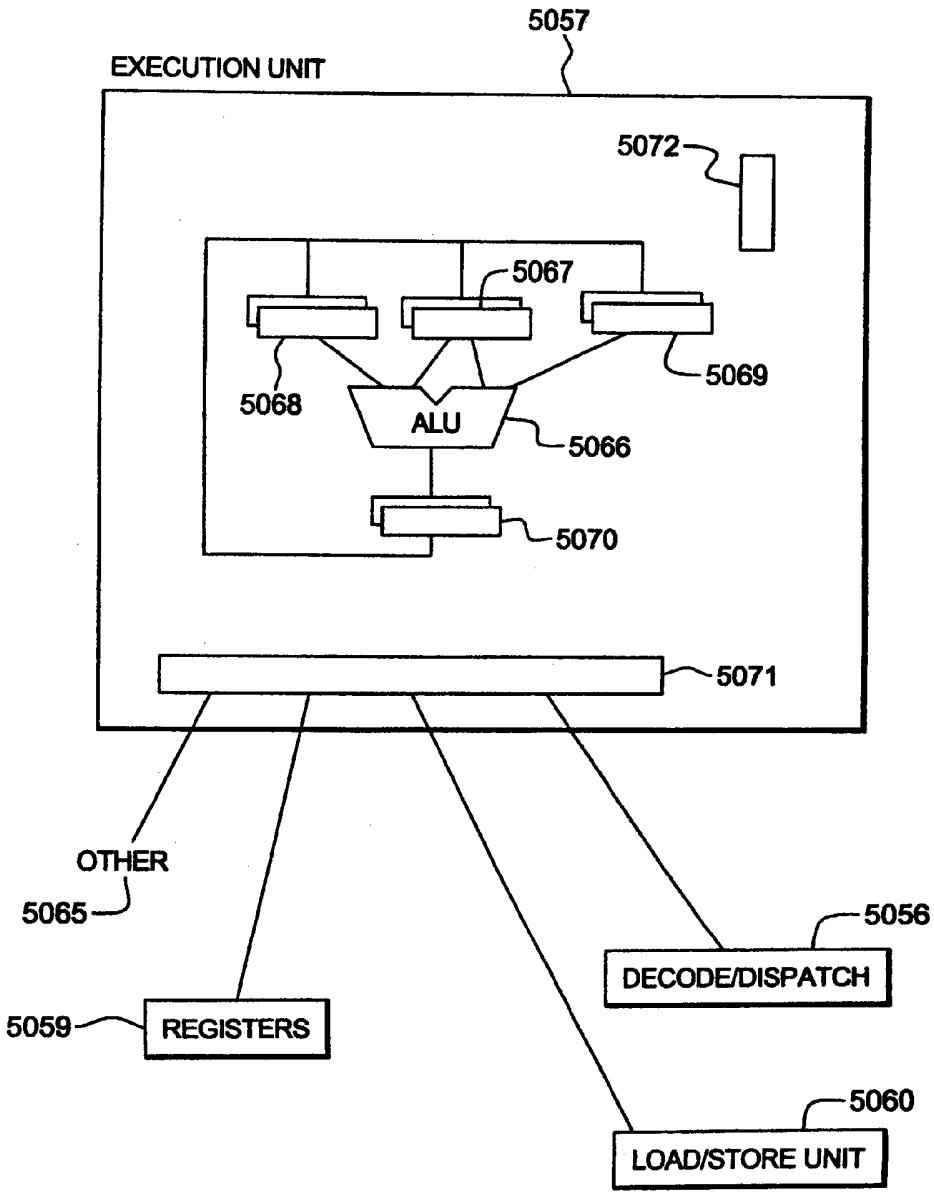



FIG. 15A

Legenda da FIG.15A

EXECUTION UNIT - UNIDADE DE EXECUÇÃO

OTHER - OUTROS

5059 - REGISTOS

5056 - DESCODIFICAR/ENVIAR

5060 - UNIDADE DE CARGA/ARMAZENAMENTO

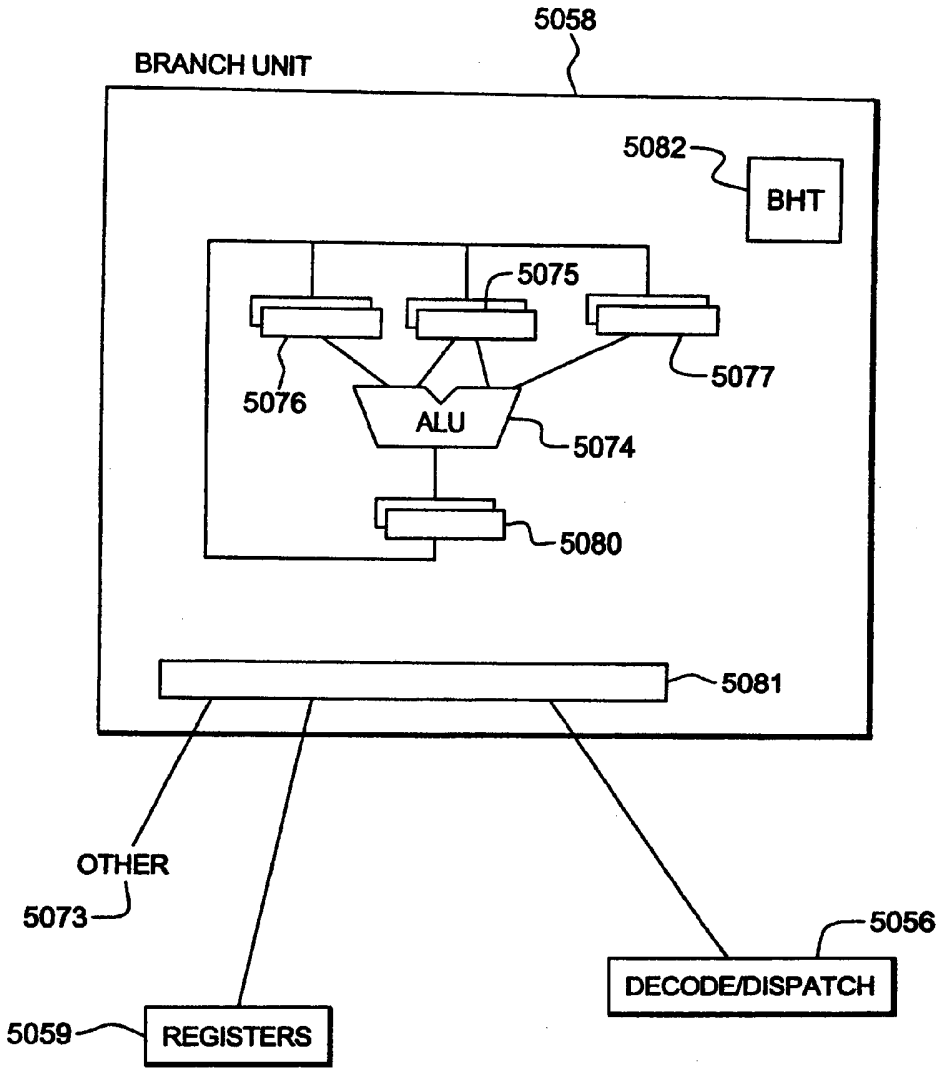



FIG. 15B

Legenda da FIG.15B

BRANCH UNIT - UNIDADE DE RAMIFICAÇÃO

OTHER - OUTROS

5059 - REGISTOS

5056 - DESCODIFICAR/ENVIAR

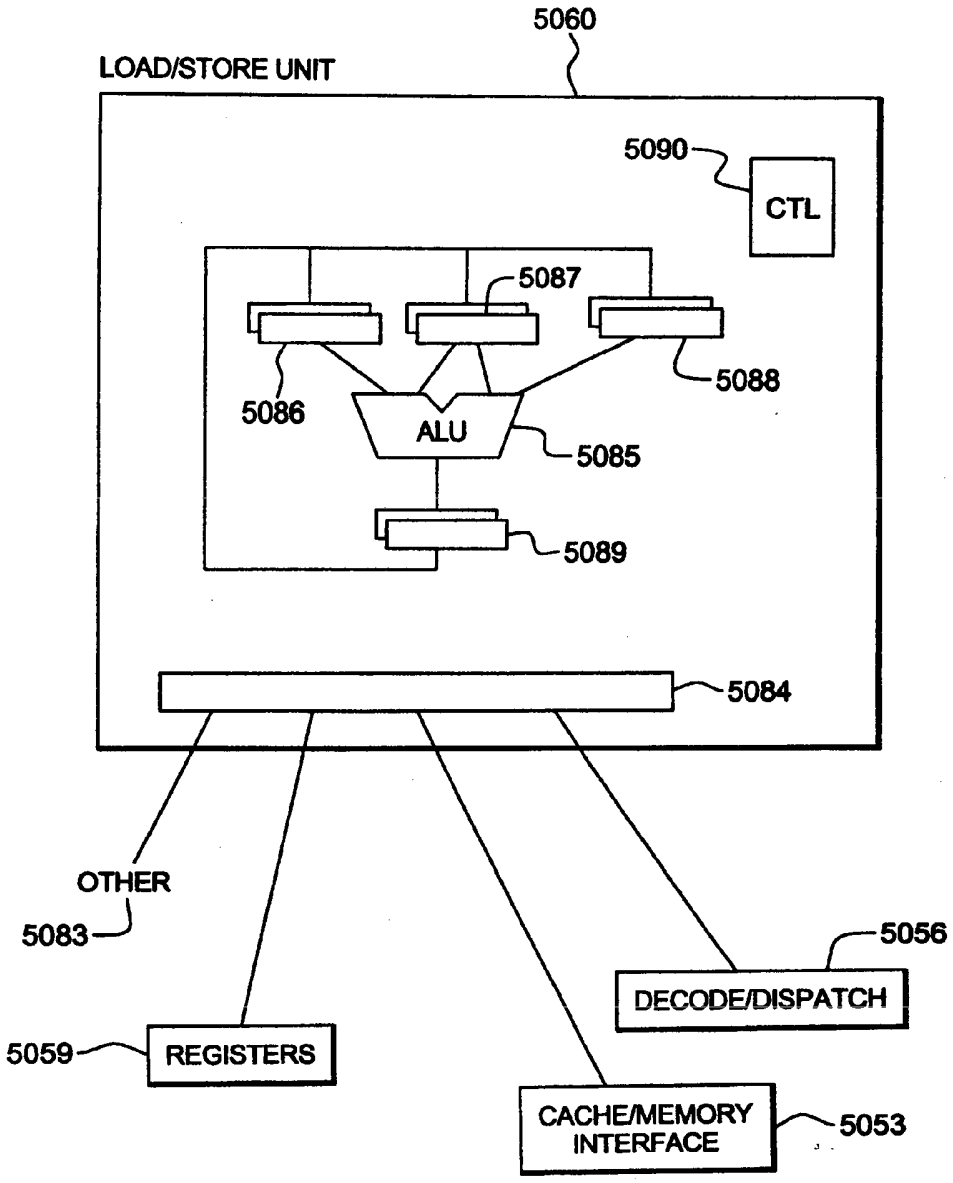



FIG. 15C

Legenda da FIG.15C

LOAD/STORE UNIT - UNIDADE DE CARGA/ARMAZENAMENTO

OTHER - OUTROS

5059 - REGISTOS

5056 - DESCODIFICAR/ENVIAR

5053 - INTERFACE DE *CACHE*/MEMÓRIA

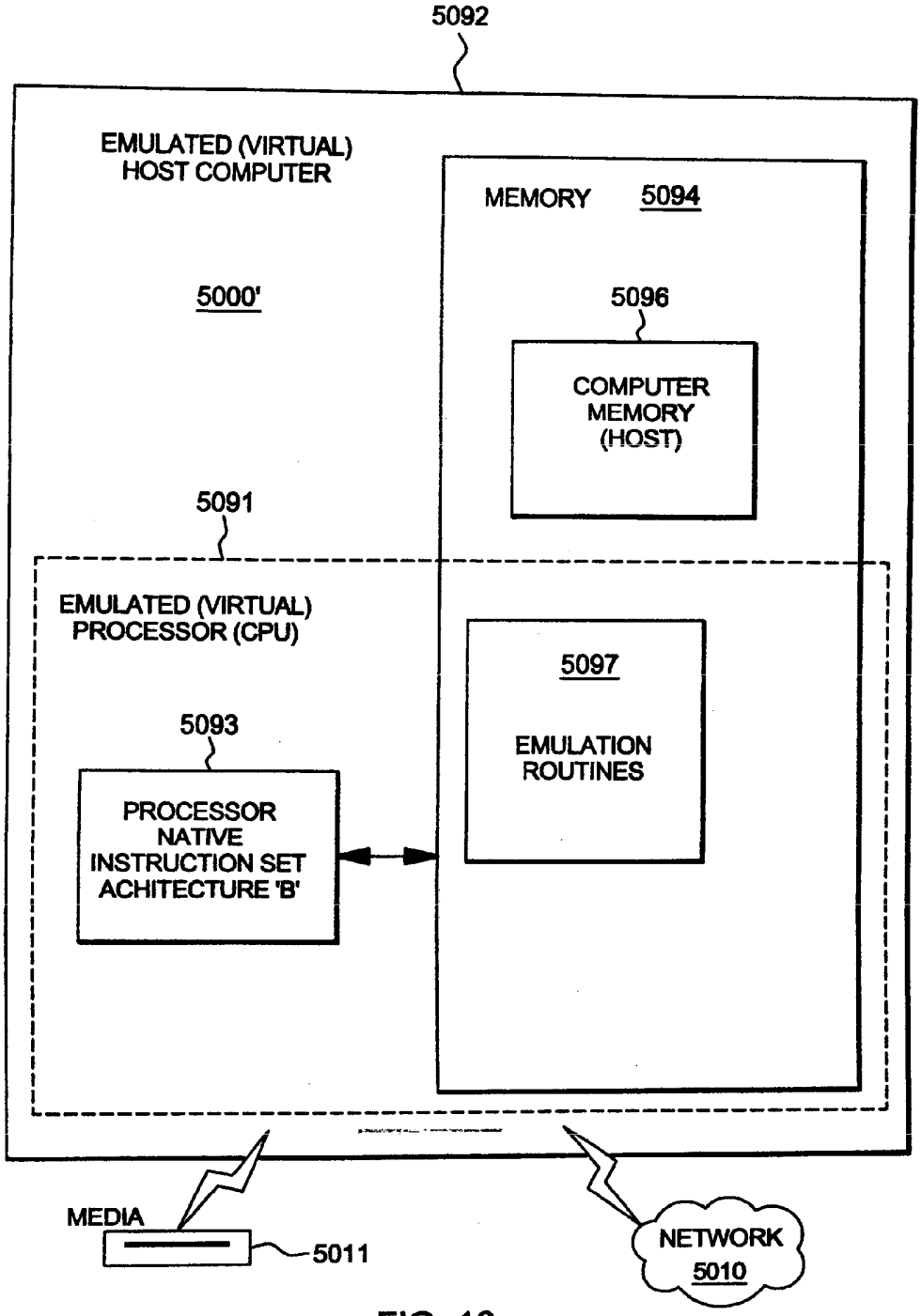

5092
EMULATED (VIRTUAL)
HOST COMPUTER
5000'
MEMORY 5094
5096
COMPUTER
MEMORY
(HOST)
5091
EMULATED (VIRTUAL)
PROCESSOR (CPU)
5097
EMULATION
ROUTINES
5093
PROCESSOR
NATIVE
INSTRUCTION SET
ACHITECTURE 'B'
MEDIA
5011
NETWORK
5010

FIG. 16

Legenda da FIG.16

EMULATED (VIRTUAL) HOST COMPUTER – COMPUTADOR *HOST* EMULADO (VIRTUAL)

MEMORY – MEMÓRIA

5096 – MEMÓRIA DO COMPUTADOR (*HOST*)

EMULATED (VIRTUAL) PROCESSOR (CPU) – PROCESSADOR (CPU) EMULADO (VIRTUAL)

5093 – CONJUNTO DE INSTRUÇÕES NATIVO DO PROCESSADOR DE ARQUITETURA ‘B’

5097 – ROTINAS DE EMULAÇÃO

5011 – SUPORTE

5010 – REDE

# REFERÊNCIAS CITADAS NA DESCRIÇÃO

*Esta lista de referências citadas pelo requerente é apenas para conveniência do leitor. A mesma não faz parte do documento da patente Europeia. Ainda que tenha sido tomado o devido cuidado ao compilar as referências, podem não estar excluídos erros ou omissões e o IEP declina quaisquer responsabilidades a esse respeito.*

**Documentos de patentes citadas na descrição**


- US 20080168208 A1, Gregg
- US 6658521 B
- US 5551013 A
- US 6308255 B
- US 5790825 A
- US 20090182966 A1, Greiner
- US 20080114906 A1
- US 5574873 A
- US 6463582 B


**Literatura que não é de patentes citada na descrição**


- z/Architecture Principles of Operation. IBM Publication No. SA22-7832-07, February 2009