РОССИЙСКАЯ ФЕДЕРАЦИЯ

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(19) **RU** (11) **2 562 372**(13) **C2**

(51) МПК
*G06F 13/28* (2006.01)

# (12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: **2012150402/08, 08.11.2010**

(24) Дата начала отсчета срока действия патента:
**08.11.2010**

Приоритет(ы):
(30) Конвенционный приоритет:
**23.06.2010 US 12/821,187**

(43) Дата публикации заявки: **10.06.2014** Бюл. № **16**

(45) Опубликовано: **10.09.2015** Бюл. № **25**

(56) Список документов, цитированных в отчете о поиске: **US 2006/0195617 A1,31.08.2006. EP 0552873 A1, 28.07.1993. RU 2008115944 A, 27.10.2009. RU 2371758 C2, 27.10.2009**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **26.11.2012**

(86) Заявка PCT:
**EP 2010/067032 (08.11.2010)**

(87) Публикация заявки PCT:
**WO 2011/160714 (29.12.2011)**

Адрес для переписки:
**105082, Москва, Спартаковский пер., д. 2, стр. 1, секция 1, этаж 3, "ЕВРОМАРКПАТ"**

(72) Автор(ы):
**Дан ГРЕЙНЕР (US),**
**Чарлз ГЕЙНИ (US),**
**Дейвид КРАДДОК (US),**
**Антони КОНЕСКИ (US),**
**Бет ГЛЕНДЕНИНГ (US),**
**Марк ФАРРЕЛ (US),**
**Томас ГРЕГГ (US),**
**Угочукву НЬОКУ-ЧАРЛЗ (US)**

(73) Патентообладатель(и):
**ИНТЕРНЭШНЛ БИЗНЕС МАШИНЗ КОРПОРЕЙШН (US)**

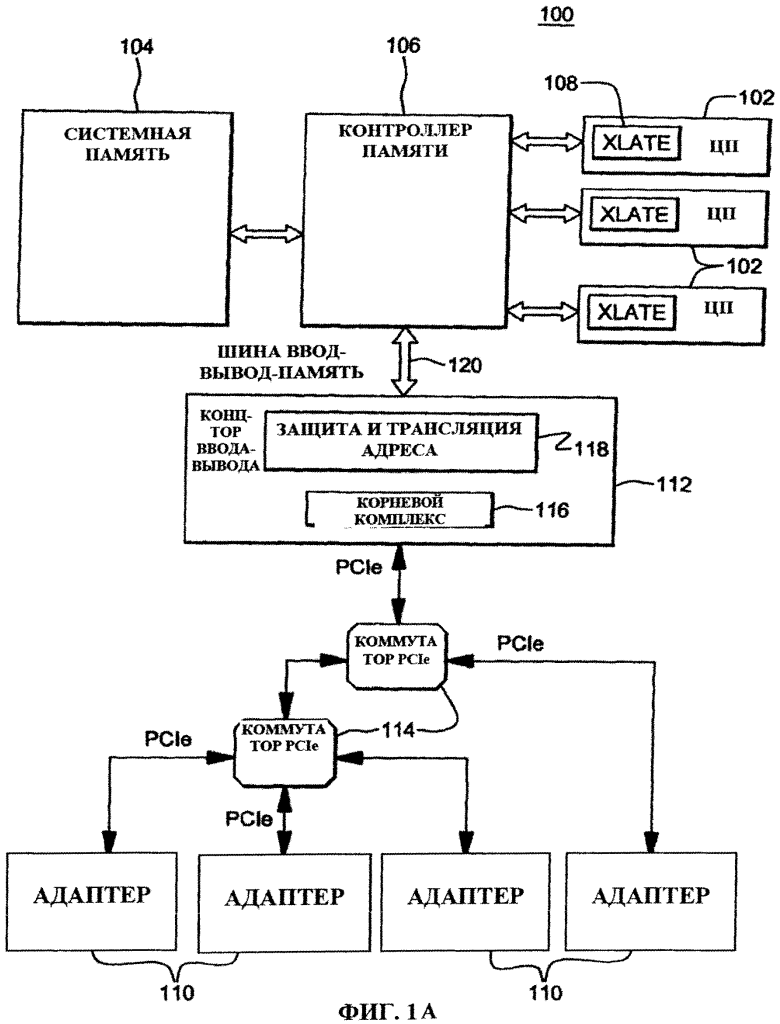
## (54) АКТИВАЦИЯ/ДЕАКТИВАЦИЯ АДАПТЕРОВ ВЫЧИСЛИТЕЛЬНОЙ СРЕДЫ

(57) Реферат:

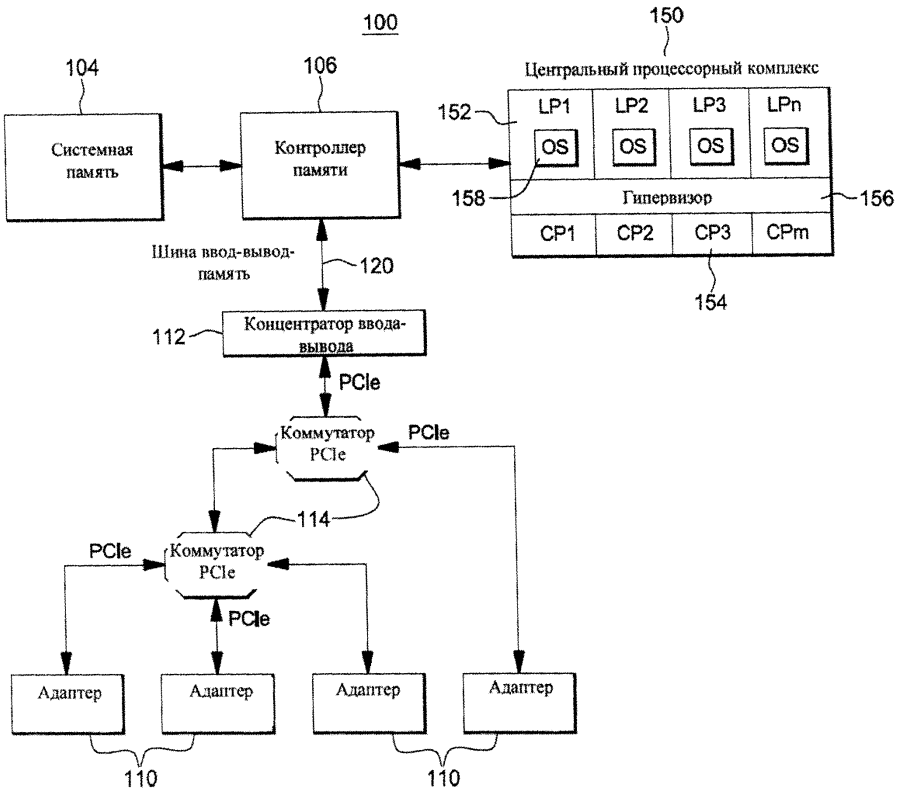
Изобретение относится к средствам обработки ввода-вывода вычислительной среды, в частности к активации/деактивации адаптеров вычислительной среды. Технический результат заключается в обеспечении независимости процедуры активации/деактивации от конкретного аппаратного обеспечения. Активация включает назначение одного или более адресных пространств адаптеру, на основании запроса. Для каждого адресного пространства, назначенного адаптеру, назначается соответствующая запись таблицы устройств. Когда адаптер больше не нужен, он деактивируется и назначенная запись таблицы устройств становится доступной. 2 н. и 19 з.п. ф-лы, 19 ил.

RU 2562372 C2

RU 2562372 C2




ФИГ. 1А




ФИГ. 1Б

RU 2562372 C2

RUSSIAN FEDERATION

(19) **RU** (11) **2 562 372**(13) **C2**

(51) Int. Cl.
*G06F 13/28* (2006.01)

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

## (12) ABSTRACT OF INVENTION

(21)(22) Application: **2012150402/08, 08.11.2010**

(24) Effective date for property rights:
**08.11.2010**

Priority:
(30) Convention priority:
**23.06.2010 US 12/821,187**

(43) Application published: **10.06.2014** Bull. № **16**

(45) Date of publication: **10.09.2015** Bull. № **25**

(85) Commencement of national phase: **26.11.2012**

(86) PCT application:
**EP 2010/067032 (08.11.2010)**

(87) PCT publication:
**WO 2011/160714 (29.12.2011)**

Mail address:
**105082, Moskva, Spartakovskij per., d. 2, str. 1, sektsija 1, ehtazh 3, "EVROMARKPAT"**

(72) Inventor(s):
**Dan GREJNER (US),**
**Charlz GEJNI (US),**
**Dejvid KRADDOK (US),**
**Antoni KONESKI (US),**
**Bet GLENDENING (US),**
**Mark FARREL (US),**
**Tomas GREGG (US),**
**Ugochukvu N'OKU-ChARLZ (US)**

(73) Proprietor(s):
**INTERNEhShNL BIZNES MAShINZ KORPOREJShN (US)**

## (54) COMPUTATION MEDIUM ADAPTER ACTIVATION/DEACTIVATION

(57) Abstract:

FIELD: physics, computation hardware.

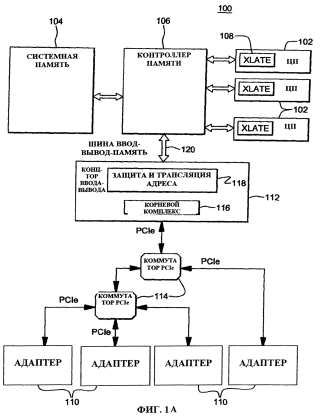
SUBSTANCE: invention relates to computation medium I/O processors, particularly, to activation/deactivation of computation medium adapters. Activation comprises assignment of one or more address spaces to the adapter proceeding from the request. Appropriate entry of the table of devices is assigned for every address space assigned to the adapter. At no need in the adapter, it is deactivated to make assigned entry of the table accessible.

EFFECT: activation/deactivation independent of particular hardware.

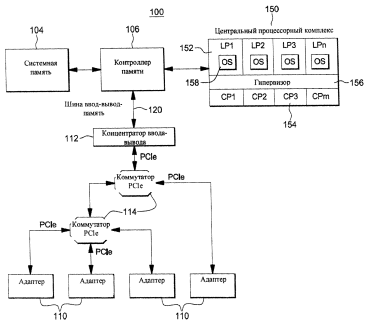
21 cl, 19 dwg

RU 2562372 C2

RU 2562372 C2




ФИГ. 1А




ФИГ. 1Б

RU 2562372 C2

RU 2562372 C2

## ПРЕДШЕСТВУЮЩИЙ УРОВЕНЬ ТЕХНИКИ

Это изобретение в целом относится к обработке ввода-вывода вычислительной среды, и в частности - к активации/деактивации адаптеров вычислительной среды.

Сегодня вычислительные среды имеют различные конфигурации и используют различные типы устройств ввода/вывода (I/O). Для того чтобы использовать устройство ввода-вывода, оно активируется, а затем, когда его использование завершается, оно деактивируется. Способ, которым устройство ввода-вывода активируется/деактивируется, зависит от устройства.

В системе z/Architecture® и ее предшественниках, предлагаемых International Business Machines Corporation, активация и деактивация устройств ввода-вывода, как правило, выполняется на основании канального тракта, элемента управления и субканала.

Различные функции команды Channel Subsystem Call предоставляют интерфейсы, посредством которых операционные системы могут управлять различными ресурсами ввода-вывода.

Однако могут использоваться и другие типы устройств ввода-вывода, которые не включают каналов и субканалов. Например, адаптеры взаимодействия периферийных компонентов (PCI) используют парадигмы соединения и связи, которые отличаются от традиционных устройств ввода-вывода. Спецификация PCI доступна в World Wide Web по адресу www.pcisig.com/home.

Публикация США №2004/0117534 A1, опубликованная 17 июня 2004 (на имя Parry и др.) “Apparatus and Method for Dynamically Enabling and Disabling Interrupt Coalescing in Data Processing System” описывает устройство и способ для динамической активации и деактивации объединения прерываний в системе обработки данных. Настоящее изобретение согласованно включает отслеживание нагрузки ввода-вывода на IOP адаптера ввода-вывода. Встроенное программное обеспечение адаптера ввода-вывода может иметь глобальную переменную, которая хранит счетчики для регистров функции PCI. Каждый счетчик отслеживает число ожидающих выполнения вводов-выводов соответствующего регистра функции PCI. Счетчик увеличивается каждый раз, когда принимается новый ввод-вывод, и уменьшается после отправки сообщения о завершении назад операционной системе. Прерывание по таймеру генерируется периодически так, что может периодически выполняться ISR. В ISR анализируется максимальное хранящееся значение каждого счетчика, видимое после последнего прерывания по таймеру. Когда максимальное хранящееся значение больше предопределенного порогового значения, активируется объединение прерываний.

Публикация США №2010/0005234 A1, опубликованная 7 января 2010 (на имя Ganga и др.). Enabling Functional Dependency in a Multi-Function Device” описывает в одном варианте осуществления, настоящее изобретение включает способ для считывания конфигурационной информации из многофункционального устройства (MFD), создания дерева зависимостей функциональной зависимости функций, выполняемых MFD, на основании конфигурационной информации, которое указывает, что MFD способно выполнять по меньшей мере одну функцию, зависящую от другой функции, и загрузки программного обеспечения, связанного с функциями, в порядке, основанном, по меньшей мере частично, на указанной функциональной зависимости. Описаны и сформулированы и другие варианты осуществления.

Публикация США №2004/0199700 A1, опубликованная 7 октября 2004 (на имя Shawn Adam Clayton), “Virtual Peripheral Component Interconnect Multiple-Function Device”, описывает устройство взаимосвязи периферийных компонентов (PCI), содержащее интерфейс шины, подключенный к шине взаимосвязи компонентов, множество наборов

регистров конфигурационного пространства и виртуальную многофункциональную логику. Каждый набор регистров конфигурационного пространства связан с функцией. Виртуальная многофункциональная логика подключена к интерфейсу шины и набору регистров конфигурационного пространства. Виртуальная многофункциональная логика обеспечивает доступ к множеству регистров конфигурационного пространства для множества функций. Виртуальная многофункциональная логика также дает возможность множеству функций разделять интерфейс шины и другую внутреннюю логику.

## КРАТКОЕ ИЗЛОЖЕНИЕ СУЩНОСТИ ИЗОБРЕТЕНИЯ

В соответствии с изобретением предоставляется функциональная способность для активации/деактивации адаптеров, таких как адаптеры PCI. При этом эта функциональная способность, как она представляется операционной системе, является общей для всех адаптеров, и поэтому считается независимой от устройств.

Недостатки имеющегося уровня техники преодолеваются и преимущества обеспечиваются в способе активации адаптеров в вычислительной среде, характеризующемся тем, что в ответ на выполнение команды Call Logical Processor (CLP), выданной операционной системой, для активации адаптера, выбранного операционной системой, содержащей дескриптор функции, идентифицирующий адаптер и имеющий указатель неактивированного адаптера, команда CLP запрашивает число адресных пространств прямого доступа к памяти (DMA),которые должны быть назначены адаптеру, причем указанное выполнение активирует одно или более адресных пространств DMA и включает: а) активацию адаптера, включающую активацию регистрации для трансляции адреса и прерываний для поддержки прямых доступов к памяти и инициируемых сообщениями прерываний для адаптера, определение того, что запрошенное число адресных пространств DMA являются доступным, выполняемое путем проверки наличия записей таблицы устройств для запрошенного числа адресных пространств DMA, и назначение адаптеру числа записей таблицы устройств, соответствующего запрошенному числу адресных пространств DMA; и б) возврат дескриптора функции, имеющего указатель активированного адаптера.

Объектом изобретения является также компьютерная система для активации адаптеров в вычислительной среде, содержащая память, процессор, связанный с памятью, и элемент запросов, реагирующий на выполнение команды Call Logical Processor (CLP), выданной операционной системой, для активации адаптера, выбранного операционной системой, и содержащей дескриптор функции, идентифицирующий адаптер и имеющий указатель неактивированного адаптера, причем команда CLP запрашивает число адресных пространств прямого доступа к памяти (DMA),которые должны быть назначены адаптеру, а указанное выполнение активирует одно или более адресных пространств DMA и включает: а) активацию адаптера, включающую активацию регистрации для трансляции адреса и прерываний для поддержки прямых доступов к памяти и инициируемых сообщениями прерываний для адаптера, определение того, что запрошенное число адресных пространств DMA являются доступным, выполняемое путем проверки наличия записей таблицы устройств для запрошенного числа адресных пространств DMA, и назначение адаптеру числа записей таблицы устройств, соответствующего запрошенному числу адресных пространств DMA; и б) возврат дескриптора функции, имеющего указатель активированного адаптера.

Технический результат, достигаемый при осуществлении изобретения, заключается в обеспечении независимости процедуры активации, деактивации адаптеров от конкретного аппаратного обеспечения. Дополнительные признаки и преимущества

реализуются посредством методик согласно настоящему изобретению. Другие варианты осуществления и признаки изобретения подробно описаны в настоящем изобретении и считаются частью заявленного изобретения.

Предпочтительный вариант осуществления настоящего изобретения теперь будет описан, исключительно в качестве примера, со ссылкой на соответствующие графические материалы, на которых:

ФИГ.1А иллюстрирует один вариант осуществления вычислительной среды для включения и использования одной или нескольких особенностей настоящего изобретения;

ФИГ.1Б иллюстрирует другой вариант осуществления вычислительной среды, в которой содержится и используется одна или несколько особенностей настоящего изобретения;

ФИГ.2 иллюстрирует один вариант осуществления дополнительных подробностей о системной памяти и концентраторе ввода-вывода, представленных на ФИГ.1А и 1Б, в соответствии с одной особенностью настоящего изобретения;

ФИГ.3А представляет собой один пример записи таблицы функций, используемой в соответствии с одной особенностью настоящего изобретения;

ФИГ.3Б представляет собой один вариант осуществления дескриптора функции, используемого в соответствии с одной особенностью настоящего изобретения;

ФИГ.4А один из вариантов осуществления команды Call Logical Processor, используемой согласно одной из особенностей настоящего изобретения;

ФИГ.4Б один из вариантов осуществления блока запроса, используемого командой Call Logical Processor, показанной на ФИГ.4А, в соответствии с одной особенностью настоящего изобретения;

ФИГ.4В иллюстрирует один вариант осуществления блока ответа, предоставляемого командой Call Logical Processor, представленной на ФИГ.4А, в соответствии с одной особенностью настоящего изобретения;

ФИГ.5 иллюстрирует один вариант осуществления логической схемы для активации функции PCI, в соответствии с одной особенностью настоящего изобретения;

ФИГ.6 иллюстрирует один вариант осуществления логической схемы для деактивации функции PCI, в соответствии с одной особенностью настоящего изобретения;

ФИГ.7 иллюстрирует один вариант осуществления компьютерного программного продукта, в котором содержится одна или несколько особенностей настоящего изобретения;

ФИГ.8 иллюстрирует один вариант осуществления хост-компьютерной системы, в которой содержится и используется одна или несколько особенностей настоящего изобретения;

ФИГ.9 иллюстрирует дополнительный пример компьютерной системы, в которой содержится и используется одна или несколько особенностей настоящего изобретения;

ФИГ.10 иллюстрирует другой пример компьютерной системы, содержащей компьютерную сеть, в которой содержится и используется одна или несколько особенностей настоящего изобретения;

ФИГ.11 иллюстрирует один вариант осуществления различных элементов компьютерной системы, в которой содержится и используется одна или несколько особенностей настоящего изобретения;

ФИГ.12А иллюстрирует один вариант осуществления исполнительного устройства компьютерной системы по ФИГ.11, в которой содержится и используется одна или несколько особенностей настоящего изобретения;

ФИГ.12Б иллюстрирует один вариант осуществления блока перехода компьютерной системы, представленной на ФИГ.11, в которой содержится и используется одна или несколько особенностей настоящего изобретения;

ФИГ.12В иллюстрирует один вариант осуществления блока загрузки/сохранения компьютерной системы, представленной на ФИГ.11, в которой содержится и используется одна или несколько особенностей настоящего изобретения; и

ФИГ.13 иллюстрирует один вариант осуществления эмулированной хост-компьютерной системы, в которой содержится и используется одна или несколько особенностей настоящего изобретения.

ПОДРОБНОЕ ОПИСАНИЕ

В соответствии с одной особенностью настоящего изобретения предоставляется функциональная способность для активации/деактивации адаптеров вычислительной среды. С точки зрения операционной системы функциональная способность является независимой от устройств. То есть операционная система выполняет одну и ту же логику независимо от типа адаптера.

Как используется в данном документе, встроенное программное обеспечение включает, например, микрокод, миликод и макрокод процессора. Оно содержит, например, команды аппаратного уровня и/или структуры данных, используемые при реализации высокоуровневого машинного кода. В одном варианте осуществления изобретения оно содержит, например, собственный код, который, как правило, поставляют как микрокод, который содержит выверенное программное обеспечение, или микрокод, характерный для базового аппаратного обеспечения, и управляет доступом операционной системы к аппаратному обеспечению системы.

Также термин “адаптер” включает любой типа адаптера (например, адаптер запоминающего устройства, адаптер обработки, адаптер PCI, другой тип адаптеров ввода-вывода, и т.п.). Кроме того, в примерах, представленных в данном документе, адаптер используется взаимозаменяемо с функцией адаптера (например, функцией PCI). В одном из вариантов осуществления адаптер содержит одну функцию. Тем не менее, в других вариантах осуществления адаптер может содержать множество функций адаптера. В зависимости от того, содержит ли адаптер одну функцию или множество функций, применима одна или несколько особенностей настоящего изобретения. В одном варианте осуществления, если адаптер содержит множество функций адаптера, каждая функция может быть активирована/деактивирована в соответствии с особенностью настоящего изобретения.

Далее будет описан один из вариантов осуществления вычислительной среды, в которой содержится и используется одна или несколько особенностей настоящего изобретения согласно ФИГ.1А. В одном примере вычислительная среда 100 представляет собой сервер System z®, поставляемый International Business Machines Corporation. System z® основан на z/Architecture®, поставляемой International Business Machines Corporation. Подробности об z/Architecture® описаны в публикации IBM® под названием "z/Architecture Principles of Operation", публикация IBM №SA22-7832-07, февраль 2009. IBM®, System z® и z/Architecture® являются зарегистрированными товарными знаками International Business Machines Corporation, Армонк, Нью-Йорк. Другие названия, используемые в заявке, могут являться зарегистрированными товарными знаками, товарными знаками или названиями продуктов International Business Machines Corporation или других компаний.

В одном из примеров вычислительная среда 100 содержит один или несколько центральных процессоров (ЦП) 102, связанных с системной памятью 104 (иначе

называемой основной памятью) посредством контроллера 106 памяти. Для доступа к системной памяти 104 центральный процессор 102 выдает запрос чтения или записи, в котором содержится адрес, используемый для доступа к системной памяти. Поскольку адрес, содержащийся в запросе, обычно не может непосредственно использоваться для доступа к системной памяти, он транслируется в адрес, который может непосредственно использоваться для доступа к системной памяти. Адрес транслируется посредством механизма 108 трансляции (XLATE). Например, адрес транслируется из виртуального адреса в действительный или абсолютный адрес с использованием, например, динамической трансляции адресов (DAT).

Запрос, содержащий транслированный адрес, принимается контроллером 106 памяти. В одном из примеров контроллер 106 памяти состоит из аппаратного обеспечения и используется для арбитража при доступе к системной памяти и для обеспечения непротиворечивости памяти. Этот арбитраж осуществляется применительно к запросам, принимаемым от ЦП 102, а также запросам, принимаемым от одного или нескольких адаптеров 110. Подобно центральным процессорам адаптеры выдают в системную память 104 запросы на получение доступа к системной памяти.

В одном примере адаптер 110 является адаптером взаимодействия периферийных компонентов (PCI) или PCI express (PCIe), содержащим одну или более функций. Функция PCI выдает запрос, который маршрутизируется в концентратор 112 ввода-вывода (например, концентратор PCI) посредством одного или нескольких коммутаторов (например, коммутаторов PCIe) 114. В одном примере, концентратор ввода-вывода состоит из аппаратного обеспечения, включая один или более конечных автоматов, и соединен с контроллером 106 памяти посредством шины 120 ввод-ввывод-память.

Концентратор ввода-вывода содержит, например, корневой комплекс 116, который принимает запрос от коммутатора. Запрос содержит адрес ввода-вывода, который подается на блок 118 защиты и трансляции адреса, который осуществляет доступ к информации, используемой для запроса. В качестве примеров, запрос может содержать адрес ввода-вывода, используемый для выполнения операции прямого доступа к памяти (DMA) или для запроса инициируемого сообщениями прерывания (MSI). Блок 118 защиты и трансляции адреса осуществляет доступ к информации, используемой для запроса DMA или MSI. В качестве конкретного примера, для операции DMA, информация может быть получена, чтобы транслировать адрес. Транслированный адрес затем направляется на контроллер памяти, чтобы осуществлять доступ к системной памяти.

В еще одном варианте осуществления вычислительной среды, в дополнение к одному или более ЦП 102 или вместо них, с контроллером памяти 106 соединяется процессорный комплекс, как показано на ФИГ.1Б. В этом примере центральный процессорный комплекс 150 содержит, например, один или более разделов или областей 152 (например, логические разделы LP1-LPn), один или более центральных процессоров (например, CP1-CPm) 154 и гипервизор 156 (например, менеджер логических разделов), каждый из которых описывается ниже.

Каждый логический раздел 152 способен функционировать как отдельная система. То есть, каждый логический раздел, если необходимо, может быть независимо перезагружен, изначально загружен операционной системой или гипервизором (таким как z/VM®, поставляемым International Business Machines Corporation, Армонк, Нью-Йорк) и работать с различными программами. Операционная система, гипервизор или прикладная программа, работающие в логическом разделе, как кажется, имеет доступ к системе целиком и полностью, но доступна лишь ее часть. Сочетание аппаратного обеспечения и лицензированного внутреннего кода (также называемого микрокодом

или милликодом) удерживает программу в логическом разделе от пересечения с программой в другом логическом разделе. Это позволяет нескольким разным логическим разделам работать на одном или множестве физических процессоров в режиме квантования времени. В этом конкретном примере каждый логический раздел обладает резидентной операционной системой 158, которая может отличаться для одного или более логических разделов. В одном варианте осуществления операционная система 158 является операционной системой z/OS® или zLinux, поставляемой International Business Machines Corporation, Армонк, Нью-Йорк. z/OS® и z/VM® являются зарегистрированными товарными знаками International Business Machines Corporation, Армонк, Нью-Йорк.

Центральные процессоры 154 представляют собой физические процессорные ресурсы, которые выделены логическим разделам. Например, логический раздел 152 содержит один или более логических процессоров, каждый из которых представляет весь физический процессорный ресурс 154, выделенный разделу, или его долю. Лежащий в основе процессорный ресурс может или назначаться только этому разделу, или разделяться с другим разделом.

Управление логическими разделами 152 осуществляется гипервизором 156, реализуемым встроенным программным обеспечением, работающим на процессорах 154. Логические разделы 152 и гипервизор 156 каждый содержат одну или более программ, постоянно находящихся в соответствующих частях центрального запоминающего устройства, связанного с центральными процессорами. Одним примером гипервизора 156 является Менеджер процессорных ресурсов/систем (PR/SM), поставляемый Business Machines Corporation, Армонк, Нью-Йорк.

Хотя в этом примере описан центральный процессорный комплекс, имеющий логические разделы, одна или более особенностей настоящего изобретения могут включаться в и использоваться другими устройствами обработки, включая одно- или многопроцессорные устройства обработки, которые не разделяются на разделы, среди прочих. Центральный процессорный комплекс, описанный в данном документе, является лишь одним примером.

Дальнейшие подробности о системной памяти и концентраторе ввода-вывода описываются со ссылкой на ФИГ.2. В этом примере контроллер памяти не показан, но может использоваться. Концентратор ввода-вывода может быть соединен с системной памятью 104 и/или процессором 204 непосредственно или посредством контроллера памяти.

Обращаясь к ФИГ.2, в одном примере системная память 104 содержит одно или более адресных пространств 200. Адресное пространство - это конкретная часть системной памяти, которая была назначена для конкретного компонента вычислительной среды, такой как конкретный адаптер. В одном примере адресное пространство доступно путем прямого доступа к памяти (DMA), инициируемого адаптером, и поэтому адресное пространство в примерах данного документа называется адресным пространством DMA. Однако в других примерах для доступа к адресному пространству прямой доступ к памяти не используется.

В одном примере имеется операционная система 202, выполняющаяся в процессоре 204 (например, ЦП 102 или ЦП 154, выделенных LP 152), которая назначает адресное пространство DMA конкретному адаптеру. Это назначение выполняется посредством процесса регистрации, который вызывает инициализацию (например, посредством надежного программного обеспечения) записи 210 таблицы устройств для этого адаптера. Имеется одна запись таблицы устройств на каждое назначенное адресное

пространство, и эта запись таблицы устройств связана с единственным адаптером. Эта запись таблицы устройств находится в таблице 212 устройств, расположенной в концентраторе 112 ввода-вывода. Например, таблица 212 устройств находится в блоке защиты и трансляции адреса концентратора ввода-вывода.

В одном примере запись 210 таблицы устройств содержит информацию, используемую для предоставления различных сервисов для адаптера. Например, запись таблицы устройств содержит указатель 214 активации, который указывает, активирована ли запись таблицы устройств для конкретного адаптера. Запись таблицы устройств может содержать больше или меньше информации, или другую информацию, для операций активации/деактивации, а также для других предоставляемых сервисов, таких как трансляция адреса, управление прерываниями и т.п.

В одном варианте осуществления запись таблицы устройств, которая должна использоваться конкретным адаптером, который подает запрос, определяется с помощью идентификатора реквестора (RID) (и/или части адреса), находящегося в запросе, поданном функцией PCI (220), связанной с адаптером. Реквестор (например, 16-битное значение, определяющее, например, номер шины, номер устройства и номер функции) включается в запрос, также как и адрес ввода-вывода, который необходимо использовать. Запрос, включая RID и адрес ввода-вывода, подается, например, на ассоциативное запоминающее устройство (CAM) 230 через, например, коммутатор 114, который используется, чтобы предоставлять индексное значение. Например, в CAM содержится множество записей, каждая из которых соответствует индексу в таблице устройств. Каждая запись в CAM содержит значение RID. Если, например, полученный RID соответствует значению, содержащемуся в записи в CAM, для определения положения записи таблицы устройств используется соответствующий индекс таблицы устройств. То есть, выход CAM используется для индексации в таблице 212 устройств, чтобы определять положение записи 210 таблицы устройств. Если соответствие отсутствует, принятый пакет отбрасывается (в других вариантах осуществления не требуется CAM или другое средство поиска, и RID используется в качестве индекса).

В дополнение к записи таблицы устройств, с адаптером также связывается другая структура данных, которая содержит информацию об адаптере. В конкретных примерах, описанных в данном документе, адаптер представляет собой функцию PCI, поэтому структура данных называется записью таблицы функций (FTE). Хотя примеры, описанные в данном документе, относятся к функциям PCI, в других вариантах осуществления могут активироваться/деактивироваться другие адаптерные функции или адаптеры, в соответствии с одной особенностью настоящего изобретения.

Как показано на ФИГ.3А, в одном примере запись 300 таблицы функций является записью в таблице 302 функций, хранящейся, например, в защищенной памяти. Каждая запись 300 таблицы функций содержит информацию, которая должна использоваться при обработке, связанной с ее адаптером. В одном примере запись 300 таблицы функций содержит номер 308 экземпляра, указывающий конкретный экземпляр функции адаптера, связанной с записью таблицы функций; один или более индексов 310 записей таблицы устройств, каждый из которых используется в качестве индекса в таблице устройств, чтобы определять положение его соответствующей записи таблицы устройств (функция PCI может иметь множество адресных пространств, приписанных к ней, поэтому и множество записей DTE); указатель 312 занятости, который указывает, занята ли функция PCI; указатель 314 состояния постоянной ошибки, который указывает, находится ли функция в состоянии постоянной ошибки; указатель 316 инициированного

восстановления, который указывает, было ли инициировано восстановление для функции; указатель 318 разрешения, который указывает, имеет ли операционная система, пытающаяся активировать функцию PCI, полномочия на это; и указатель 320 активации, указывающий, активирована ли функция (например, 1=активирована, 0= деактивирована).

В одном примере указатель занятости, указатель состояния постоянной ошибки и указатель инициированного восстановления устанавливаются на основании текущего контроля, выполняемого встроенным программным обеспечением. Кроме того, указатель разрешения устанавливается, например, на основании политики. В других вариантах осуществления запись таблицы функций может содержать больше или меньше информации, или другую информацию.

В одном варианте осуществления для определения положения записи таблицы функций в таблице функций, которая содержит одну или более записей, используется идентификатор функции, такой как дескриптор функции. Например, один или несколько бит дескриптора функции используются в качестве индекса таблицы функции для размещения конкретной записи таблицы функции.

Со ссылкой на ФИГ.3Б описываются дополнительные подробности о дескрипторе функции. В одном примере дескриптор 350 функции включает указатель 352 активации, который указывает, активирован ли дескриптор функции PCI; номер 354 функции PCI, который идентифицирует функцию (это статический идентификатор) и, в одном варианте осуществления, - является индексом в таблице функций; и номер 356 экземпляра, который указывает конкретный экземпляр этого дескриптора функции. Например, каждый раз, когда функция активируется, номер экземпляра увеличивается, чтобы предоставлять новый номер экземпляра.

Чтобы использовать функцию PCI, ее необходимо активировать. Например, операционная система, которая хотела бы использовать функцию PCI, выполняет запрос, чтобы определить одну или более функций, которые она может использовать (на основании конфигурации ввода-вывода), и выбирает одну из этих функций для активации. В одном примере функция активируется с использованием команды установки функции PCI команды Call Logical Processor. Один вариант осуществления этой команды представлен на ФИГ.4А. Как показано, в одном примере команда 400 Call Logical Processor включает код 402 операции, указывающий, что это команда Call Logical Processor; и указание 404 для команды. В одном примере это указание является адресом блока запроса, описывающего команду, которую необходимо выполнить. Один вариант осуществления такого блока запроса представлен на ФИГ.4Б.

Как показано на ФИГ.4Б, в одном примере блок 420 запроса содержит ряд параметров, таких как, например, поле 422 длины, указывающее длину блока запроса; поле 424 команды, указывающее команду установки функции PCI; дескриптор 426 функции PCI, который является дескриптором, который необходимо передать функции активации или деактивации; код 428 операции, который используется для обозначения операции активации или деактивации; и число адресных пространств (DMAAS) 430, которое указывает запрошенное число адресных пространств, которое необходимо связывать с конкретной функцией PCI. В других вариантах осуществления может включаться больше или меньше информации, или другая информация.

Например, в виртуальной среде, в которой команда подается хостом или постраничным гостем режима сохранения, предоставляется обозначение гостя. Также возможны и другие варианты. В одном примере, в z/Architecture®, постраничный гость интерпретативно выполняется посредством команды Start Interpretive Execution (SIE),

на 2 уровне интерпретации. Например, гипервизор логического разбиения (LPAR) выполняет команду SIE, чтобы начать логическое разбиение в физической постоянной памяти. Если в этом логическом разделе операционной системой является z/VM®, она подает команду SIE, чтобы выполнять свои гостевые (виртуальные) машины в своем V=V (виртуальном) запоминающем устройстве. Таким образом, гипервизор LPAR использует уровень - 1 SIE, а гипервизор z/VM® использует уровень - 2 SIE.

С учетом подачи и обработки команды Call Logical Processor, возвращается блок ответа, а информация, включаемая в блок ответа, зависит от операции, которую необходимо выполнить. Один вариант осуществления блока ответа представлен на ФИГ.4В. В одном примере блок 450 ответа включает поле 452 длины, указывающее длину блока ответа; код 454 ответа, указывающий состояние команды; и дескриптор 456 функции PCI, который определяет функцию PCI. С учетом команды активации дескриптор функции PCI является активированным дескриптором функции PCI. Кроме того, по завершении операции деактивации, дескриптор функции PCI является общим дескриптором, который может быть активирован функцией активации в будущем.

Один вариант осуществления логической схемы для активации функции PCI описывается со ссылкой на ФИГ.5. В одном примере эта логическая схема инициируется с учетом подачи команды Call Logical Processor, в которой устанавливается команда для команды установки функции PCI и устанавливается код операции для функции активации. Эта логическая схема выполняется, например, процессором с учетом операционной системы или драйвера устройства операционной системы, имеющих разрешение на выполнение этой логики, подающей команду. В других вариантах осуществления эта логическая схема может выполняться без использования команды Call Logical Processor.

Как показано на ФИГ.5, сначала на этапе 500 запроса осуществляется определение, является ли дескриптор, предоставленный в блоке запроса команды Call Logical Processor, достоверным дескриптором. То есть, указывает ли дескриптор на достоверную запись в таблице функций или же он находится за пределами диапазона достоверных записей (например, обозначает ли часть дескриптора с номером функции установленную функцию). Если дескриптор неизвестен, то предоставляется соответствующий код ответа, указывающий, что дескриптор не распознан. Однако если дескриптор известен, то на этапе 504 запроса осуществляется следующий запрос, активирован ли дескриптор. Это определение осуществляется проверкой указателя активации в дескрипторе функции PCI. Если указание, указывающее, что дескриптор активирован, установлено, то на этапе 506 возвращается код ответа, указывающий на это.

Однако если дескриптор известен и не активирован (т.е. действительный для активации), то на этапе 508 запроса осуществляется следующее определение, является ли запрошенное число адресных пространств, которые должны быть назначены функции PCI, большим, чем максимальное значение. Чтобы осуществить это определение, число адресных пространств DMA, как определено в блоке запроса, сравнивается с максимальным значением (предоставленным на основании политики, в одном примере). Если число адресных пространств больше, чем максимальное значение, то на этапе 510 предоставляется код ответа, указывающий на недостоверное значение для адресных пространств DMA. Иначе на этапе 512 осуществляется определение, доступно ли число запрошенных адресных пространств. Это определение осуществляется путем проверки, имеются ли записи таблицы устройств, доступные для запрошенного числа адресных пространств. Если число запрошенных адресных пространство недоступно, то на этапе 514 возвращается код ответа, указывающий, что ресурсов недостаточно. Иначе

обработка продолжается, чтобы активировать функцию PCI.

Предоставленный дескриптор на этапе 516 используется для определения положения записи таблицы функций. Например, один или более указанных битов дескриптора используются в качестве индекса в таблице функций, чтобы определять положение конкретной записи таблицы функций. С учетом определения положения соответствующей записи таблицы функций, на этапе 518 запроса осуществляется определение, активирована ли функция. Это определение осуществляется путем проверки указателя активации в записи таблицы функций. Если функция уже активирована (т.е., указатель установлен в единицу), то на этапе 520 возвращается код ответа, указывающий, что функция PCI уже находится в запрошенном состоянии.

Если функция еще не активирована, то на этапе 522 запроса обработка продолжается определением, находится ли функция в состоянии постоянной ошибки. Если указатель состояния постоянной ошибки в записи таблицы функций указывает, что она находится в состоянии постоянной ошибки, то на этапе 524 возвращается код ответа, указывающий на это. Однако если функция не находится в состоянии постоянной ошибки, на этапе 526 запроса осуществляется следующее определение, было ли для функции инициировано восстановление после ошибки. Если указатель инициированного восстановления в записи таблицы функций установлен, то на этапе 528 предоставляется код ответа, указывающий, что восстановление было инициировано. Иначе на этапе 530 запроса осуществляется следующий запрос, занята ли функция PCI. И снова, если проверка указателя занятости в записи таблицы функций указывает, что функция PCI занята, то на этапе 532 предоставляется указание этого. Однако если функция PCI не находится в состоянии постоянной ошибки, восстановление не инициировано и она не занята, то на этапе 534 осуществляется следующий запрос, разрешается ли операционной системе активировать эту функцию PCI. Если это не разрешается на основании указателя разрешения записи таблицы функций, то на этапе 536 предоставляется код ответа, указывающий несанкционированное действие. Однако если все эти проверки успешно пройдены, то на этапе 538 запроса осуществляется следующее определение, имеются ли какие-либо записи DTE, доступные для этой функции PCI. В качестве примеров, определение доступных записей PCI может основываться на записях DTE, которые в текущее время не активированы в концентраторе ввода-вывода. Дополнительно для дальнейшего ограничения числа записей DTE, доступных для данной операционной системы или логического раздела, может применяться политика. Может быть назначена любая находящаяся в наличии DTE, доступная адаптеру. Если доступных записей DTE нет, то на этапе 540 возвращается код ответа, указывающий, что одна или более запрошенных записей DTE недоступны.

Если записи DTE доступны, то на этапе 542 ряд записей DTE, соответствующий запрошенному числу адресных пространств, назначается и активируется. В одном примере активация включает установку указателя активации в каждой DTE, подлежащей активации. Кроме того, активация включает, в этом примере, установку CAM, чтобы предоставлять индекс каждой DTE. Например, для каждой DTE, запись в CAM сохраняется с индексом.

Кроме того, на этапе 544 записи DTE связываются с записью таблицы функций. Это включает, например, включение каждого индекса DTE в запись таблицы функций. Функция затем на этапе 546 отмечается как активированная путем установки указателя активации в записи таблицы функций. Более того, на этапе 548 устанавливается бит активации в дескрипторе и обновляется номер экземпляра. Этот активированный дескриптор затем возвращается на этапе 550, позволяя использовать адаптер PCI.

Например, с учетом активации функции, могут выполняться регистрация для трансляций адреса и прерываний, функцией PCI могут выполняться операции DMA; функцией могут запрашиваться прерывания; и/или функции могут подаваться команды загрузить, сохранить, сохранить блок и/или изменить управление функцией (например, PCI Load, PCI Store, PCI Store Block, Modify PCI Function Controls).

Один вариант осуществления логической схемы для деактивации функции PCI описывается со ссылкой на ФИГ.6. В этом примере команда установки функции PCI запрашивается посредством команды управления Call Logical Processor, в которой код операции установлен в состояние деактивировать; однако в других вариантах осуществления такая команда управления не используется. В одном примере эту логику выполняет операционная система или драйвер устройства операционной системы.

Как представлено на ФИГ.6, сначала на этапе 600 запроса осуществляется определение, является ли дескриптор, предоставленный в блоке запроса для команды Call Logical Processor, известным дескриптором. Например, осуществляется проверка, указывает ли дескриптор на достоверную запись в таблице функций. Если дескриптор указывает на достоверную запись, то дескриптор является известным дескриптором. Если нет, то на этапе 602 предоставляется код ответа, указывающий на неизвестный дескриптор. Однако если дескриптор известен, то на этапе 604 запроса осуществляется следующая проверка, не деактивирован ли уже дескриптор. Если указатель активации в дескрипторе указывает, что дескриптор уже деактивирован, то предоставляется код ответа, указывающий это. Иначе, если дескриптор известен и активирован, дескриптор является действительным для операции деактивации и используется для определения положения записи таблицы функций на этапе 608.

С учетом получения записи таблицы функций, на этапе 610 осуществляется определение, является ли функция уже деактивированной, как указано указателем активации в записи таблицы функций. Если указатель не установлен (т.е., указатель активации=0), то на этапе 612 предоставляется код ответа, указывающий, что функция уже деактивирована.

Если указатель установлен (например, активирован=1), то на этапе 614 запроса осуществляется определение, находится ли функция в состоянии постоянной ошибки. Если она находится в состоянии постоянной ошибки, то на этапе 616 предоставляется код ответа, указывающий ошибку. Иначе на этапе 618 запроса осуществляется определение, инициировано ли восстановление после ошибки. Если восстановление после ошибки инициировано, то на этапе 620 предоставляется код ответа, указывающий это. Если восстановление после ошибки не инициировано, то на этапе 622 запроса осуществляется определение, занята ли функция PCI. Если она занята, то на этапе 624 предоставляется код ответа, указывающий это. Иначе на этапе 626 осуществляется определение, уполномочена ли операционная система подавать эту команду деактивации. Это определение осуществляется, например, путем проверки указателя разрешения в записи таблицы функций, а также сравнением номера экземпляра в дескрипторе с номером экземпляра в записи таблицы функций. Если они неравны, то делается запрос деактивировать другой экземпляр функции, которая была активирована. Если указатель разрешения говорит об отсутствии разрешения, или номера экземпляра неравны, операционная система не уполномочена, и на этапе 628 предоставляется код ответа, указывающий отсутствие полномочий. Однако, если указатель разрешения говорит о наличии разрешения, и номера экземпляра равны, операционная система имеет полномочия.

Если все проверки успешны, то на этапе 630 функция деактивируется. В одном

примере это включает установку указателя активации в записи таблицы функций в ноль (или иным образом в отключенное состояние). После этого параметры регистрации в записях DTE, связанных с этой функцией PCI, сбрасываются на этапе 632, и на этапе 634 эти записи DTE освобождаются для использования другими функциями PCI. Например, бит активации в DTE сбрасывается, и запись CAM, связанная с DTE, удаляется. Кроме того, указатель активации в дескрипторе на этапе 636 переустанавливается в ноль (или некоторое другое значение, указывающее деактивацию или отключение), и на этапе 638 возвращается деактивированный дескриптор.

В еще одном варианте осуществления, если одна или более проверок на этапах 614, 618 и 622 запроса не проходят, то деактивация все равно продолжается, и может предоставляться код ответа, указывающий на это.

Выше подробно описана функциональная способность для активации/деактивации функции PCI. Эта функциональная способность является независимой от устройств с точки зрения операционной системы и обеспечивает тонкую детализацию управления, при которой операционная система способна активировать и деактивировать функцию PCI. С учетом деактивации функции другая операционная система может активировать функцию. Это позволяет множеству операционных систем (например, в среде с логическим разбиением разделов) совместно использовать функции адаптера.

В описанных в данном документе вариантах осуществления адаптеры являются адаптерами PCI. PCI, как используется в данном документе, относится к любым адаптерам, реализованным в соответствии со спецификацией на основе PCI, как определяется организацией Peripheral Component Interconnect Special Interest Group (PCI-SIG), включая, без ограничения, PCI или PCIe. В одном конкретном примере Peripheral Component Interconnect Express (PCIe) представляет собой стандарт взаимосвязи на уровне компонентов, который определяет двунаправленный протокол связи для транзакций между адаптерами ввода-вывода и хост-системами. Осуществление связи PCIe инкапсулируется в пакетах в соответствии со стандартом PCIe для передачи по шине PCIe. Транзакции, возникающие в адаптерах ввода-вывода и завершающиеся в хост-системах, называются направленными вверх транзакциями. Транзакции, возникающие в хост-системах и завершающиеся в адаптерах ввода-вывода, называются направленными вниз транзакциями. Топология PCIe основана на двухточечных однонаправленных соединениях, которые сочетаются попарно (например, одно направленное вверх соединение, одно направленное вниз соединение), чтобы образовывать шину PCIe. Стандарт PCIe поддерживается и обнародуется организацией PCI-SIG, как указано выше в разделе, описывающем предшествующий уровень техники.

Как будет понятно специалисту в данной области техники, особенности настоящего изобретения могут быть реализованы системой, способом или компьютерным программным продуктом. Соответственно, особенности настоящего изобретения могут иметь форму полностью аппаратного варианта осуществления, полностью программного варианта осуществления (включая встроенное программное обеспечение, резидентное программное обеспечение, микрокод и т.п.) или варианта осуществления, сочетающего программные и аппаратные особенности, которые все в данном документе в целом могут называться “контуром”, “модулем” или “системой”. Кроме того, особенности настоящего изобретения могут иметь форму компьютерного программного продукта, реализованного в одной или нескольких машиночитаемой среде(-ах), с реализованным на них машиночитаемым программным кодом.

Может использоваться любое сочетание одной или нескольких машиночитаемых сред. Машиночитаемая среда может быть машиночитаемой запоминающей средой.

Машиночитаемая запоминающая среда может быть, например, без ограничения, электронной, магнитной, оптической, электромагнитной, инфракрасной или полупроводниковой системой, аппаратом или устройством, или любым подходящим сочетанием вышеуказанного. Более конкретные примеры (не исчерпывающий список) машиночитаемых запоминающих сред включают следующее: электрическое соединение, имеющее один или несколько проводов, съемную дискету для компьютера, жесткий диск, оперативное запоминающее устройство (ОЗУ), постоянное запоминающее устройство (ПЗУ), стираемое программируемое постоянное запоминающее устройство (СПЗУ или флэш-память), оптоволокно, постоянное запоминающее устройство на съемном диске (CD-ROM), оптическое запоминающее устройство, магнитное запоминающее устройство или любое подходящее сочетание вышеуказанного. В контексте этого документа машиночитаемая запоминающая среда может быть любой материальной средой, которая может содержать или хранить программу для использования посредством системы, аппарата или устройства выполнения команд или в связи с ними.

Как показано на ФИГ.7, в одном примере компьютерный программный продукт 700 содержит, например, одну или несколько машиночитаемых запоминающих сред 702 для хранения в них машиночитаемых средств или логики 704 программного кода, чтобы обеспечивать и продвигать одну или более особенностей настоящего изобретения.

Программный код, воплощенный в машиночитаемой среде, может передаваться с использованием соответствующей среды, включая, без ограничения, беспроводную, проводную линию, оптоволоконный кабель, ВЧ-среду и т.д. или любое подходящее сочетание вышеупомянутого.

Компьютерный программный код для выполнения операций, обеспечивающих особенности настоящего изобретения, может быть написан на любом сочетании одного или нескольких языков программирования, включая объектно-ориентированный язык программирования, такой как Java, Smalltalk, C++ и т.п., и традиционные процедурные языки программирования, такие как язык программирования C, ассемблер или подобные языки программирования. Программный код может полностью выполняться на компьютере пользователя, частично на компьютере пользователя, в качестве автономного пакета программного обеспечения, частично на компьютере пользователя и частично на удаленном компьютере или полностью удаленном компьютере или сервере. В последнем случае удаленный компьютер может быть соединен с компьютером пользователя посредством сети любого типа, включая локальную вычислительную сеть (LAN) или глобальную вычислительную сеть (WAN), или соединение может осуществляться с внешним компьютером (например, по сети Интернет с помощью Интернет-провайдера).

Особенности настоящего изобретения описаны в данном документе со ссылкой на структурные схемы и/или блок-схемы способов, устройств (систем) и компьютерных программных продуктов в соответствии с вариантами осуществления изобретения.

Будет понятно, что каждый блок на структурных схемах и/или блок-схемах и сочетания блоков на структурных схемах и/или блок-схемах могут быть реализованы посредством команд компьютерных программ. Эти команды компьютерной программы могут подаваться на процессор компьютера общего назначения, компьютера специального назначения или другого программируемого устройства для обработки данных с целью создания машины, в которой команды, которые выполняются посредством процессора компьютера или другого программируемого устройства для обработки данных, создают средство для реализации функций/действий, определенных

блоком или блоками на структурных схемах и/или блок-схемах.

Эти команды компьютерной программы также могут храниться в машиночитаемой среде, которая может предписывать компьютеру, другому программируемому устройству обработки данных или другим устройствам действовать конкретным образом, так что команды, хранящиеся в машиночитаемой среде, образуют изделие обработки, содержащее команды, которые реализуют функцию/действие, определенные блоком или блоками на структурных схемах и/или блок-схемах.

Команды компьютерной программы также могут загружаться в компьютер, другое программируемое устройство обработки данных или другие устройства, чтобы вызывать выполнение последовательности оперативных этапов компьютером, другим программируемым устройством или другими устройствами, чтобы создавать компьютерно-реализованный процесс, так что команды, которые выполняются на компьютере или другом программируемом устройстве, обеспечивают процессы для реализации функций/действий, заданных блоком или блоками на структурных схемах и/или блок-схемах.

Приведенные на чертежах структурные схемы и блок-схемы иллюстрируют архитектуру, функциональные возможности и работу возможных осуществлений систем, способов и компьютерных программных продуктов в соответствии с различными вариантами осуществления настоящего изобретения. В связи с этим каждый блок на структурных схемах или блок-схемах может представлять модуль, сегмент или часть кода, которая содержит одну или несколько выполняемых команд для реализации заданной логической функции(-й). Следует также отметить, что в некоторых альтернативных осуществлениях указанные в блоке функции могут происходить не в том порядке, в котором они представлены на графических материалах. Например, два блока, показанные последовательно, на самом деле могут выполняться в значительной мере одновременно, или блоки могут иногда выполняться в обратном порядке, в зависимости от связанных функциональных возможностей. Следует также отметить, что каждый блок на блок-схемах и/или структурных схемах и сочетания блоков на блок-схемах и/или структурных схемах могут быть реализованы посредством специализированных аппаратных систем, которые выполняют заданные функции или действия, или сочетаний специализированного аппаратного обеспечения и компьютерных команд.

В придачу к вышесказанному, одна или несколько особенностей настоящего изобретения может предоставляться, предлагаться, вводиться, управляться, обслуживаться и т.п. поставщиком услуг, который предлагает управление пользовательскими средами. Например, поставщик услуг может создавать, поддерживать, обслуживать и т.п. компьютерный код и/или вычислительную инфраструктуру, которая выполняет одну или несколько особенностей настоящего изобретения для одного или нескольких пользователей. В ответ поставщик услуг может получать оплату от пользователя, например, согласно подписке и/или соглашению. Дополнительно или в качестве альтернативы, поставщик услуг может получать плату за продажу рекламы одной или нескольким третьим сторонам.

Согласно одной из особенностей настоящего изобретения для выполнения одной или нескольких особенностей настоящего изобретения может быть развернуто приложение. В качестве одного из примеров, разворачивание приложения включает предоставление вычислительной инфраструктуры, способной выполнять одну или несколько особенностей настоящего изобретения.

В качестве еще одной особенности настоящего изобретения, разворачивание

вычислительной инфраструктуры может включать интегрирование машиночитаемого кода в вычислительную систему, в которой код в сочетании с вычислительной системой способен выполнять одну или несколько особенностей настоящего изобретения.

В качестве еще одной особенности настоящего изобретения может быть представлен способ интегрирования вычислительной инфраструктуры, включающий интегрирование машиночитаемого кода в компьютерную систему. Компьютерная система содержит машиночитаемую среду, причем машиночитаемая среда содержит одну или несколько особенностей настоящего изобретения. Код в сочетании с компьютерной системой способен выполнять одну или несколько особенностей настоящего изобретения.

Хотя выше описаны различные варианты осуществления, они являются лишь примерами. Например, вычислительные среды других архитектур могут содержать и использовать одну или несколько особенностей настоящего изобретения. В качестве примеров, сервера, отличающиеся от серверов System z®, такие как сервера Power Systems или другие сервера, поставляемые International Business Machines Corporation, или сервера других компаний могут включать, использовать и/или получать преимущества от одной или более особенностей настоящего изобретения. Кроме того, хотя в приведенном в данном документе примере адаптеры и концентратор PCI рассматриваются как часть сервера, в других вариантах осуществления они не обязательно должны рассматриваться как часть сервера, а могут просто рассматриваться как соединенные с системной памятью и/или другими компонентами вычислительной среды. Не требуется, чтобы вычислительная среда была сервером. Кроме того, хотя описаны таблицы, могут использоваться любые структуры данных, и термин “таблица” должен включать все такие структуры данных. Кроме того, хотя адаптеры и основаны на PCI, одна или более особенностей настоящего изобретения могут использоваться с другими адаптерами или другими компонентами ввода-вывода. Адаптер и адаптер PCI являются лишь примерами. Более того, FTE или параметры FTE могут располагаться и поддерживаться не только в защищенной памяти, а, например, в аппаратном обеспечении (например, аппаратном обеспечении функции PCI). DTE, FTE и/или дескриптор могут включать больше или меньше информации, или другую информацию, также как и блок запроса и/или ответа. Дополнительно команда Call Logical Processor может иметь больше или меньше полей, или другие поля. Возможны многие другие изменения.

Кроме того, от одной или нескольких особенностей настоящего изобретения могут получать выгоду и другие типы вычислительных сред. Например, может применяться система обработки данных, подходящая для хранения и/или выполнения программного кода, которая содержит по меньшей мере два процессора, прямо или непрямо соединенных с элементами памяти через системную шину. Элементы памяти содержат, например, локальную память, используемую во время фактического выполнения программного кода, устройство массовой памяти и кэш-память, которая обеспечивает временное хранение по меньшей мере некоторого программного кода с целью уменьшения количества извлечений кода из устройства массовой памяти во время выполнения.

Устройства ввода-вывода или I/O устройства (включая без ограничения клавиатуры, экраны, устройства указания, DASD, ленту, CD диски, DVD диски, флэш-устройства и другие среды памяти, и т.п.) могут подключаться к системе или прямо, или посредством промежуточных контроллеров ввода-вывода. Сетевые адаптеры также могут подключаться к системе, чтобы позволять системе обработки данных соединяться с другими системами обработки данных или удаленным принтерам или запоминающим

устройствам посредством промежуточных частных или общих сетей. Модемы, кабельные модемы и сетевые карты Ethernet являются лишь некоторыми из доступных типов сетевых адаптеров.

ФИГ.8 представляет собой изображение характерных компонентов хост-компьютерной системы 5000 для реализации одной или нескольких особенностей настоящего изобретения. Характерный хост-компьютер 5000 содержит один или несколько ЦП 5001, связанных с памятью 5002 (т.е. центральным запоминающим устройством) компьютера, а также интерфейсы ввода-вывода к запоминающим устройствам 5011 и сети 5010 для связи с другими компьютерами или SAN и т.п. ЦП 5001 совместим с архитектурой, имеющей структурированный набор команд и структурированные функциональные возможности. ЦП 5001 может иметь динамическую трансляцию 5003 адреса (DAT) для преобразования программных адресов (виртуальных адресов) в достоверные адреса памяти. DAT обычно содержит буфер 5007 быстрого преобразования адреса (TLB) для кэширования трансляций, чтобы при последующих доступах к блоку памяти 5002 компьютера не требовалась задержка трансляции адреса. Обычно между памятью 5002 компьютера и процессором 5001 используется кэш 5009. Кэш 5009 может быть иерархическим, имеющим кэш большой емкости, доступный для более чем одного ЦП, и более быстродействующих кэшей (низкого уровня) меньшей емкости между кэшем большой емкости и каждым ЦП. В некоторых осуществлениях кэши низкого уровня разделяются, чтобы предоставлять отдельные кэши низкого уровня для извлечения команд и доступа к данным. В одном варианте осуществления команда извлекается из памяти 5002 блоком 5004 извлечения команд посредством кэша 5009. Команда декодируется в блоке 5006 декодирования команд и отправляется (с другими командами в некоторых вариантах осуществления) в блок или блоки 5008 выполнения команд. Обычно используется несколько блоков 5008 выполнения, например, блок выполнения арифметических команд, блок выполнения с плавающей точкой и блок выполнения команд перехода. Команда выполняется блоком выполнения, который осуществляет доступ к операндам из определяемых командами регистров или памяти по мере необходимости. Если доступ (загрузку или сохранение) к операнду необходимо осуществлять из памяти 5002, блок 5005 загрузки/сохранения обычно осуществляет доступ под управлением выполняемой команды. Команды могут выполняться в аппаратных схемах или во внутреннем микрокоде (аппаратно-программном обеспечении), или с использованием их сочетания.

Как указано, компьютерная система содержит информацию в локальном (или основном) запоминающем устройстве, а также адресные, защитные, контрольные и корректирующие записи. Некоторые особенности адресации включают формат адресов, концепцию адресных пространств, различные типы адресов и способ, которым адрес одного типа транслируется в адрес другого типа. Некоторые из основных запоминающих устройств содержат постоянно назначенные местоположения в запоминающем устройстве. Основное запоминающее устройство обеспечивает систему запоминающим устройством с прямой адресацией и быстрым доступом к данным. В основное запоминающее устройство должны загружаться (из устройств ввода) и данные, и программы, до того как они могут быть обработаны.

Основное запоминающее устройство может содержать одно или несколько буферных запоминающих устройств меньшей емкости более быстрого доступа, иногда называемых кэшами. Кэш обычно физически связан с ЦП или процессором ввода-вывода. Физическая структура и использование различных запоминающих сред, как правило, не видимы программой, за исключением производительности.

Для команд и операндов, хранимых в памяти, могу быть предусмотрены отдельные кэши. Информация в кэше содержится в форме непрерывных байтов на целочисленной границе, называемой блоком кэша или строкой кэша (или для краткости строкой). Одна модель может предоставлять команду EXTRACT CACHE ATTRIBUTE, которая возвращает размер строки кэша в байтах. Одна модель также может предоставлять команду PREFETCH DATA и команду PREFETCH DATA RELATIVE LONG для предварительного извлечения данных из запоминающего устройства в кэш данных или команд или для освобождения данных из кэша.

Запоминающее устройство рассматривается как длинная горизонтальная строка битов. Для большинства операций доступ к запоминающему устройству осуществляется последовательно слева направо. Строка битов подразделяется на блоки из восьми битов. Восьмибитный блок называется байтом, который является основным конструктивным блоком всех форматов представления информации. Местоположение каждого байта в запоминающем устройстве идентифицируется уникальным неотрицательным целым числом, которое является адресом местоположения этого байта или просто адресом байта. Соседние местоположения байтов имеют последовательные адреса, начинающиеся слева с 0 и последовательно следующие слева направо. Адреса представляют собой беззнаковые двоичные целые числа, содержащие 24, 31 или 64 бита.

Информация между запоминающим устройством и ЦП или канальной подсистемой передается по одному байту или группой байтов за один раз. Если не указано иное, например, в z/Architecture®, обращение по адресу к группе байтов в запоминающем устройстве осуществляется посредством крайнего слева байта группы. Число байтов в группе подразумевается или прямо определяется операцией, которую необходимо выполнить. Используемая в работе ЦП группа байтов называется полем. В каждой группе байтов, например, в z/Architecture®, биты нумеруются в последовательности слева направо. В z/Architecture® крайние слева биты иногда называются “старшими” битами, а крайние справа биты - “младшими” битами. Однако номера битов не являются адресами запоминающего устройства. Обращаться по адресу можно только к байтам. Чтобы работать с отдельными битами байта в запоминающем устройстве, осуществляется доступ ко всему байту. Биты в байте нумеруются от 0 до 7, слева направо (например, в z/Architecture®). Биты в адресе могут быть пронумерованы от 8 до 31 или от 40 до 63 в случае 24-битных адресов или от 1 до 31 или от 33 до 63 в случае 31-битных адресов, и от 0 до 63 в случае 64-битных адресов. В любом другом формате фиксированной длины из множества байтов биты, образующие формат, последовательно нумеруются, начиная с 0. В целях обнаружения ошибок, и предпочтительно их исправления, с каждым байтом или группой байтов может передаваться один или несколько контрольных битов. Такие контрольные биты генерируются автоматически машиной и не могут непосредственно управляться программой. Емкость запоминающего устройства выражается в числе байтов. Когда длина хранящегося поля операнда подразумевается кодом операций команды, говорят, что поле имеет фиксированную длину, которая может составлять один, два, четыре, восемь или шестнадцать байтов. Для некоторых команд могут подразумеваться более длинные поля. Когда длина хранящегося поля операнда не подразумевается, а прямо указывается, говорят, что поле имеет переменную длину. Операнды переменной длины могут различаться по длине приращениями в один байт (или для некоторых команд с шагом в два байта и другими шагами). При сохранении информации в запоминающем устройстве замещается содержимое местоположений только тех байтов, которые включены в указанное поле,

несмотря на то, что ширина физического пути к запоминающему устройству может быть больше длины сохраняемого поля.

Некоторые единицы информации должны находиться на целочисленной границе в запоминающем устройстве. Применительно к единице информации граница называется целочисленной, когда адрес запоминающего устройства кратен длине единицы информации в байтах. Полям длиной 2, 4, 8 и 16 байтов на целочисленной границе даются особые названия. Полуслово представляет собой группу из 2 идущих подряд байтов на двухбайтной границе и является основным структурным блоком команд. Слово представляет собой группу из 4 идущих подряд байтов на четырехбайтной границе. Двойное слово представляет собой группу из 8 идущих подряд байтов на восьмибайтной границе. Учетверенное слово представляет собой группу из 16 идущих подряд байтов на 16-байтной границе. Когда адреса запоминающего устройства обозначают полуслова, слова, двойные слова и учетверенные слова, двоичное представление адреса содержит один, два, три или четыре крайних правых нулевых бита, соответственно. Команды должны находиться на двухбайтных целочисленных границах. Хранящиеся операнды большинства команд не содержат требования выравнивания на границах.

В устройствах, в которых реализованы раздельные кэши для команд и операндов данных, могут происходить значительные задержки, если программа сохраняется в строке кэша, из которой впоследствии осуществляется извлечение команд, независимо от того, изменяются ли при сохранении команды, извлечение которых осуществляется впоследствии.

В одном варианте осуществления изобретение может быть реализовано на практике программным обеспечением (иногда называемым лицензионным внутренним кодом, аппаратно-программным обеспечением, микрокодом, милликодом, пикокодом и т.п., любое из чего будет согласоваться с настоящим изобретением). Как показано на ФИГ.8, к программному коду программного обеспечения, которое воплощает настоящее изобретение, обычно получает доступ процессор 5001 хост-системы 5000 из запоминающих устройств 5011 долговременной памяти, таких как накопитель на CD-диске, накопитель на магнитной ленте или накопитель на жестких дисках. Программный код программного обеспечения может быть воплощен в любой из множества известных сред для использования с системой обработки данных, такой как дискета, накопитель на жестких дисках или CD-диск. Код может распределяться в таких средах или может распределяться пользователям из компьютерной памяти 5002 или запоминающего устройства одной компьютерной системы по сети 5010 другим компьютерным системам для использования пользователями таких других систем.

Программный код включает операционную систему, которая управляет функцией и взаимодействием различных компонентов компьютера и одной или нескольких прикладных программ. Обычно программный код подкачивается по частям из запоминающего устройства 5011 в относительно более быстродействующее запоминающее устройство 5002 компьютера, где он доступен для обработки процессором 5001. Методы и способы для воплощения программного кода в памяти, в физических средах и/или распределения программного кода через сети хорошо известны, и более подробно в данном документе обсуждаться не будут. Программный код, созданный и хранящийся в материальной среде (включая без ограничения модули электронной памяти (ОЗУ), флэш-память, компакт-диски (CD), DVD, магнитную ленту и т.п.) часто называют “компьютерным программным продуктом”. Содержащая компьютерный программный продукт среда обычно может считываться устройством

обработки данных предпочтительно в компьютерной системе для выполнения устройством обработки данных.

ФИГ.9 представляет собой изображение характерной рабочей станции или серверной аппаратной системы, в которой настоящее изобретение может быть реализовано на практике. Система 5020, представленная на ФИГ.9, содержит характерную базовую компьютерную систему 5021, такую как персональный компьютер, рабочая станция или сервер, включая необязательные периферийные устройства. Базовая компьютерная система 5021 имеет один или несколько процессоров 5026 и шину, используемую для соединения и связи процессора (-ов) 5026 и других компонентов системы 5021 в соответствии с известными способами. Шина соединяет процессор 5026 с памятью 5025 и долговременным запоминающим устройством 5027, которое может содержать накопитель на жестких дисках (например, включающий любое из магнитного накопителя, CD-диска, DVD-диска и флэш-памяти) или, например, накопитель на магнитной ленте. Система 5021 также может содержать адаптер пользовательского интерфейса, который соединяет микропроцессор 5026 через шину с одним или несколькими интерфейсными устройствами, такими как клавиатура 5024, мышь 5023, принтер/сканнер 5030 и/или другие интерфейсные устройства, которые могут быть любыми устройствами пользовательского интерфейса, такими как сенсорный экран, клавиатура цифрового ввода и т.п. Шина также соединяет устройство 5022 отображения, такое как ЖК-экран или монитор, с микропроцессором 5026 посредством адаптера устройства отображения.

Система 5021 может поддерживать связь с другими компьютерами или компьютерными сетями посредством сетевого адаптера, способного поддерживать связь 5028 с сетью 5029. Примерами сетевых адаптеров являются каналы связи, кольцевая сеть с маркерным доступом, сеть Ethernet или модемы. Альтернативно система 5021 может поддерживать связь с помощью беспроводного интерфейса, такого как карта CDPD (сотовой системы передачи пакетов цифровых данных). Система 5021 может быть связана с другими такими компьютерами в локальной вычислительной сети (LAN) или глобальной вычислительной сети (WAN), или система 5021 может быть клиентом в клиент/серверной распределенной системе с другим компьютером, и т.п. Все эти конфигурации, а также соответствующее коммуникационное аппаратное и программное обеспечение, известны в данной области техники.

ФИГ.10 представляет собой изображение сети 5040 обработки данных, в которой настоящее изобретение может быть реализовано на практике. Сеть 5040 обработки данных может содержать множество отдельных сетей, таких как беспроводная сеть и проводная сеть, каждая из которых может содержать множество отдельных рабочих станций 5041, 5042, 5043, 5044. Кроме того, как известно специалистам в данной области техники, она может содержать одну или несколько сетей LAN, где LAN может содержать множество интеллектуальных рабочих станций, соединенных с хост-процессором.

Также согласно ФИГ.10 сети также могут содержать мэйнфреймы или серверы, такие как шлюзовый компьютер (клиент-сервер 5046) или сервер приложений (удаленный сервер 5048, который может осуществлять доступ к хранилищу данных, а также может быть доступен непосредственно с рабочей станции 5045). Шлюзовый компьютер 5046 служит точкой входа в каждую отдельную сеть. Шлюз необходим при соединении одного сетевого протокола с другим. Шлюз 5046 предпочтительно может быть соединен линией связи с другой сетью (например, сетью Интернет 5047). Шлюз 5046 также может быть прямо соединен с одной или несколькими рабочими станциями 5041, 5042, 5043, 5044 посредством линии связи. Компьютер-шлюз может быть реализован с помощью

сервера IBM eServer™ System z®, поставляемого International Business Machines Corporation.

Одновременно на ФИГ.9 и ФИГ.10 представлен программный код программного обеспечения, который может реализовывать настоящее изобретение, может быть доступен процессору 5026 системы 5020 из долговременных запоминающих сред 5027, таких как накопитель на CD-дисках или накопитель на жестких дисках. Программный код программного обеспечения может быть воплощен в любой из множества известных сред для использования с системой обработки данных, такой как дискета, жесткий диск или CD-диск. Код может распределяться в таких средах или может распределяться пользователям 5050, 5051 из памяти или запоминающего устройства одной компьютерной системы по сети другим компьютерным системам для использования пользователями таких других систем.

Альтернативно программный код может быть осуществлен в памяти 5025 с доступом к нему процессором 5026 с применением процессорной шины. Такой программный код содержит операционную систему, которая управляет функцией и взаимодействием различных компонентов компьютера и одной или нескольких прикладных программ 5032. Программный код обычно подкачивается по частям из запоминающих сред 5027 в быстродействующую память 5025, где он доступен для обработки процессором 5026. Методы и способы осуществления программного кода программного обеспечения в памяти, в физических средах и/или распределения кода программного обеспечения посредством сетей хорошо известны, и в данном документе подробнее обсуждаться не будут. Программный код, созданный и хранящийся в материальной среде (включая без ограничения модули электронной памяти (ОЗУ), флэш-память, компакт-диски (CD), DVD-диски, магнитную ленту и т.п.) часто называют “компьютерным программным продуктом”. Содержащая компьютерный программный продукт среда обычно может считываться устройством обработки данных предпочтительно в компьютерной системе для выполнения устройством обработки данных.

Кэш, который является наиболее легкодоступным для процессора (обычно более быстродействующий и менее объемный, чем другие кэши процессора), представляет собой низший (L1 или уровень один) кэш, а основное запоминающее устройство (основная память) представляет собой кэш высшего уровня (L3, если имеется 3 уровня). Кэш низшего уровня часто делится на кэш команд (I-Cache), хранящий машинные команды, подлежащие выполнению, и кэш данных (D-Cache), содержащий операнды, хранимые в памяти.

ФИГ.11 представляет собой изображение иллюстративного варианта осуществления процессора для процессора 5026. Обычно, чтобы буферизировать блоки памяти с целью повышения производительности процессора, используется один или несколько уровней кэша 5053. Кэш 5053 представляет собой высокоскоростной буфер, хранящий строки кэша данных памяти, которые предположительно будут использоваться. Типичные строки кэша содержат 64, 128 или 256 байтов данных памяти. Отдельные кэши используются часто для кэширования команд, а не кэширования данных. Согласованность кэшей (синхронизация копий строк в памяти и в кэшах) часто обеспечивается различными алгоритмами слежения, хорошо известными в данной области техники. Основное запоминающее устройство 5025 процессорной системы часто называют кэшем. В процессорной системе, содержащей 4 уровня кэша 5053, основное запоминающее устройство 5025 иногда называют кэшем 5 уровня (L5), поскольку оно обычно является более быстродействующими и содержит лишь часть энергонезависимого запоминающего устройства (DASD, лента и т.д.), которое доступно для компьютерной системы. Основное запоминающее устройство 5025 “кэширует”

страницы данных, которые постранично перемещаются в основное запоминающее устройство 5025 и из него операционной системой.

Программный счетчик (счетчик команд) 5061 отслеживает адрес текущей команды, подлежащей выполнению. Программный счетчик в процессоре z/Architecture® содержит 64 бит, и может быть усечен до 31 или 24 бит, чтобы поддерживать предшествующие адресные ограничения. Счетчик команд обычно воплощен в PSW (слове статуса программы) компьютера, так что он сохраняется при переключении контекста. Таким образом, работающая программа, имеющая значение счетчика команд, может прерываться, например, операционной системой (при переключении контекста из программной среды в среду операционной системы). PSW программы сохраняет значение счетчика команд, пока программа неактивна, и программный счетчик (в PSW) операционной системы используется, пока выполняется операционная система. Обычно счетчик команд увеличивается на величину, равную числу байтов текущей команды. RISC-команды (вычисления с сокращенным набором команд) обычно имеют фиксированную длину, тогда как CISC-команды (вычисления со сложным набором команд) обычно имеют переменную длину. Команды IBM z/Architecture® являются CISC-командами, имеющими длину 2, 4 или 6 байт. Счетчик 5061 команд изменяется, например, или операцией переключения контекста, или операцией выбранного перехода команды перехода. В операции переключения контекста текущее значение счетчика команд сохраняется в слове статуса программы вместе с другой информацией о состоянии выполняемой программы (такой как коды состояний), и загружается новое значение счетчика команд, указывающее на команду нового программного модуля, который необходимо выполнить. Операция выбранного перехода выполняется, чтобы позволить программе принимать решения или выполнять цикл в программе путем загрузки результата команды перехода в счетчик 5061 команд.

Обычно для извлечения команд от имени процессора 5026 используется блок 5055 извлечения команд. Блок извлечения или извлекает “следующие последовательные команды”, целевые команды команд выбранного перехода, или первые команды программы, следующей за переключением контекста. В современных блоках извлечения команд часто применяют методы предварительного извлечения, чтобы эмпирически осуществлять предварительное извлечение команд на основании вероятности того, что предварительно извлеченные команды могли бы быть использованы. Например, блок извлечения может осуществлять извлечение 16 байтов команды, которая содержит следующую последовательную команду, и дополнительных байтов других последовательных команд.

Затем извлеченные команды выполняются процессором 5026. В одном варианте осуществления извлеченная команда(-ы) передается блоку 5056 диспетчеризации блока извлечения. Блок диспетчеризации декодирует команду(-ы) и пересылает информацию о декодированной команде(-ах) соответствующим блокам 5057, 5058, 5060. Блок 5057 выполнения обычно принимает информацию о декодированных арифметических командах от блока 5055 извлечения команд и выполняет арифметические операции с операндами в соответствии с кодом операции команды. Операнды подаются на блок 5057 выполнения предпочтительно или из памяти 5025, структурированных регистров 5059 или из ближайшего поля выполняемой команды. Сохраненные результаты выполнения хранятся или в памяти 5025, регистрах 5059 или в другом машинном аппаратном обеспечении (таком как управляющие регистры, регистры PSW и т.п.).

Процессор 5026 обычно имеет один или несколько блоков 5057, 5058, 5060 для выполнения функции команды. Согласно ФИГ.12А блок 5057 выполнения может

сообщаться со структурированными регистрами 5059 общего назначения, блоком 5056 декодирования/диспетчеризации, блоком 5060 загрузки/сохранения и другими процессорными блоками 5065 посредством интерфейсной логики 5071. Блок 5057 выполнения может применять несколько регистровых схем 5067, 5068, 5069, чтобы хранить информацию, с которой будет работать арифметическое логическое устройство (ALU) 5066. ALU выполняет арифметические операции, такие как сложение, вычитание, умножение и деление, а также логические функции, такие как И, ИЛИ и исключающее ИЛИ (XOR), циклический сдвиг и смещение. ALU предпочтительно поддерживает специализированные операции, зависящие от структуры. Другие схемы могут обеспечивать другие структурированные средства 5072, содержащие, например, коды ситуаций и логику поддержки восстановления.

Обычно результат операции ALU хранится в схеме 5070 выходного регистра, которая может направлять результат ряду других функций обработки. Хотя существует множество конструкций процессоров, настоящее описание предназначено лишь обеспечить характерное понимание одного варианта осуществления.

Например, команда ADD выполняется блоком 5057 выполнения, обладающим арифметическими и логическими функциональными возможностями, тогда как, например, команда с плавающей точкой выполняется блоком выполнения с плавающей точкой, обладающим специальными возможностями работы с плавающей точкой. Блок выполнения предпочтительно работает с указанными командой операндами путем выполнения функции, определенной кодом операции, на операндах. Например, команда ADD может выполняться блоком 5057 выполнения на операндах, находящихся в двух регистрах 5059, указанных в регистровых полях команды.

Блок 5057 выполнения выполняет арифметическое сложение двух операндов и сохраняет результат в третьем операнде, где третий операнд может быть третьим регистром или одним из двух исходных регистров. Блок выполнения предпочтительно использует арифметическое логическое устройство (ALU) 5066, способное выполнять ряд логических функций, таких как смещение, циклический сдвиг, И, ИЛИ и исключающее ИЛИ, а также ряд алгебраических функций, включая любую из сложения, вычитания, умножения, деления. Некоторые ALU 5066 разработаны для скалярных операций, а некоторые - для плавающей точки. В зависимости от архитектуры данные могут иметь обратный порядок следования байтов (когда наименьший значимый байт соответствует старшему байтовому адресу) или прямой порядок следования байтов (когда наименьший значимый байт соответствует младшему байтовому адресу). IBM z/Architecture® имеет обратный порядок следования байтов. В зависимости от архитектуры поля чисел со знаком могут быть представлены в виде прямого кода, дополнения до 1 или дополнения до 2. Число в форме дополнения до 2 выгодно в том смысле, что ALU не нужно поддерживать процедуру вычитания, поскольку и отрицательное, и положительное значение в форме дополнения до 2 в ALU требует только сложения. Числа обычно описаны в сокращенном виде, в котором 12-битное поле определяет адрес блока из 4096 байтов и обычно описывается, например, как 4 кбайтный (килобайтный) блок.

Согласно ФИГ.12Б, информация команды перехода для выполнения команды перехода, как правило, отправляется на блок 5058 перехода, который часто использует алгоритм предсказания перехода, такой как таблица 5082 истории переходов, чтобы предсказывать исход перехода до завершения других условных операций. Цель текущей команды перехода извлекается и выполняется по предположению до завершения условных операций. Когда условные операции завершаются, выполненные по

предположению команды перехода или завершаются, или отбрасываются на основании ситуаций условной операции и предположенного исхода. Типичная команда перехода может проверять коды ситуаций и переход к целевому адресу, если коды ситуаций отвечают требованию перехода команды перехода, причем целевой адрес может рассчитываться на основании нескольких чисел, включая, например, числа из полей регистра или непосредственного поля команды. Блок 5058 перехода может применять ALU 5074, имеющее множество схем 5075, 5076, 5077 входных регистров и схему 5080 выходного регистра. Блок 5058 перехода может поддерживать связь, например, с регистрами 5059 общего назначения, блоком 5056 декодирования/диспетчеризации или другими схемами 5073.

Выполнение группы команд может прерываться по ряду причин, включая, например, переключение контекста, инициированное операционной системой, исключительную ситуацию или ошибку программы, приводящую к переключению контекста, сигнал прерывания ввода-вывода, приводящий к переключению контекста, или многопоточную деятельность множества программ (в многопоточной среде). Предпочтительно переключение контекста сохраняет информацию о состоянии выполняемой в данный момент программы, а затем загружает информацию о состоянии другой запускаемой программы. Информация о состоянии может сохраняться, например, в аппаратных регистрах или в памяти. Информация о состоянии предпочтительно содержит значение счетчика команд, указывающее на следующую команду, которую необходимо выполнить, коды ситуаций, информацию о трансляции памяти и содержимое структурированного регистра. Деятельность по переключению контекста может осуществляться аппаратными схемами, прикладными программами, программами операционной системы или аппаратно-программным кодом (микрокодом, пикокодом или лицензионным внутренним кодом (LIC) по отдельности или в сочетании).

Процессор осуществляет доступ к операндам в соответствии с определенными командами способами. Команда может предоставлять непосредственный операнд, использующий значение части команды, может предоставлять одно или несколько регистровых полей, прямо указывающих или на регистры общего назначения, или на регистры особого назначения (например, регистры с плавающей точкой). Команда может использовать подразумеваемые регистры, идентифицируемые полем кода операции как операнды. Команда может использовать для операндов ячейки памяти. Местоположение в памяти операнда может предоставляться регистром, непосредственным полем или сочетанием регистров и непосредственного поля, пример чего дает средство длинного смещения z/Architecture®, где команда определяет базовый регистр, индексный регистр и непосредственное поле (поле смещения), которые складываются вместе, чтобы предоставлять, например, адрес операнда в памяти. Под ячейкой в данном документе понимается местоположение в основной памяти (основном запоминающем устройстве), если не указано иное.

Согласно ФИГ.12В, процессор осуществляет доступ к запоминающему устройству с помощью блока 5060 загрузки/сохранения. Блок 5060 загрузки/сохранения может выполнять операцию загрузки путем получения адреса целевого операнда в памяти 5053 и загрузки операнда в регистр 5059 или другую ячейку памяти 5053, или может выполнять операцию сохранения путем получения адреса целевого операнда в памяти 5053 и сохранения данных, полученных из регистра 5059 или другой ячейки памяти 5053, в ячейке целевого операнда в памяти 5053. Блок 5060 загрузки/сохранения может действовать по предположению и осуществлять доступ к памяти в последовательности, которая по порядку не соответствует последовательности команд, однако блок 5060

загрузки/сохранения должен поддерживать для программ видимость того, что команды выполнялись по порядку. Блок 5060 загрузки/сохранения может поддерживать связь с регистрами 5059 общего назначения, блоком 5056 декодирования/диспетчеризации, интерфейсом 5053 кэша/памяти или другими элементами 5083 и содержит различные регистровые схемы, устройства ALU 5085 и управляющую логику 5090, чтобы вычислять адреса памяти и обеспечивать формирование последовательности конвейера для сохранения порядка операций. Некоторые операции могут не сохранять порядок, но блок загрузки/сохранения обеспечивает функциональные возможности для того, чтобы операции с нарушенным порядком казались программе выполненными по порядку, как хорошо известно в данной области техники.

Предпочтительно адреса, которые "видит" прикладная программа, часто называют виртуальными адресами. Виртуальные адреса иногда называют "логическими адресами" и "исполнительными адресами". Эти виртуальные адреса являются виртуальными в том смысле, что они перенаправляются в местоположение в физической памяти посредством одной из ряда технологий динамической трансляции адреса (DAT), включая без ограничения простую префиксацию виртуального адреса величиной сдвига, трансляцию виртуального адреса посредством одной или нескольких таблиц трансляции, причем таблицы трансляции предпочтительно содержат, по меньшей мере, таблицу сегментов и таблицу страниц по отдельности или в сочетании, предпочтительно таблицу сегментов, содержащую запись, указывающую на таблицу страниц. В z/Architecture® предоставляется иерархия трансляции, включающая первую таблицу областей, вторую таблицу областей, третью таблицу областей, таблицу сегментов и необязательную таблицу страниц. Эффективность трансляции адресов часто повышается за счет использования буфера (TLB) ассоциативной трансляции, который содержит записи, отображающие виртуальный адрес для соответствующего местоположения в физической памяти. Записи создаются, когда DAT транслирует виртуальный адрес с помощью таблиц трансляции. Последующее использование виртуального адреса может затем использовать запись быстродействующего TLB, а не медленный последовательный доступ к таблице трансляции. Содержимым TLB может управлять ряд алгоритмов замещения, включая алгоритм замещения, включая LRU (наиболее давней по использованию).

В том случае, когда процессор является процессором многопроцессорной системы, каждый процессор отвечает за сохранение совместно используемых ресурсов, таких как средства ввода-вывода, кэши, TLB и память, взаимно заблокированных для обеспечения непротиворечивости. Обычно для поддержания непротиворечивости кэшей используются технологии "слежения". Чтобы облегчать совместное использование, в среде слежения каждая строка кэша может отмечаться как находящаяся в любом состоянии из состояния совместного использования, состояния монопольного использования, измененного состояния, недостоверного состояния и т.п.

Блоки 5054 ввода-вывода (ФИГ.11) обеспечивают процессор средствами подключения к периферийным устройствам, включая, например, накопители на магнитной ленте, диски, принтеры, устройства отображения и сети. Блоки ввода-вывода часто представляются компьютерной программе драйверами программного обеспечения. В мэйнфреймах, таких как System z® от IBM®, канальные адаптеры и адаптеры открытых систем являются элементами ввода-вывода мэйнфрейма, которые обеспечивают связь между операционной системой и периферийными устройствами.

Кроме того, от одной или более особенностей настоящего изобретения могут получать выгоду и другие типы вычислительных сред. Например, среда может содержать

эмулятор (например, программные или другие механизмы эмуляции), в котором эмулируется конкретная архитектура (включая, например, выполнение команд, структурированные функции, такие как трансляция адреса, и структурированные регистры) или ее подмножество (например, в частной компьютерной системе, имеющей процессор и память). В такой среде одна или несколько функций эмуляции эмулятора могут осуществлять одну или несколько особенностей настоящего изобретения, даже если компьютер, на котором работает эмулятор, может иметь архитектуру, отличающуюся от эмулируемых средств. Например, в режиме эмуляции декодируется конкретная эмулируемая команда или операция, и для реализации отдельной команды или операции создается соответствующая функция эмуляции.

В эмулирующей среде хост-компьютер содержит, например, память для хранения команд и данных; блок извлечения команд для извлечения команд из памяти и, необязательно, обеспечения локальной буферизации извлеченной команды; блок декодирования команд для приема извлеченных команд и определения типа команд, которые были извлечены; и блок выполнения команд для выполнения команд. Выполнение может включать загрузку данных из памяти в регистр; сохранение данных обратно в память из регистра; или выполнение арифметической или логической операции некоторого типа, как определяется блоком декодирования. В одном примере каждый блок реализован в программном обеспечении. Например, выполняемые блоками операции реализуются в виде одной или нескольких подпрограмм в программном обеспечении эмулятора.

Более конкретно, в мэйнфрейме структурированные машинные команды используются программистами, сегодня, как правило, программистами, работающими на языке C, часто посредством приложения-компиллятора. Эти команды, хранящиеся в запоминающей среде, могут выполняться в родном формате в z/Architecture® IBM® Server, или в другом случае в машинах, реализующих другие архитектуры. Они могут быть эмулированы на существующих и будущих мэйнфрейм-серверах IBM® и на других машинах IBM® (например, серверах Power Systems и серверах System x®). Они могут выполняться на машинах под управлением Linux, на широком ряде машин, использующих аппаратное обеспечение производства IBM®, Intel®, AMD™ и других. Кроме выполнения на этом аппаратном обеспечении под z/Architecture®, может использоваться Linux, также как и машины, которые используют эмуляцию посредством Hercules или FSI (Fundamental Software, Inc), где, как правило, выполнение происходит в режиме эмуляции. В режиме эмуляции программное обеспечение эмуляции выполняется собственным процессором, чтобы эмулировать архитектуру эмулируемого процессора. Информация о вышеуказанных продуктах эмуляции доступна в World Wide Web, соответственно, на www.hercules-390.org и www.funsoft.com.

Собственный процессор, чтобы выполнять эмуляцию эмулируемого процессора, как правило, выполняет программное обеспечение эмуляции, содержащее или аппаратно-программное обеспечение или собственную операционную систему. Программное обеспечение эмуляции отвечает за извлечение и выполнение команд архитектуры эмулируемого процессора. Программное обеспечение эмуляции поддерживает счетчик эмулируемых программ, чтобы отслеживать границы команд. Программное обеспечение эмуляции может извлекать одну или несколько эмулируемых машинных команд за раз и преобразовывать одну или несколько эмулируемых машинных команд в соответствующую группу собственных машинных команд для выполнения собственным процессором. Эти преобразованные команды могут кэшироваться, так что можно добиться более быстрого преобразования. Тем не менее, программное обеспечение

эмуляции должно поддерживать правила архитектуры эмулируемого процессора, чтобы обеспечивать корректную работу операционных систем и приложений, написанных для эмулируемого процессора. Кроме того, программное обеспечение эмуляции должно обеспечивать ресурсы, определенные архитектурой эмулируемого процессора, включая без ограничения управляющие регистры, регистры общего назначения, регистры с плавающей точкой, функцию динамической трансляции адреса, включая, например, таблицы сегментов и таблицы страниц, механизмы прерывания, механизмы переключения контекста, часы истинного времени (TOD) и структурированные интерфейсы к подсистемам ввода-вывода, чтобы операционная система или прикладная программа, созданная для работы на эмулируемом процессоре, могла запускаться на собственном процессоре, имеющем программное обеспечение эмуляции.

Конкретная эмулируемая команда декодируется, и для выполнения функции отдельной команды вызывается подпрограмма. Функция программного обеспечения эмуляции, эмулирующая функцию эмулируемого процессора, реализуется, например, в подпрограмме или драйвере на языке C или каким-либо другим способом предоставления драйвера для конкретного аппаратного обеспечения, который будет доступен специалистам в данной области техники после понимания описания предпочтительного варианта осуществления. Различные патенты, относящиеся к эмуляции программного и аппаратного обеспечения, включая, без ограничения, патент США на изобретение №5551013 под названием “Multiprocessor for Hardware Emulation”, на имя Beausoleil и др.; и патент США на изобретение №6009261 под названием “Preprocessing of Stored Target Routines for Emulating Incompatible Instructions on a Target Processor”, на имя Scaizi и др.; и патент США на изобретение №5574873 под названием “Decoding Guest Instruction to Directly Access Emulation Routines that Emulate the Guest Instructions”, на имя Davidian и др.; и патент США на изобретение №6308255 под названием “Symmetrical Multiprocessing Bus and Chipset Used for Coprocessor Support Allowing Non-Native Code to Run in a System”, на имя Gorishek и др.; и патент США на изобретение №6463582 под названием “Dynamic Optimizing Object Code Translator for Architecture Emulation and Dynamic Optimizing Object Code Translation Method”, на имя Lethin и др.; и патент США на изобретение №5790825 под названием “Method for Emulating Guest Instructions on a Host Computer Through Dynamic Recompilation of Host Instructions”, на имя Eric Traut; и многие другие представляют ряд известных способов осуществления эмуляции формата команд управления, разработанного для другой машины, для целевой машины, доступной специалисту в данной области техники.

ФИГ.13 представляет собой изображение примера эмулируемой компьютерной хост-системы 5092, которая эмулирует компьютерную хост-систему 5000' хост-архитектуры. В эмулируемой компьютерной хост-системе 5092 хост-процессор (ЦП) 5091 является эмулируемым хост-процессором (или виртуальным хост-процессором) и содержит процессор 5093 эмуляции, имеющий другую собственную архитектуру набора команд, чем у процессора 5091 хост-компьютера 5000'. Эмулируемая компьютерная хост-система 5092 содержит память 5094, доступную для процессора 5093 эмуляции. В иллюстративном варианте осуществления память 5094 разделена на часть памяти 5096 хост-компьютера и часть 5097 программ эмуляции. Память 5096 хост-компьютера доступна для программ эмулируемого хост-компьютера 5092 в соответствии с архитектурой хост-компьютера. Процессор 5093 эмуляции выполняет собственные команды структурированного набора команд, имеющих структуру, отличную от структуры команд эмулируемого процессора 5091, собственные команды извлекаются из памяти 5097 подпрограмм эмуляции, и может осуществлять доступ к хост-команде для выполнения из программы в памяти

5096 хост-компьютера путем применения одной или нескольких команд, полученных в подпрограмме последовательности и доступа/декодирования, которая может декодировать хост-команду(-ы), к которой осуществляется доступ, чтобы определять подпрограмму выполнения собственных команд для эмуляции функции хост-команды, к которой осуществляется доступ. Другие средства, которые определены для архитектуры компьютерной хост-системы 5000', могут эмулироваться подпрограммами структурированных средств, включая такие средства, как, например, регистры общего назначения, управляющие регистры, поддержка подсистемы динамического преобразования адреса и ввода-вывода и кэш процессора. Подпрограммы эмуляции также могут получать преимущество от функций, доступных в процессоре 5093 эмуляции (таких как регистры общего назначения и динамическая трансляция виртуальных адресов), чтобы улучшать производительность подпрограмм эмуляции. Также может предоставляться специальное аппаратное обеспечение и механизмы разгрузки, чтобы помогать процессору 5093 эмулировать функции хост-компьютера 5000'.

Используемая в данном документе терминология имеет целью описание лишь конкретных вариантов осуществления и не предназначена ограничивать изобретение. Как используются в данном документе, формы единственного числа также включают формы множественного числа, если иное прямо не следует из контекста. Также будет понятно, что термины “содержит” и/или “содержащий”, когда используются в этом описании, определяют наличие указанных признаков, чисел, этапов, операций, элементов и/или компонентов, но не исключают наличия или добавления одного или нескольких других признаков, чисел, шагов, операций, элементов, компонентов и/или их групп.

Подразумевается, что соответствующие структуры, материалы, действия и эквиваленты всех элементов “средство или этап плюс функция” в предоставленной ниже формуле изобретения, если таковые имеются, включают любую структуру, материал или действие для выполнения функции в сочетании с другими заявленными элементами, как заявлено конкретным образом. Описание настоящего изобретения было представлено с целью демонстрации и описания, и не подразумевается полным или ограниченным изобретением в раскрытой форме. Множество модификаций и изменений будут очевидны специалисту обыкновенной квалификации в данной области техники, без отхода от объема изобретения. Вариант осуществления был выбран и описан для того, чтобы наилучшим образом объяснить принципы изобретения и применение на практике, и чтобы позволить другим специалистам обыкновенной квалификации в данной области техники понять изобретение для различных вариантов осуществления с различными модификациями, которые подходят для конкретного рассматриваемого применения.

## Формула изобретения

1. Способ активации адаптеров в вычислительной среде, характеризующийся тем, что:

в ответ на выполнение команды Call Logical Processor (CLP), выданной операционной системой, для активации адаптера, выбранного операционной системой, и содержащей дескриптор функции, идентифицирующий адаптер и имеющий указатель неактивированного адаптера, команда CLP запрашивает число адресных пространств прямого доступа к памяти (DMA), которые должны быть назначены адаптеру, причем указанное выполнение активирует одно или более адресных пространств DMA и включает:

а) активацию адаптера, включающую активацию регистрации для трансляции адреса

и прерываний для поддержки прямых доступов к памяти и инициируемых сообщениями прерываний для адаптера, определение того, что запрошенное число адресных пространств DMA является доступным, выполняемое путем проверки наличия записей таблицы устройств для запрошенного числа адресных пространств DMA, и назначение адаптеру числа записей таблицы устройств, соответствующего запрошенному числу адресных пространств DMA; и

б) возврат дескриптора функции, имеющего указатель активированного адаптера.

2. Способ по п. 1, отличающийся тем, что активация включает активацию одной или более команд, которые должны быть поданы адаптеру.

3. Способ по п. 1, отличающийся тем, что число адресных пространств, которые должны быть назначены, указывают в блоке запроса команды CLP.

4. Способ по п. 1, отличающийся тем, что активация дополнительно включает:

применение дескриптора функции адаптера для определения положения записи таблицы функций, связанной с адаптером; и

применение информации в записи таблицы функций для определения того, должен ли быть активирован адаптер, и причем активация назначает одну или более записей таблицы устройств адаптеру с учетом определения того, что адаптер должен быть активирован.

5. Способ по п. 4, отличающийся тем, что дескриптор функции связан с записью таблицы функций и содержит номер функции и номер экземпляра, и при этом способ дополнительно включает определение достоверности дескриптора, причем определение включает:

проверку того, что указатель отсутствия активации указывает на отсутствие активации; и

проверку того, что номер функции обозначает установленную функцию, причем применение выполняют с учетом определения достоверного дескриптора.

6. Способ по п. 4, отличающийся тем, что применение включает проверку по меньшей мере одного из указателя отсутствия активации, указателя состояния постоянной ошибки, указателя инициированного восстановления после ошибки, указателя занятости или указателя разрешения в записи таблицы функций, для определения того, должен ли быть активирован адаптер.

7. Способ по п. 1, отличающийся тем, что активация дополнительно включает связывание одной или более записей таблицы устройств с записью таблицы функций, связанной с адаптером, причем запись таблицы функций предоставляет информацию об адаптере.

8. Способ по п. 7, отличающийся тем, что запись таблицы функций связана с дескриптором функции, и при этом активация дополнительно включает выполнение по меньшей мере одного из:

установки указателя активации функции в записи таблицы функций для указания активированного состояния;

установки одного или более указателей активации устройства в одной или более записях таблицы устройств для указания активированного состояния;

включения в ассоциативное запоминающее устройство одного или более индексов для одной или более записей таблицы устройств, причем ассоциативное запоминающее устройство применяют при определении положения записи таблицы устройств с учетом запроса от адаптера;

установки указателя активированного адаптера в дескрипторе функции для указания активированного состояния; и

обновления номера экземпляра дескриптора функции.

9. Способ по п. 1, отличающийся тем, что способ дополнительно включает деактивацию адаптера.

10. Способ по п. 9, отличающийся тем, что деактивация включает:

применение дескриптора функции для определения положения записи таблицы функций, связанной с адаптером; и

применение информации в записи таблицы функций для определения того, должен ли быть деактивирован адаптер, и продолжение деактивации с учетом определения того, что адаптер должен быть деактивирован.

11. Способ по п. 10, отличающийся тем, что продолжение деактивации включает по меньшей мере одно из:

установки указателя активации функции в записи таблицы функций в деактивированное состояние;

сброса и освобождения одной или более записей таблицы устройств, связанных с адаптером; и

установки указателя отсутствия активации дескриптора функции, для указания деактивированного состояния.

12. Способ по п. 10, отличающийся тем, что деактивация дополнительно включает определение достоверности дескриптора функции, при этом определение включает:

проверку того, что указатель активации адаптера установлен в активированное состояние; и

проверку того, что дескриптор указывает на достоверную запись в таблице функций, причем применение выполняют с учетом определения того, что дескриптор функции достоверный.

13. Способ по п. 12, отличающийся тем, что применение включает сравнение номера экземпляра в дескрипторе функции с номером экземпляра в записи таблицы функций, причем продолжение деактивации происходит с учетом того, что сравнение указывает на равенство.

14. Способ по п. 1, отличающийся тем, что адаптер включает функцию взаимосвязи периферийных компонентов (PCI).

15. Компьютерная система для активации адаптеров в вычислительной среде, содержащая:

память;

процессор, связанный с памятью;

элемент запросов, реагирующий на выполнение команды Call Logical Processor (CLP), выданной операционной системой, для активации адаптера, выбранного операционной системой, и содержащей дескриптор функции, идентифицирующий адаптер и имеющий указатель неактивированного адаптера, причем команда CLP запрашивает число адресных пространств прямого доступа к памяти (DMA), которые должны быть назначены адаптеру, а указанное выполнение активирует одно или более адресных пространств DMA и включает:

а) активацию адаптера, включающую активацию регистрации для трансляции адреса и прерываний для поддержки прямых доступов к памяти и инициируемых сообщениями прерываний для адаптера, определение того, что запрошенное число адресных пространств DMA является доступным, выполняемое путем проверки наличия записей таблицы устройств для запрошенного числа адресных пространств DMA, и назначение адаптеру числа записей таблицы устройств, соответствующего запрошенному числу адресных пространств DMA; и

б) возврат дескриптора функции, имеющего указатель активированного адаптера.

16. Компьютерная система по п. 15, отличающаяся тем, что активация включает активацию одной или более команд, которые должны быть поданы адаптеру.

17. Компьютерная система по п. 15, отличающаяся тем, что активация дополнительно включает:

применение дескриптора функции адаптера для определения положения записи таблицы функций, связанной с адаптером; и

применение информации в записи таблицы функций для определения того, должен ли быть активирован адаптер, и причем активация назначает одну или более записей таблицы устройств адаптеру с учетом определения того, что адаптер должен быть активирован.

18. Компьютерная система по п. 15, отличающаяся тем, что активация дополнительно включает связывание одной или более записей таблицы устройств с записью таблицы функций, связанной с адаптером, причем запись таблицы функций предоставляет информацию об адаптере.

19. Компьютерная система по п. 18, отличающаяся тем, что запись таблицы функций связана с дескриптором функции, и при этом активация дополнительно включает выполнение по меньшей мере одного из:

установки указателя активации функции в записи таблицы функций для указания активированного состояния;

установки одного или более указателей активации устройства в одной или более записях таблицы устройств для указания активированного состояния;

включения в ассоциативное запоминающее устройство одного или более индексов для одной или более записей таблицы устройств, причем ассоциативное запоминающее устройство применяют при определении положения записи таблицы устройств с учетом запроса от адаптера;

установки указателя активированного адаптера в дескрипторе функции для указания активированного состояния; и

обновления номера экземпляра дескриптора функции.

20. Компьютерная система по п. 15, отличающаяся тем, что дополнительно предусмотрена деактивация адаптера, включающая:

применение дескриптора функции для определения положения записи таблицы функций, связанной с адаптером; и

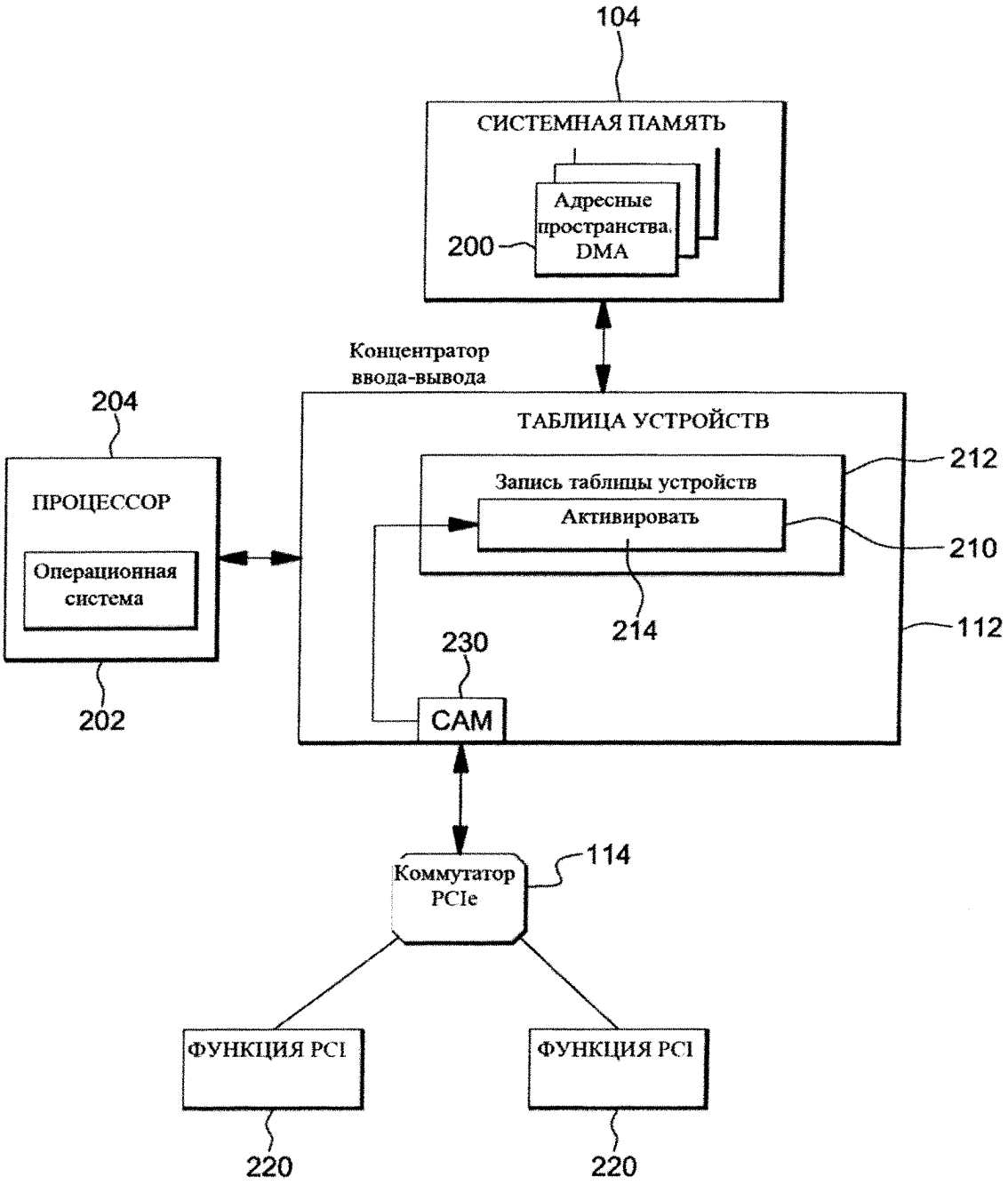
применение информации в записи таблицы функций для определения того, должен ли быть деактивирован адаптер, и продолжение деактивации с учетом определения того, что адаптер должен быть деактивирован.

21. Компьютерная система по п. 20, отличающаяся тем, что продолжение деактивации включает по меньшей мере одно из:

установки указателя активации функции в записи таблицы функций в деактивированное состояние;

сброса и освобождения одной или более записей таблицы устройств, связанных с адаптером; и

установки указателя отсутствия активации дескриптора функции для указания деактивированного состояния.


104
СИСТЕМНАЯ ПАМЯТЬ
Адресные пространства DMA
200
Концентратор ввода-вывода
204
ТАБЛИЦА УСТРОЙСТВ
212
Запись таблицы устройств
Активировать
210
ПРОЦЕССОР
Операционная система
214
112
230
CAM
202
Коммутатор PCIe
114
ФУНКЦИЯ PCI
ФУНКЦИЯ PCI
220
220

**ФИГ. 2**

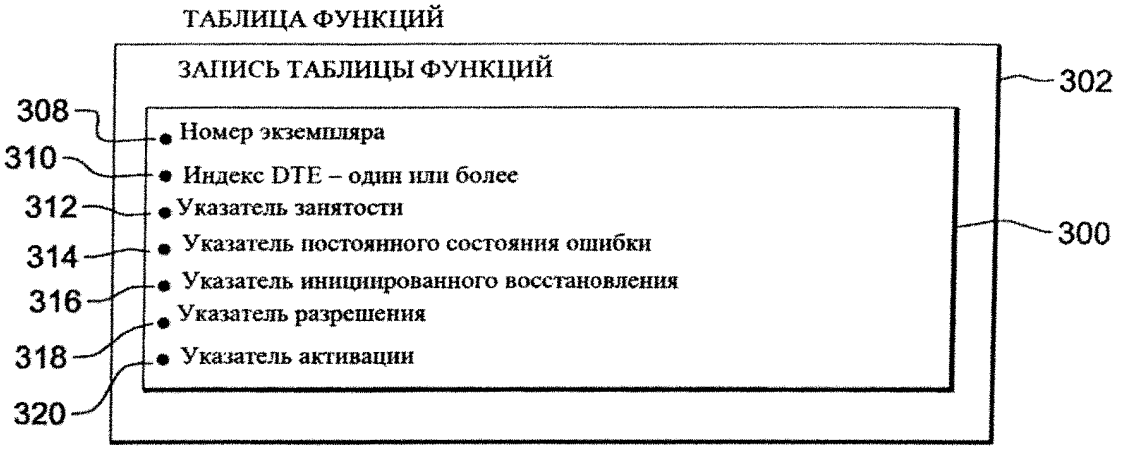
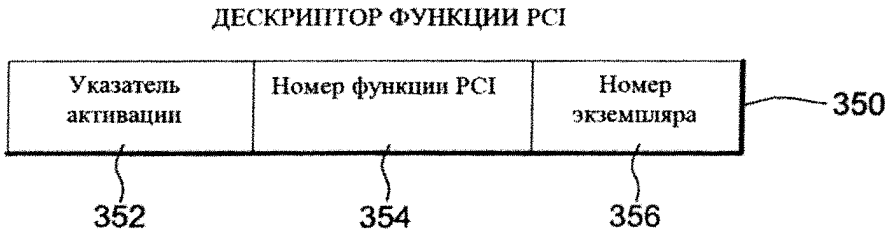
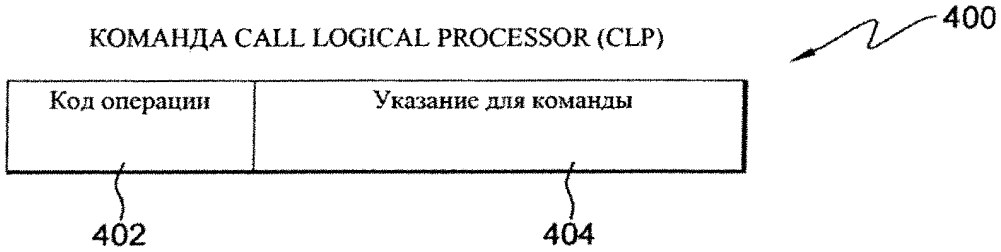


ТАБЛИЦА ФУНКЦИЙ
ЗАПИСЬ ТАБЛИЦЫ ФУНКЦИЙ
302
308
Номер экземпляра
310
Индекс DTE – один или более
312
Указатель занятости
314
Указатель постоянного состояния ошибки
316
Указатель инициированного восстановления
Указатель разрешения
318
Указатель активации
320
300

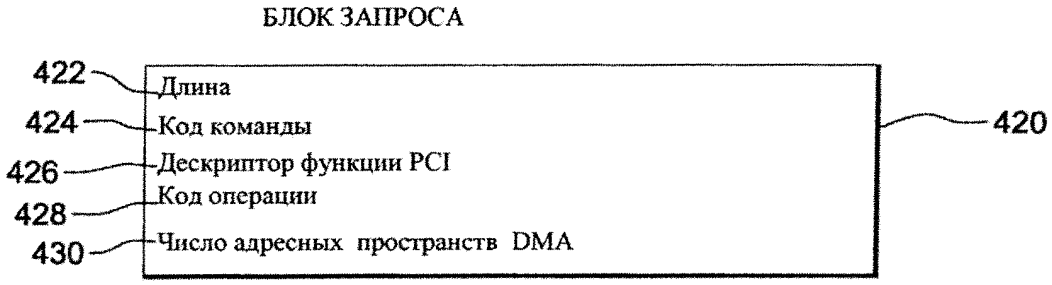
**ФИГ. 3А**



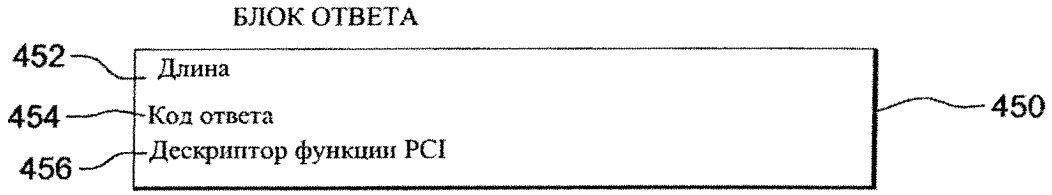

ФИГ. 3Б



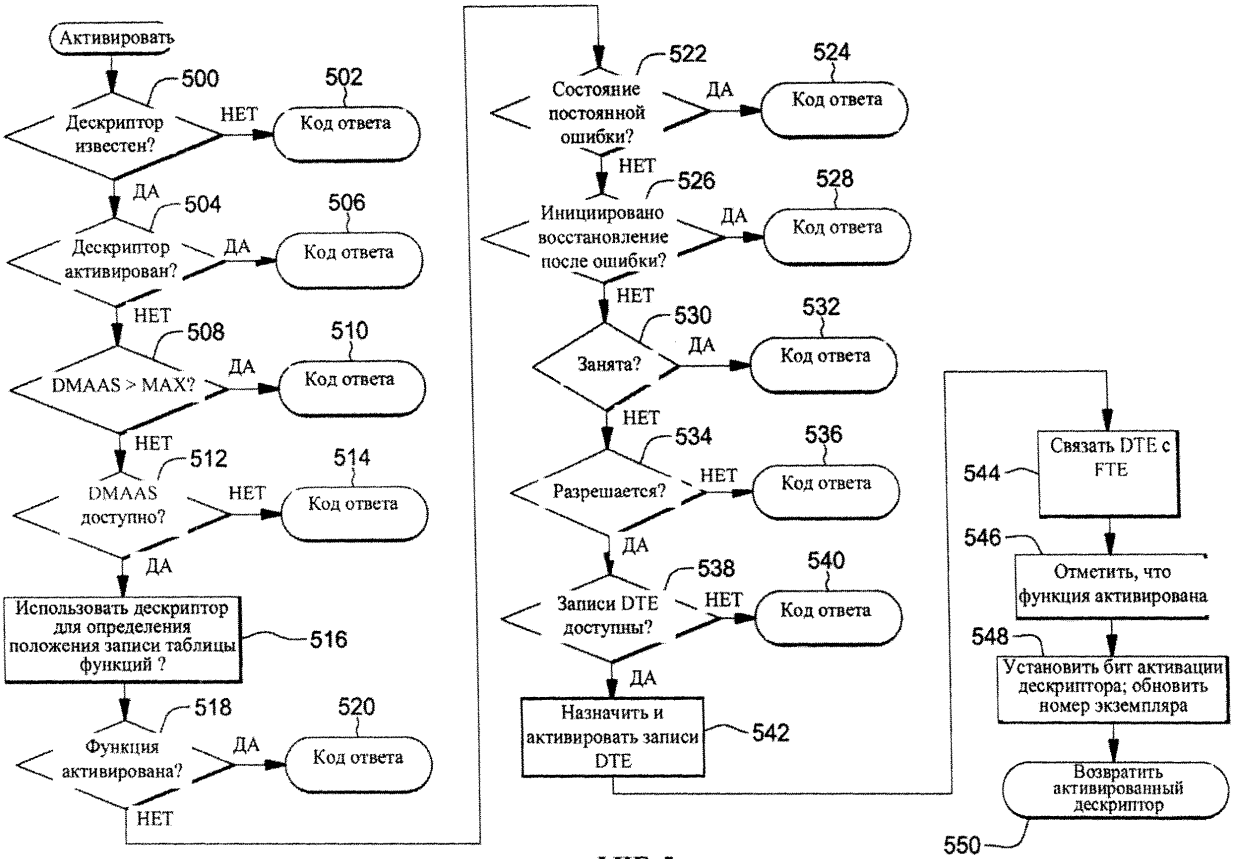

ФИГ. 4А



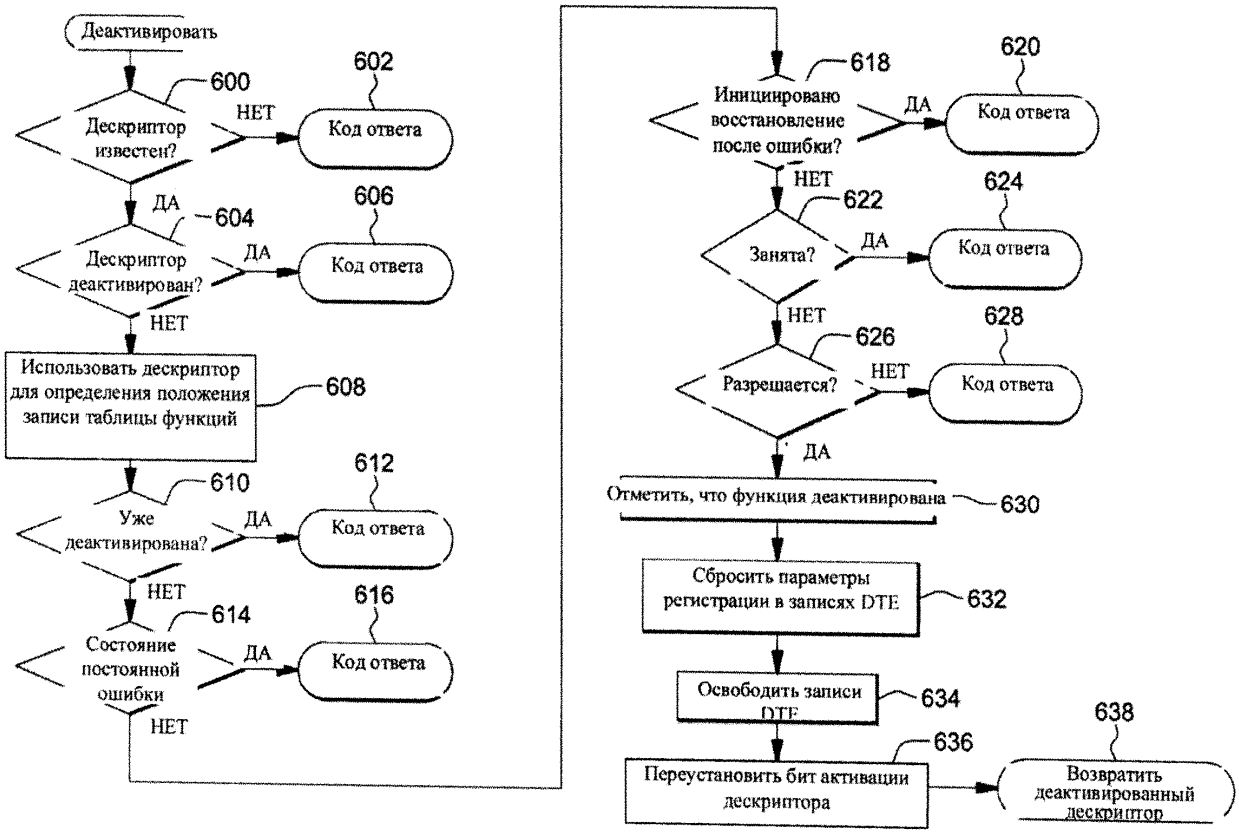

ФИГ. 4Б



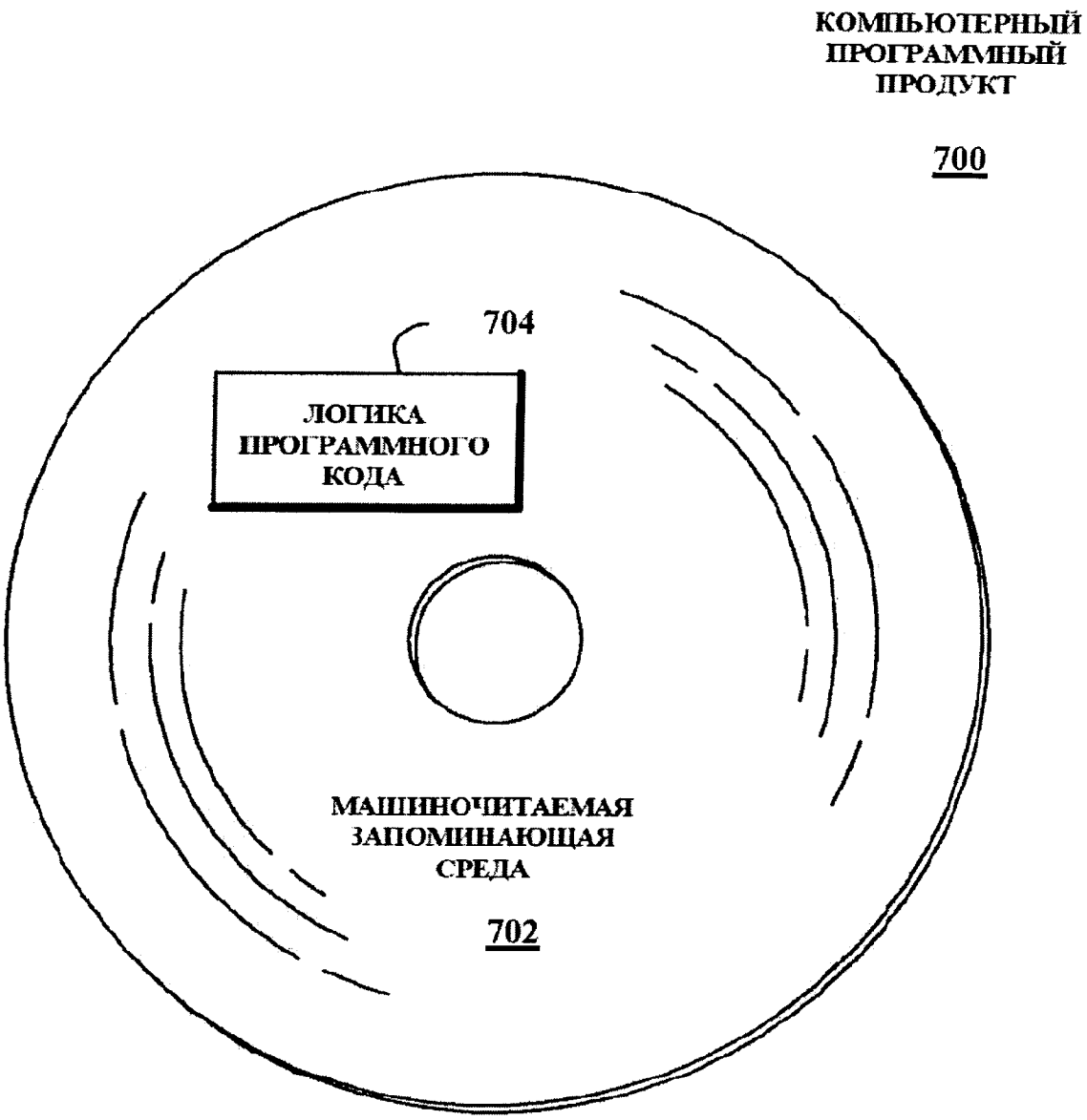

ФИГ. 4В


Активировать
500
Дескриптор известен?
НЕТ
502
Код ответа
ДА
504
Дескриптор активирован?
ДА
506
Код ответа
НЕТ
508
DMAAS > MAX?
ДА
510
Код ответа
НЕТ
512
DMAAS доступно?
НЕТ
514
Код ответа
ДА
Использовать дескриптор для определения положения записи таблицы функций ?
516
518
Функция активирована?
ДА
520
Код ответа
НЕТ
522
Состояние постоянной ошибки?
ДА
524
Код ответа
НЕТ
526
Инициировано восстановление после ошибки?
ДА
528
Код ответа
НЕТ
530
Занята?
ДА
532
Код ответа
НЕТ
534
Разрешается?
НЕТ
536
Код ответа
ДА
538
Записи DTE доступны?
НЕТ
540
Код ответа
ДА
Назначить и активировать записи DTE
542
544
Связать DTE с FTE
546
Отметить, что функция активирована
548
Установить бит активации дескриптора; обновить номер экземпляра
Возвратить активированный дескриптор
550

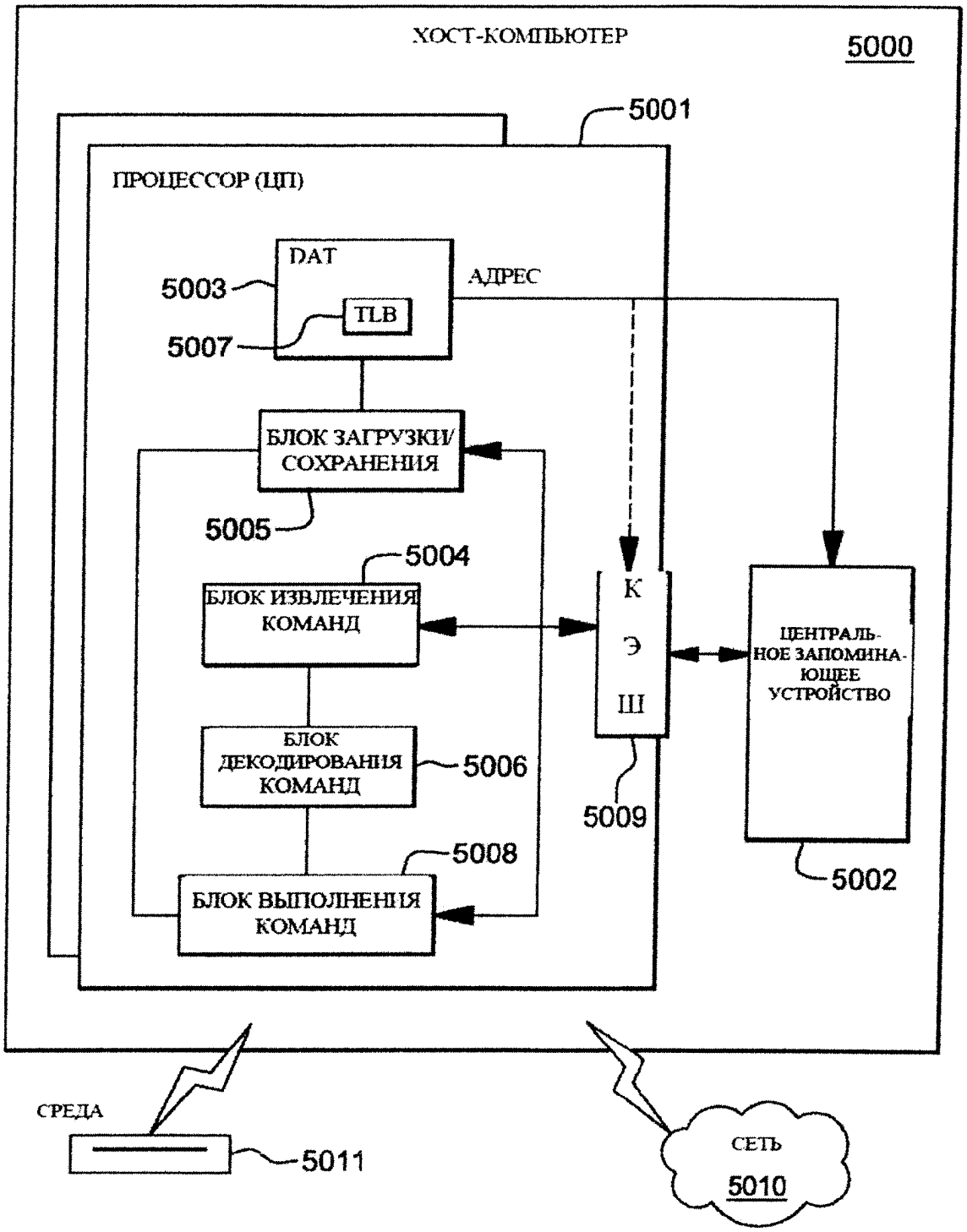
ФИГ. 5


Деактивировать
600
Дескриптор известен?
НЕТ
602
Код ответа
ДА
604
Дескриптор деактивирован?
ДА
606
Код ответа
НЕТ
Использовать дескриптор для определения положения записи таблицы функций
608
610
Уже деактивирована?
ДА
612
Код ответа
НЕТ
614
Состояние постоянной ошибки
ДА
616
Код ответа
НЕТ
618
Инициировано восстановление после ошибки?
ДА
620
Код ответа
НЕТ
622
Занята?
ДА
624
Код ответа
НЕТ
626
Разрешается?
НЕТ
628
Код ответа
ДА
Отметить, что функция деактивирована
630
Сбросить параметры регистрации в записях DTE
632
Освободить записи DTE
634
636
Переустановить бит активации дескриптора
638
Возвратить деактивированный дескриптор

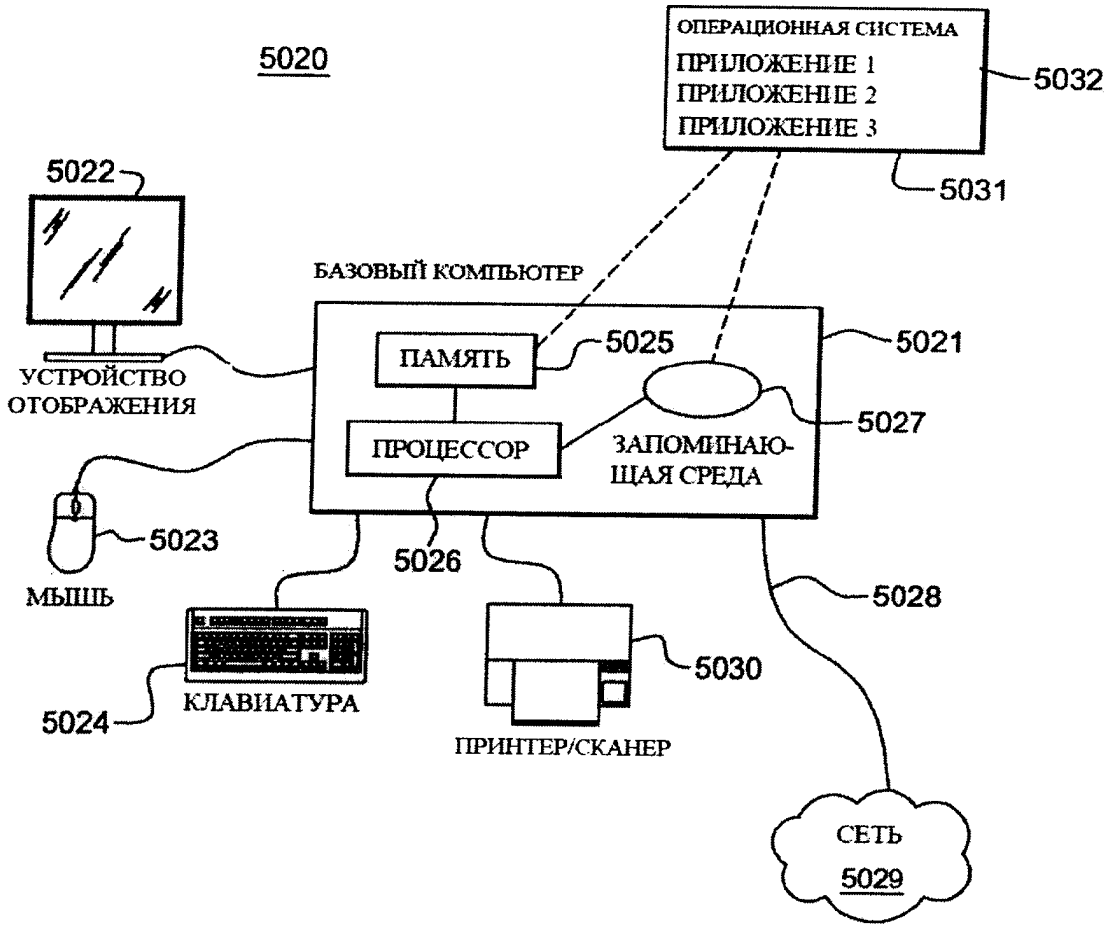
ФИГ. 6


КОМПЬЮТЕРНЫЙ ПРОГРАММНЫЙ ПРОДУКТ
700
704
ЛОГИКА ПРОГРАММНОГО КОДА
МАШИНОЧИТАЕМАЯ ЗАПОМИНАЮЩАЯ СРЕДА
702

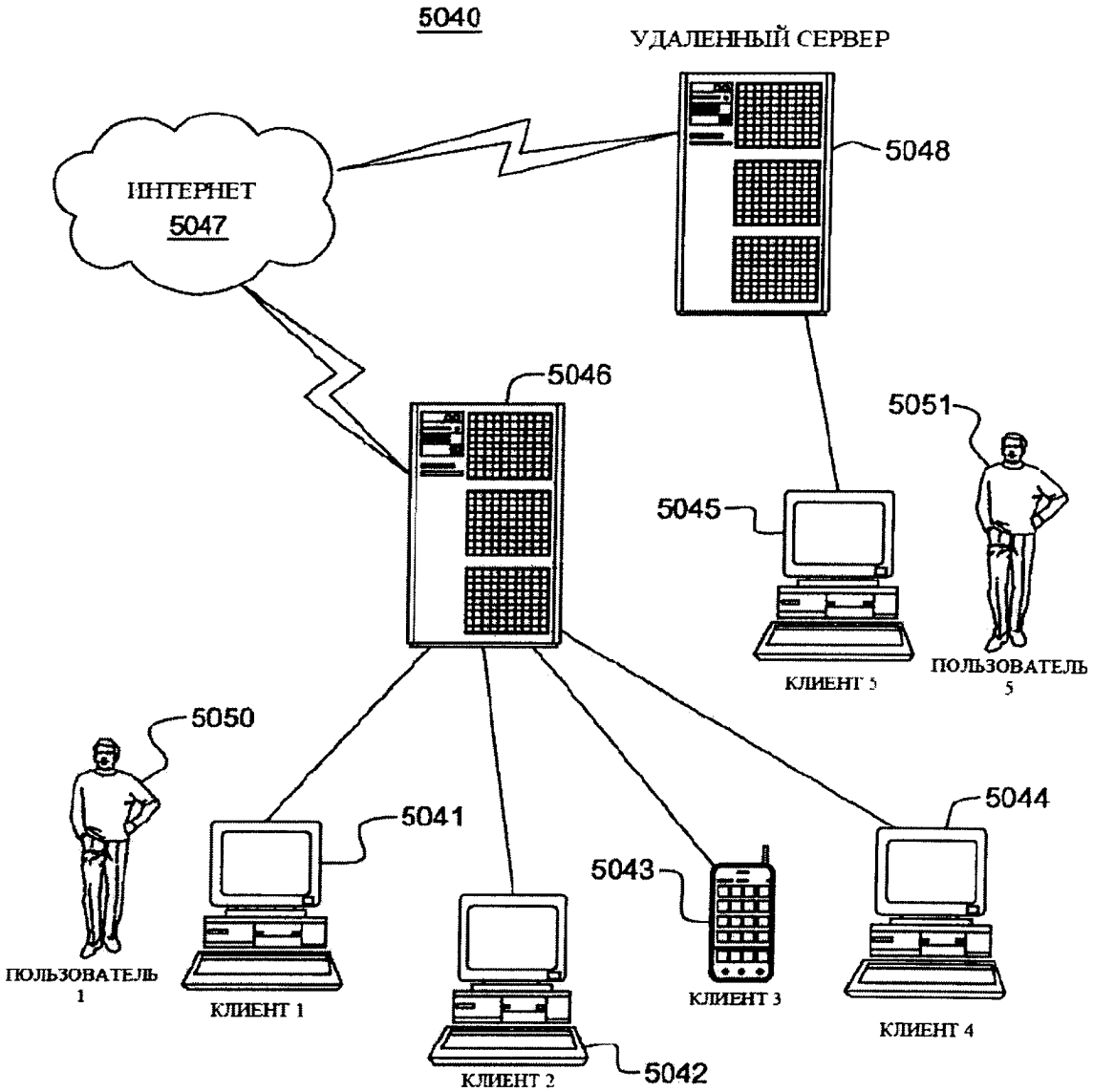
ФИГ. 7


ХОСТ-КОМПЬЮТЕР
5000
5001
ПРОЦЕССОР (ЦП)
DAT
TLB
5003
5007
АДРЕС
БЛОК ЗАГРУЗКИ/ СОХРАНЕНИЯ
5005
5004
БЛОК ИЗВЛЕЧЕНИЯ КОМАНД
БЛОК ДЕКОДИРОВАНИЯ КОМАНД
5006
5008
БЛОК ВЫПОЛНЕНИЯ КОМАНД
К
Э
Ш
5009
ЦЕНТРАЛЬНОЕ ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО
5002
СРЕДА
5011
СЕТЬ
5010

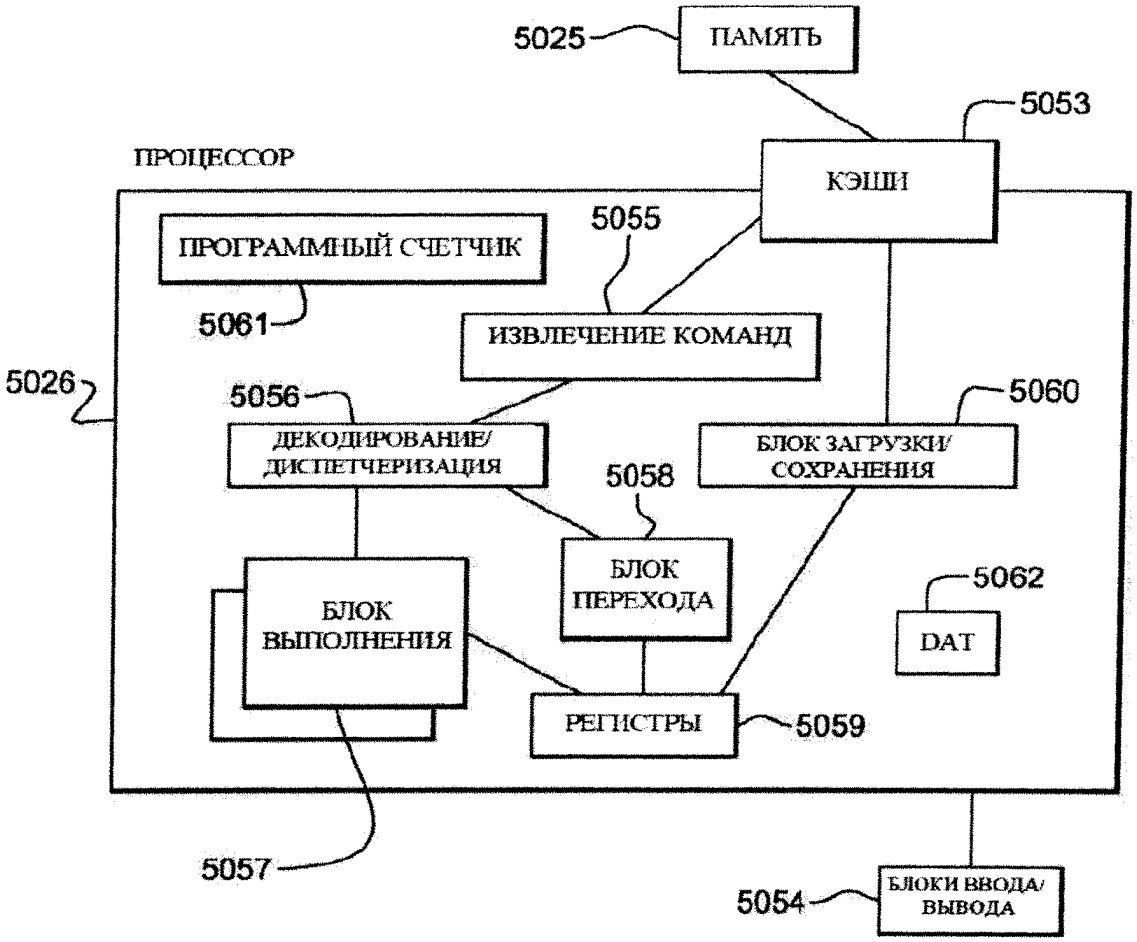
**ФИГ. 8**


5020
ОПЕРАЦИОННАЯ СИСТЕМА
ПРИЛОЖЕНИЕ 1
ПРИЛОЖЕНИЕ 2
ПРИЛОЖЕНИЕ 3
5032
5031
5022
БАЗОВЫЙ КОМПЬЮТЕР
ПАМЯТЬ
5025
5021
УСТРОЙСТВО ОТОБРАЖЕНИЯ
ПРОЦЕССОР
ЗАПОМИНАЮЩАЯ СРЕДА
5027
5023
МЫШЬ
5026
5028
КЛАВИАТУРА
5024
5030
ПРИНТЕР/СКАНЕР
СЕТЬ
5029

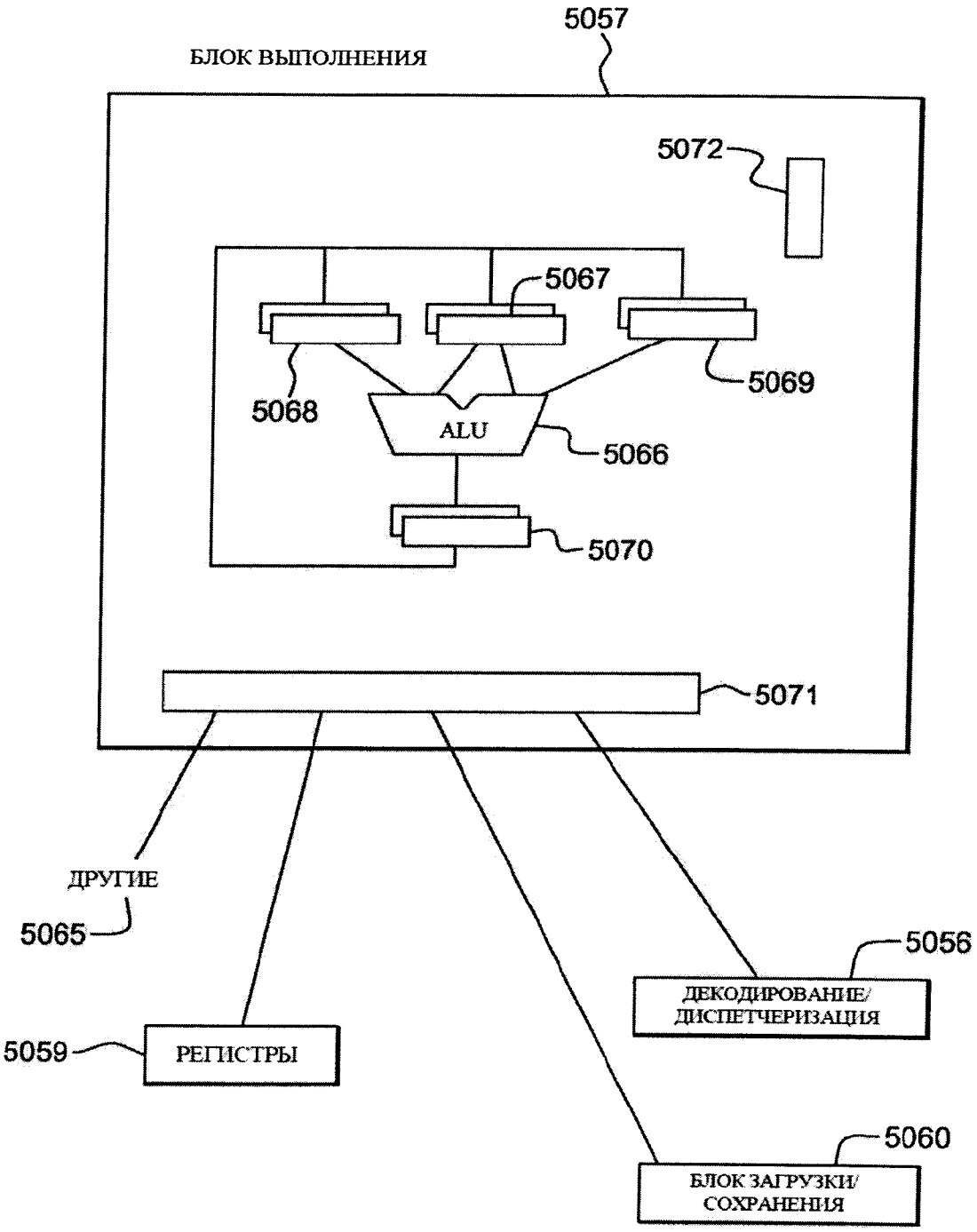
**ФИГ. 9**


5040
УДАЛЕННЫЙ СЕРВЕР
5048
ИНТЕРНЕТ
5047
5046
5051
5045
КЛИЕНТ 5
ПОЛЬЗОВАТЕЛЬ 5
5050
5041
5044
5043
ПОЛЬЗОВАТЕЛЬ 1
КЛИЕНТ 1
КЛИЕНТ 3
КЛИЕНТ 4
КЛИЕНТ 2
5042

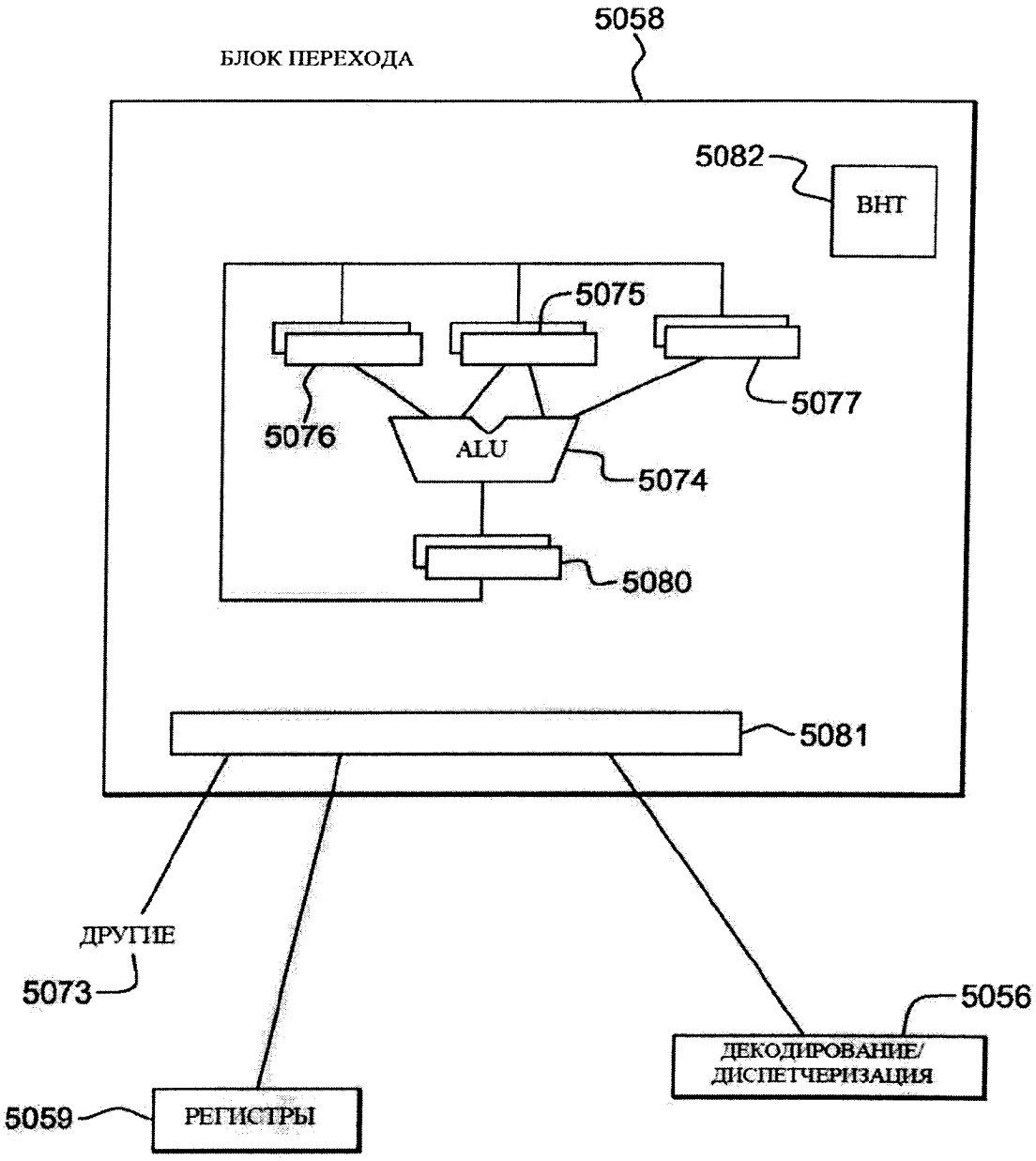
**ФИГ. 10**


5025
ПАМЯТЬ
5053
ПРОЦЕССОР
КЭШИ
ПРОГРАММНЫЙ СЧЕТЧИК
5055
5061
ИЗВЛЕЧЕНИЕ КОМАНД
5026
5056
5060
ДЕКОДИРОВАНИЕ/ ДИСПЕТЧЕРИЗАЦИЯ
БЛОК ЗАГРУЗКИ/ СОХРАНЕНИЯ
5058
БЛОК ПЕРЕХОДА
5062
БЛОК ВЫПОЛНЕНИЯ
DAT
РЕГИСТРЫ
5059
5057
5054
БЛОКИ ВВОДА/ ВЫВОДА

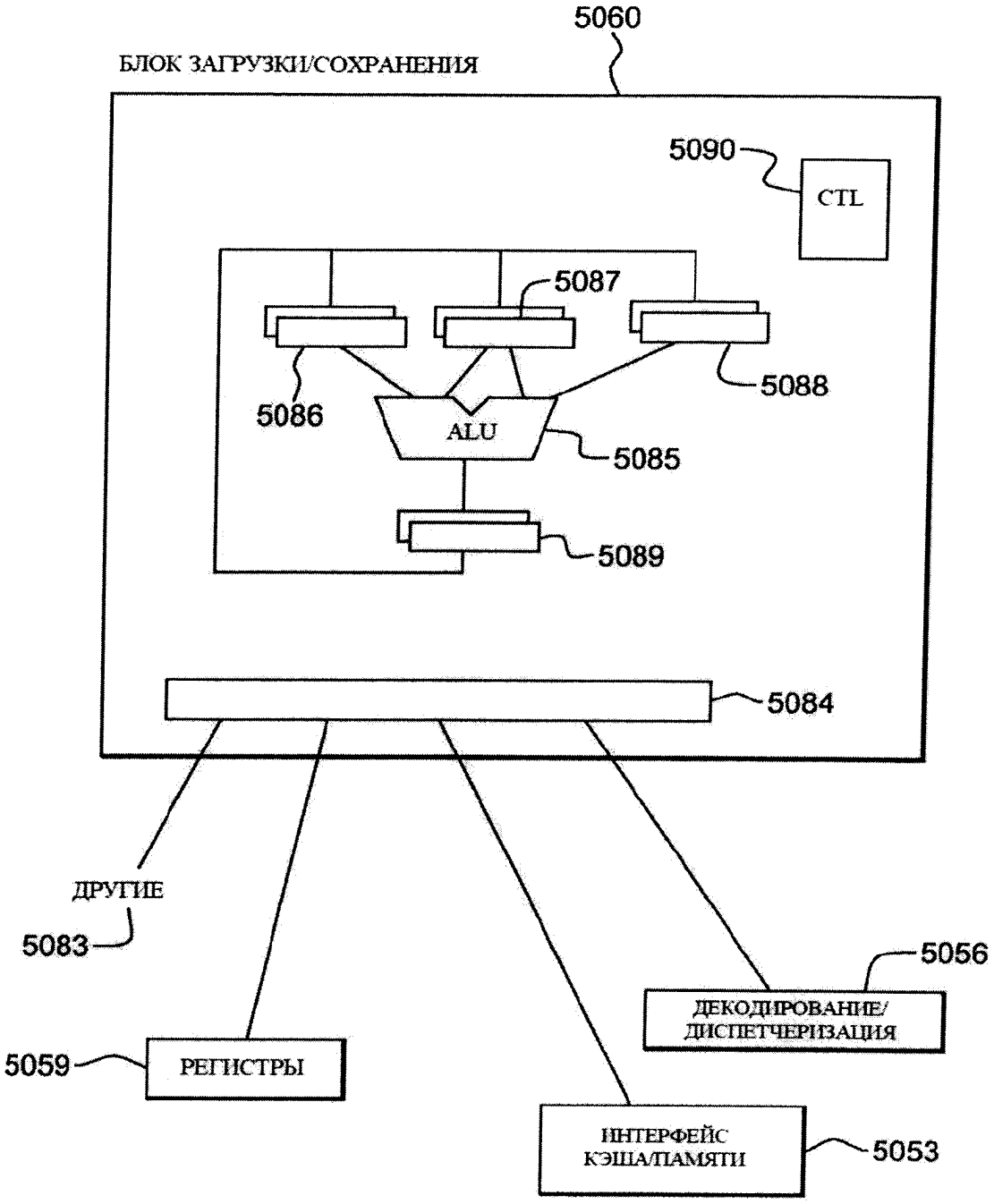
**ФИГ. 11**


5057
БЛОК ВЫПОЛНЕНИЯ
5072
5067
5068
5069
ALU
5066
5070
5071
ДРУГИЕ
5065
5056
ДЕКОДИРОВАНИЕ/
ДИСПЕТЧЕРИЗАЦИЯ
5059
РЕГИСТРЫ
5060
БЛОК ЗАГРУЗКИ/
СОХРАНЕНИЯ

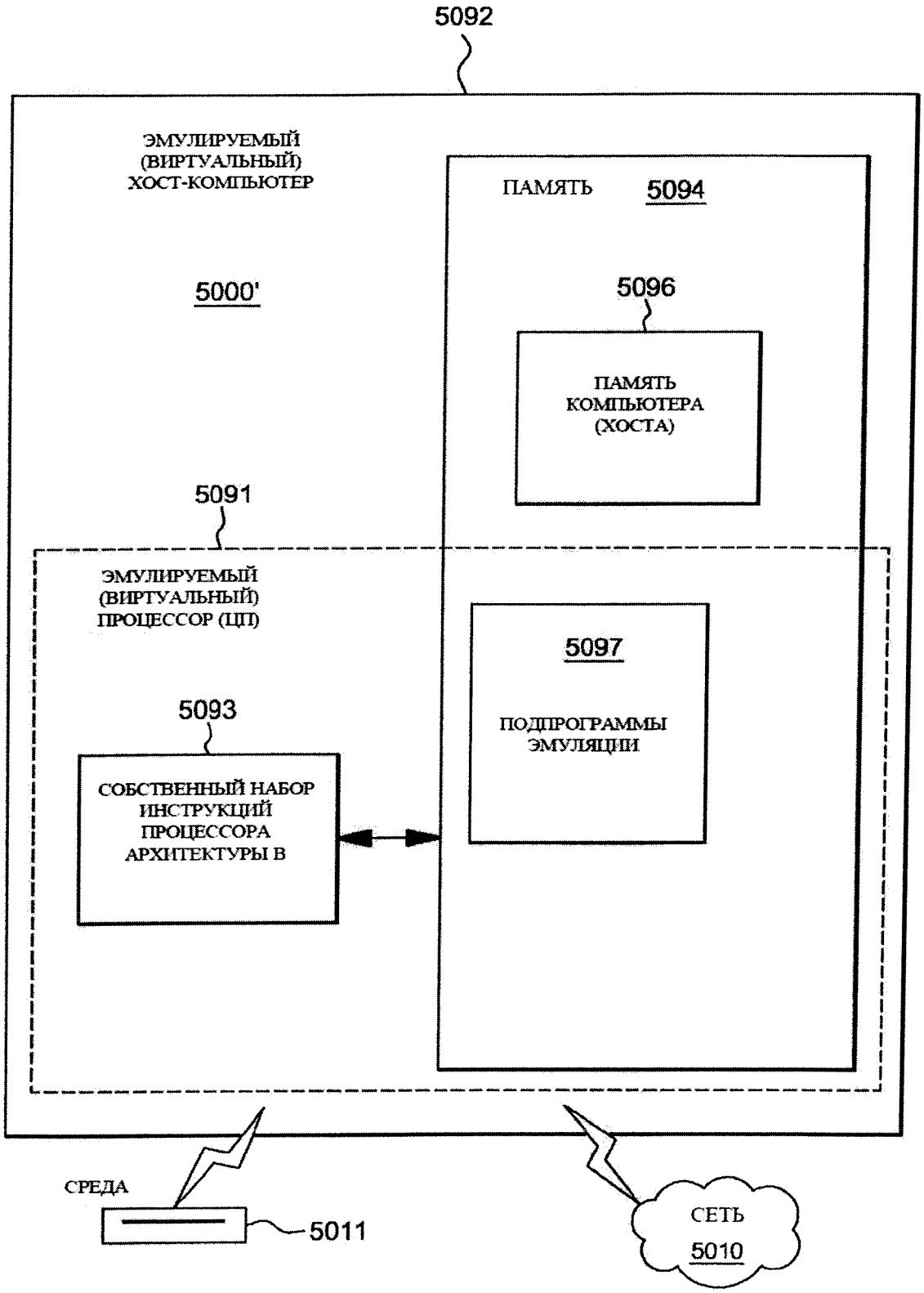
**ФИГ. 12А**


5058
БЛОК ПЕРЕХОДА
5082
ВНТ
5075
5077
5076
ALU
5074
5080
5081
ДРУГИЕ
5073
5056
ДЕКОДИРОВАНИЕ/
ДИСПЕТЧЕРИЗАЦИЯ
5059
РЕГИСТРЫ

**ФИГ. 12Б**


5060
БЛОК ЗАГРУЗКИ/СОХРАНЕНИЯ
5090
CTL
5087
5086
5088
ALU
5085
5089
5084
ДРУГИЕ
5083
5056
ДЕКОДИРОВАНИЕ/
ДИСПЕТЧЕРИЗАЦИЯ
5059
РЕГИСТРЫ
ИНТЕРФЕЙС
КЭША/ПАМЯТИ
5053

**ФИГ. 12B**


5092
ЭМУЛИРУЕМЫЙ (ВИРТУАЛЬНЫЙ) ХОСТ-КОМПЬЮТЕР
5000'
ПАМЯТЬ 5094
5096
ПАМЯТЬ КОМПЬЮТЕРА (ХОСТА)
5091
ЭМУЛИРУЕМЫЙ (ВИРТУАЛЬНЫЙ) ПРОЦЕССОР (ЦП)
5097
ПОДПРОГРАММЫ ЭМУЛЯЦИИ
5093
СОБСТВЕННЫЙ НАБОР ИНСТРУКЦИЙ ПРОЦЕССОРА АРХИТЕКТУРЫ В
СРЕДА
5011
СЕТЬ
5010

**ФИГ. 13**