РОССИЙСКАЯ ФЕДЕРАЦИЯ

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(19) **RU** (11) **2 664 413**(13) **C2**

(51) МПК
G06F 9/44 (2006.01)

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
G06F 9/44 (2006.01)

(21)(22) Заявка: **2016127224, 09.03.2015**

(24) Дата начала отсчета срока действия патента:
09.03.2015

Дата регистрации:
17.08.2018

Приоритет(ы):
(30) Конвенционный приоритет:
18.03.2014 US 14/217,840;
26.11.2014 US 14/554,806

(43) Дата публикации заявки: **18.04.2018** Бюл. № 11

(45) Опубликовано: **17.08.2018** Бюл. № **23**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **18.10.2016**

(86) Заявка PCT:
EP 2015/054850 (09.03.2015)

(87) Публикация заявки PCT:
WO 2015/139992 (24.09.2015)

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

(72) Автор(ы):
ГШВИНД Михаэль Карл (US)

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

(56) Список документов, цитированных в отчете о поиске: **US 2012/260064 A1, 11.10.2012. US 2012/0260071 A1, 11.10.2012. US 2014/0122843 A1, 01.05.2014.**

(54) КОНФИГУРАЦИЯ АРХИТЕКТУРНОГО РЕЖИМА В ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ

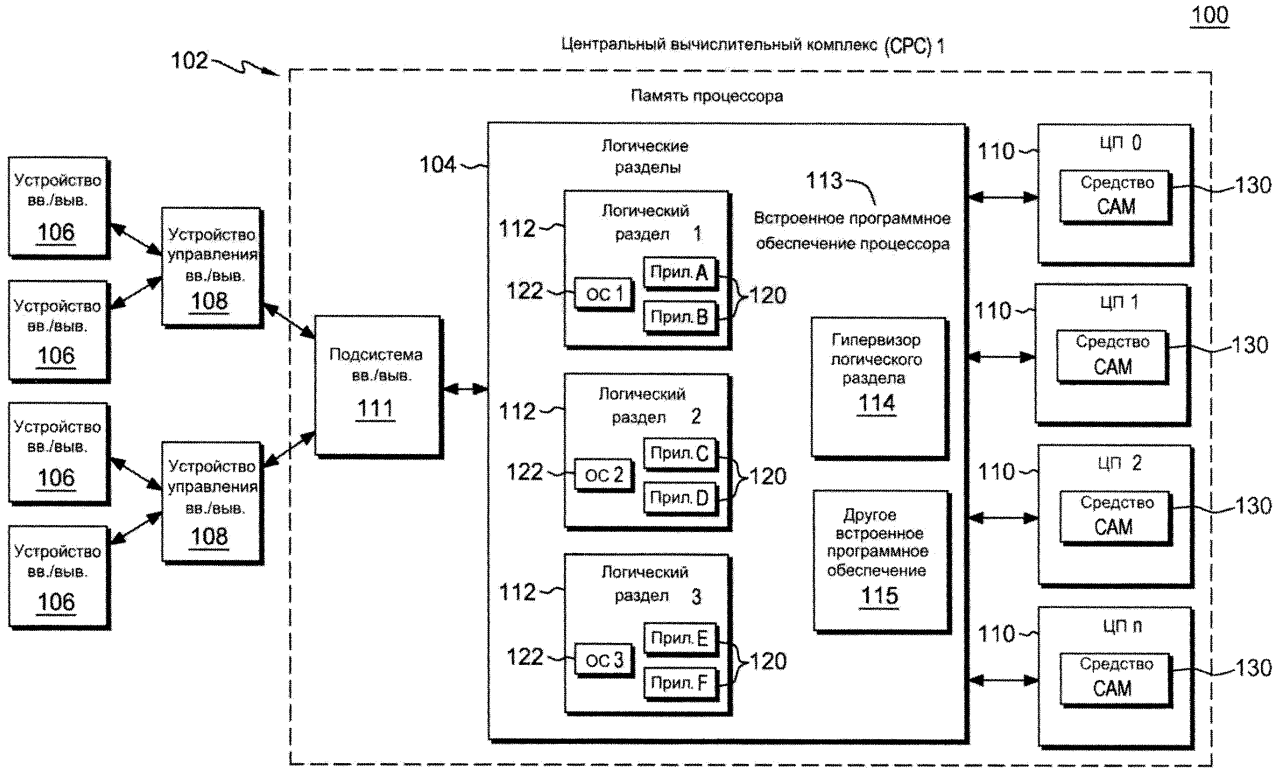
(57) Реферат:

Изобретение относится к технологиям сетевой связи. Технический результат заключается в повышении скорости обработки данных. Содержит: выявление посредством процессора того, что средство конфигурации архитектурного режима инсталлировано в вычислительном окружении, сконфигурированном для нескольких архитектурных режимов и имеющем заданную последовательность включения, которая предназначена для включения вычислительного окружения в одном архитектурном режиме из нескольких архитектурных режимов, причем один архитектурный режим содержит первую архитектуру системы команд и имеет первый набор поддерживаемых сервисов, проведение посредством процессора реконфигурирования вычислительного окружения для ограничения использования одного архитектурного режима, причем реконфигурирование включает в себя: выборку отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме из нескольких архитектурных режимов и - выполнение отличной последовательности

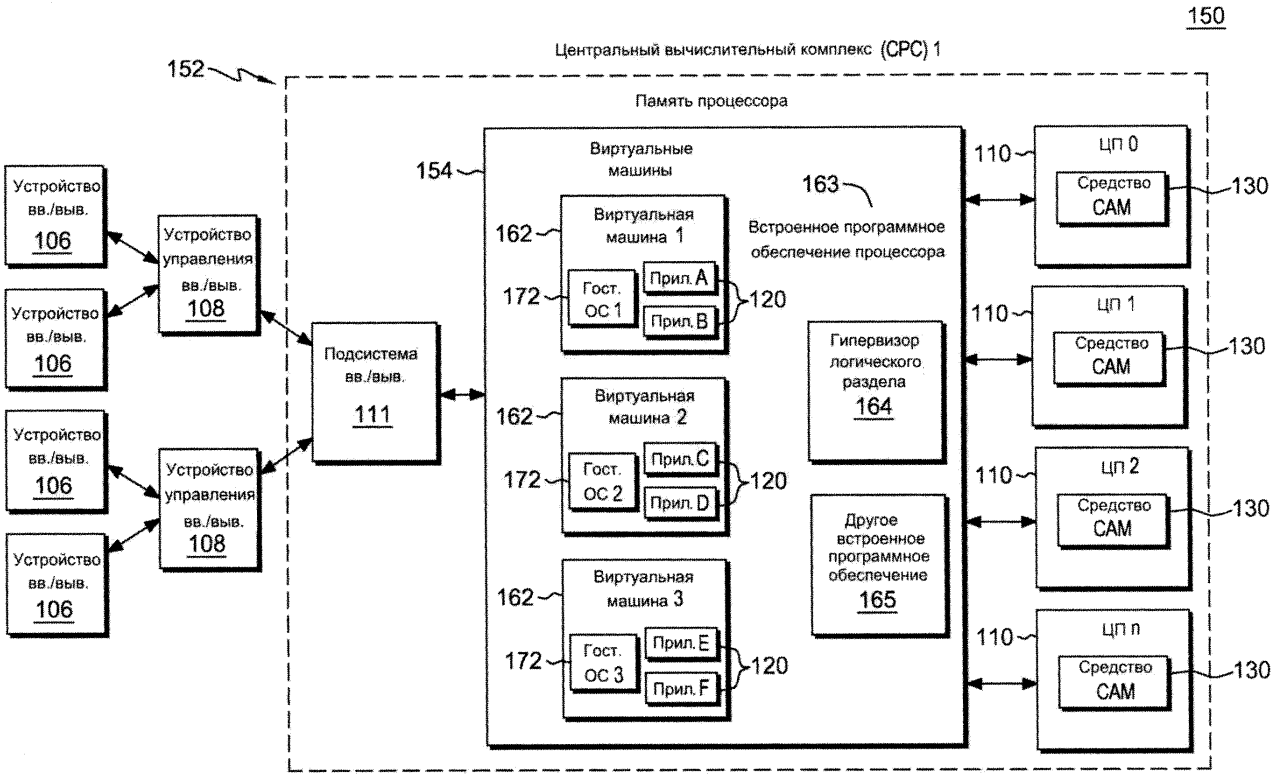
RU 2664413 C2

RU 2664413 C2

включения для включения вычислительного окружения в другом архитектурном режиме вместо одного архитектурного режима, ограничивая использование одного архитектурного режима. 6 н. и 14 з.п. ф-лы, 22 ил.



Фиг. 1А



Фиг. 1Б

RU 2664413 C2

RU 2664413 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 664 413**(13) **C2**

(51) Int. Cl.
G06F 9/44 (2006.01)

(12) ABSTRACT OF INVENTION

(52) CPC
G06F 9/44 (2006.01)

(21)(22) Application: **2016127224, 09.03.2015**

(24) Effective date for property rights:
09.03.2015

Registration date:
17.08.2018

Priority:
(30) Convention priority:
18.03.2014 US 14/217,840;
26.11.2014 US 14/554,806

(43) Application published: **18.04.2018** Bull. № 11

(45) Date of publication: **17.08.2018** Bull. № **23**

(85) Commencement of national phase: **18.10.2016**

(86) PCT application:
EP 2015/054850 (09.03.2015)

(87) PCT publication:
WO 2015/139992 (24.09.2015)

Mail address:
105082, Moskva, Spartakovskij per., 2, str. 1, sektsiya 1, etazh 3, EVROMARKPAT

(72) Inventor(s):
GSHVIND Mikhael Karl (US)

(73) Proprietor(s):
INTERNESHNL BIZNES MASHINZ KORPOREJSHN (US)

RU 2664413 C2

(54) ARCHITECTURAL MODE CONFIGURATION IN COMPUTING SYSTEM

(57) Abstract:

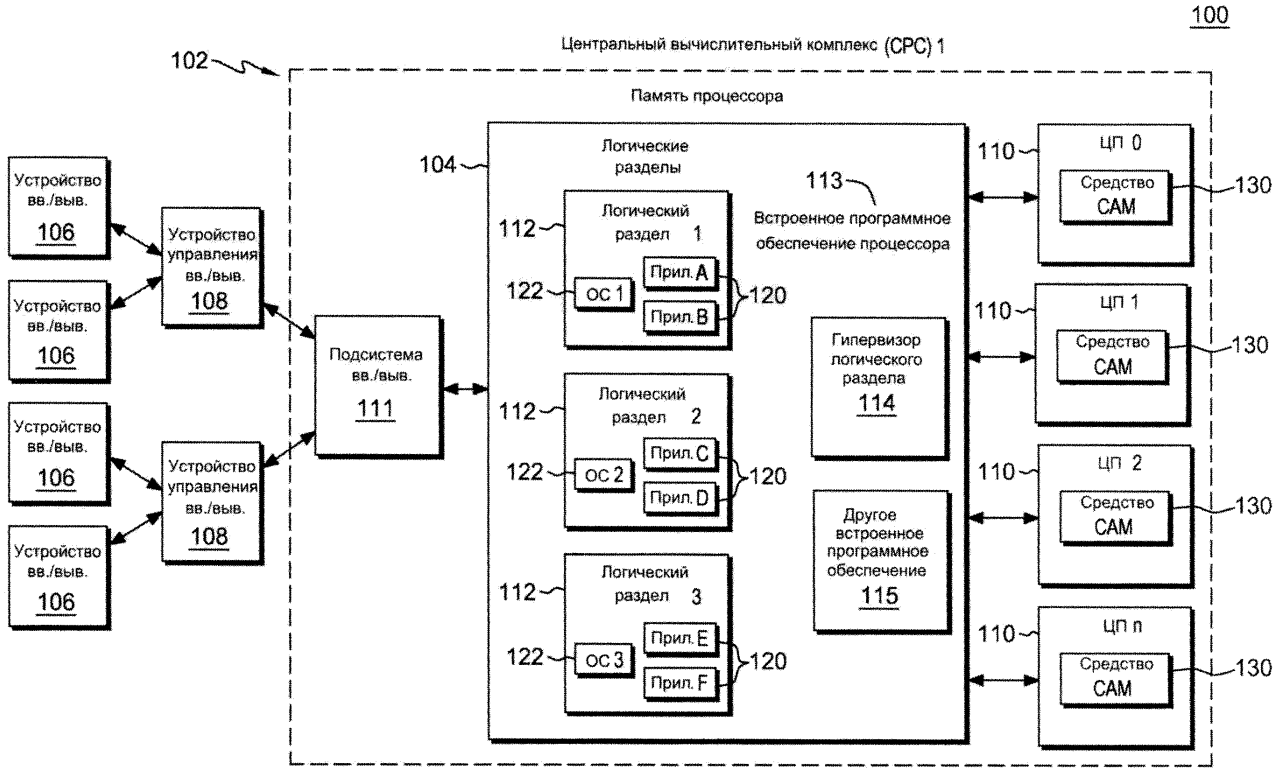
FIELD: computing; counting.

SUBSTANCE: invention relates to network communication technologies. Method comprises: determining, by a processor, that a configuration architectural mode facility is installed in a computing environment that is configured for a plurality of architectural modes and has a defined power-on sequence that is to power-on the computing environment in one architectural mode of the plurality of architectural modes, the one architectural mode comprising a first instruction set architecture and having a first set of supported features, reconfiguring, by the processor, the computing environment to restrict use of the one architectural mode, wherein the reconfiguring comprises: selecting a different power-on sequence to power-on the computing environment in another architectural mode of the plurality of architectural modes and executing the different power-on sequence to power-on the computing environment in the another architectural mode in place of the one architectural mode restricting use of the one architectural mode.

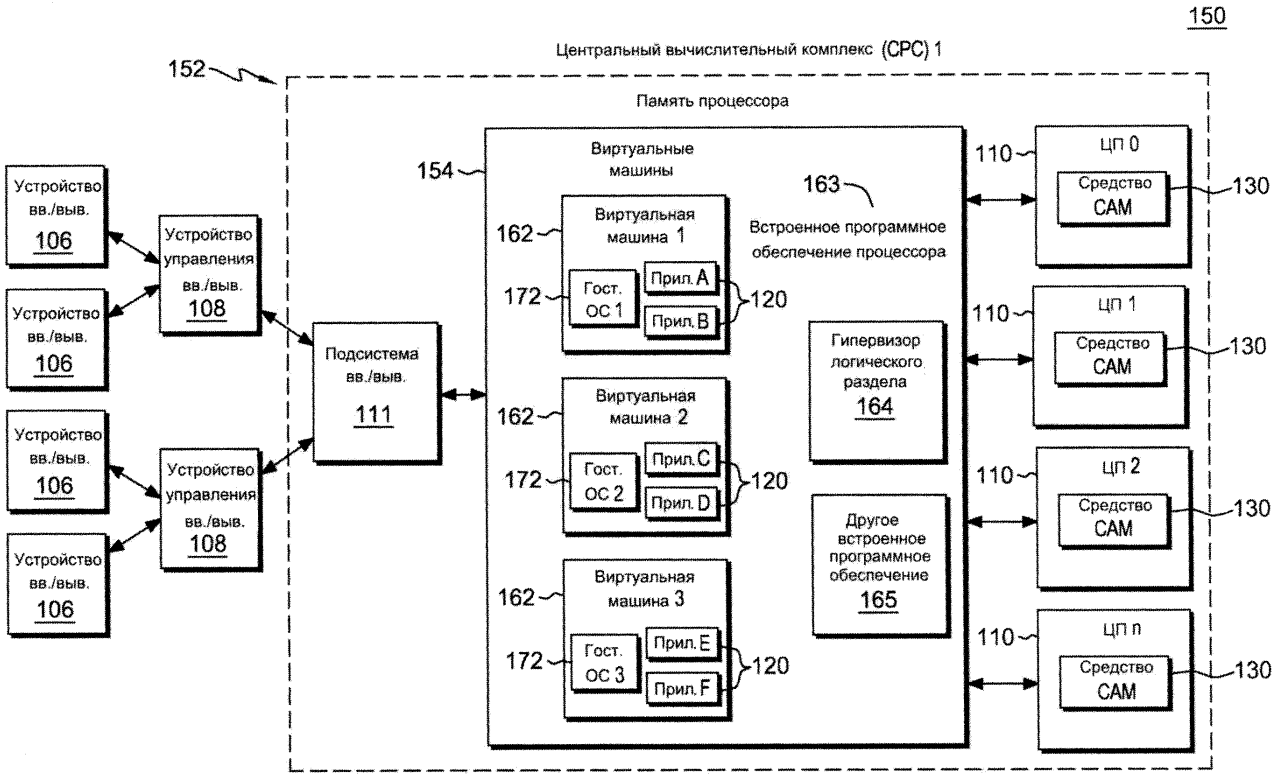
EFFECT: technical result consists in improvement of data processing speed.

20 cl, 22 dwg

RU 2664413 C2



Фиг. 1А



Фиг. 1Б

RU 2664413 C2

Область техники

Один или несколько аспектов относятся, в общем, к конфигурациям вычислительных окружений и, прежде всего, к изменению конфигураций таких окружений.

Уровень техники

Вычислительные окружения предлагают диапазон инструментов и функций в зависимости от архитектурных конфигураций окружений. Две архитектуры, предложенные International Business Machines Corporation, Армонк, Нью-Йорк, включают в себя ESA/390 и z/Архитектуру.

ESA/390 является предшествующей архитектурой по отношению к z/Архитектуре. Однако, когда z/Архитектура была введена, поддержка ESA/390 была продолжена. Для поддержки обеих архитектур в одном окружении требуется следовать некоторым процедурам. Например, при включении загружается ESA/390, а затем, при желании, может быть сделано переключение на z/Архитектуру.

Это позволяет унаследованному программному обеспечению продолжать выполнение без изменений. Другие такие процедуры предоставляются с целью поддержки обеих архитектурных конфигурации в одном окружении.

Однако тестирование виртуальной памяти является затратным. По мере того, как архитектура приближается к концу своего жизненного цикла, может оказаться желательным предоставление традиционных окружений, например для систем с использованием минимальной поддержки архитектуры, таких как операционные системы DOS (например, такие как MS DOS или CMS), которые функционируют, прежде всего, в качестве окружений интерпретатора командной строки, или для окружений, использующихся для выполнения части BIOS (и которые могут производить выполнение без связанных с виртуальной памятью сложностей).

Поэтому на уровне техники существует потребность в рассмотрении вышеупомянутой проблемы.

Сущность изобретения

Недостатки известного уровня техники преодолеваются и преимущества обеспечиваются посредством предоставления компьютерного программного продукта для реконфигурирования вычислительного окружения. Компьютерный программный продукт включает в себя, например, машиночитаемый информационный носитель, считываемый посредством устройства обработки данных, и сохраняющий подлежащие выполнению посредством устройства обработки данных команды для осуществления способа. Способ включает в себя, например, выявление посредством процессора того, что средство конфигурации архитектурного режима инсталлировано в вычислительном окружении, сконфигурированном для нескольких архитектурных режимов и имеющем заданную последовательность включения, которая предназначена для включения вычислительного окружения в одном архитектурном режиме из нескольких архитектурных режимов, причем один архитектурный режим содержит первую архитектуру системы команд и имеет первый набор поддерживаемых сервисов, на основании выявления того, что средство конфигурации архитектурного режима инсталлировано, проведение посредством процессора реконфигурирования вычислительного окружения для ограничения использования одного архитектурного режима, причем реконфигурирование включает в себя: выборку отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме из нескольких архитектурных режимов, причем другой архитектурный режим отличен от одного архитектурного режима, и другой архитектурный режим содержит вторую архитектуру системы команд и имеет второй

набор поддерживаемых сервисов, и выполнение отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме вместо одного архитектурного режима, ограничивая использование одного архитектурного режима.

При рассмотрении с позиции первого аспекта, настоящее изобретение предоставляет способ реконфигурирования вычислительного окружения, причем способ содержит: выявление посредством процессора того, что средство конфигурации архитектурного режима инсталлировано в вычислительном окружении, сконфигурированном для нескольких архитектурных режимов и имеющем заданную последовательность включения, которая предназначена для включения вычислительного окружения в одном архитектурном режиме из нескольких архитектурных режимов, причем один архитектурный режим содержит первую архитектуру системы команд и имеет первый набор поддерживаемых сервисов, на основании выявления того, что средство конфигурации архитектурного режима инсталлировано, проведение посредством процессора реконфигурирования вычислительного окружения для ограничения использования одного архитектурного режима, причем реконфигурирование включает в себя: выборку отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме из нескольких архитектурных режимов, причем другой архитектурный режим отличен от одного архитектурного режима, и другой архитектурный режим содержит вторую архитектуру системы команд и имеет второй набор поддерживаемых сервисов, и выполнение отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме вместо одного архитектурного режима, ограничивая использование одного архитектурного режима.

При рассмотрении с позиции другого аспекта, настоящее изобретение предоставляет способ конфигурирования вычислительного окружения, причем способ содержит: конфигурирование посредством процессора вычислительного окружения для выполнения операций в выбранном архитектурном режиме, конфигурирование содержащее: начало инициализации вычислительного окружения с использованием сохраненного слова состояния программы, причем сохраненное слово состояния программы имеет формат архитектурного режима, отличающегося от выбранного архитектурного режима, выявление того, что сохраненное слово состояния программы имеет формат архитектурного режима, отличающегося от выбранного архитектурного режима, на основании выявления того, что сохраненное слово состояния программы имеет формат архитектурного режима, отличающегося от выбранного архитектурного режима, проведение автоматического изменения сохраненного слова состояния программы для приобретения им формата выбранного архитектурного режима, причем автоматическое изменение выполняется в отсутствии явного запроса на переключение на выбранный архитектурный режим, и завершение инициализации вычислительного окружения с помощью измененного слова состояния программы для конфигурирования вычислительного окружения в выбранном архитектурном режиме.

При рассмотрении с позиции другого аспекта, настоящее изобретение предоставляет компьютерную систему для реконфигурирования вычислительного окружения, компьютерную систему содержащую: память, а также сообщенный с памятью процессор, причем компьютерная система сконфигурирована для осуществления способа, способа содержащего: выявление посредством процессора того, что средство конфигурации архитектурного режима инсталлировано в вычислительном окружении, сконфигурированном для нескольких архитектурных режимов и имеющем заданную

последовательность включения, которая предназначена для включения вычислительного окружения в одном архитектурном режиме из нескольких архитектурных режимов, причем один архитектурный режим содержит первую архитектуру системы команд и имеет первый набор поддерживаемых сервисов, на основании выявления того, что средство конфигурации архитектурного режима инсталлировано, проведение посредством процессора реконфигурирования вычислительного окружения для ограничения использования одного архитектурного режима, причем реконфигурирование включает в себя: выборку отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме из нескольких архитектурных режимов, причем другой архитектурный режим отличен от одного архитектурного режима, и другой архитектурный режим содержит вторую архитектуру системы команд и имеет второй набор поддерживаемых сервисов, и выполнение отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме вместо одного архитектурного режима, ограничивая использование одного архитектурного режима.

При рассмотрении с позиции другого аспекта, настоящее изобретение предоставляет машиночитаемый информационный носитель, считываемый посредством устройства обработки данных и содержащий программу для реконфигурирования вычислительного окружения, включающую в себя команды, выполняемые посредством устройства обработки данных для осуществления соответствующего предлагаемого в изобретении способа. При рассмотрении с позиции другого аспекта, настоящее изобретение предоставляет машиночитаемый информационный носитель, считываемый посредством устройства обработки данных и содержащий программу для конфигурирования вычислительного окружения, включающую в себя команды, выполняемые посредством устройства обработки данных для осуществления соответствующего предлагаемого в изобретении способа.

При рассмотрении с позиции другого аспекта, настоящее изобретение предоставляет машиночитаемый информационный носитель, в котором сохранена компьютерная программа, содержащая участки программного кода и загружаемая во внутреннюю память цифровой вычислительной машины, когда данная программа выполняется на компьютере для осуществления предлагаемого в изобретении способа.

Относящиеся к одному или нескольким вариантам осуществления способы и системы также описаны и заявлены в настоящем документе. Кроме того, относящиеся к одному или нескольким вариантам осуществления услуги также описаны и могут быть заявлены в настоящем документе.

Осуществлены дополнительные функции и преимущества. Другие варианты осуществления и аспекты подробно описываются в настоящем документе и считаются частью заявленного изобретения.

Краткое описание чертежей

Настоящее изобретение в дальнейшем описывается исключительно в качестве примера со ссылками на предпочтительные варианты осуществления, как показано на последующих чертежах:

Фиг. 1А изображает один пример вычислительного окружения для охвата и использования одного или нескольких аспектов средства конфигурации архитектурного режима согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 1Б изображает один пример виртуального вычислительного окружения для охвата и использования одного или нескольких аспектов средства конфигурации архитектурного режима согласно предпочтительному варианту осуществления

настоящего изобретения,

Фиг. 2 изображает другой пример вычислительного окружения для охвата и использования одного или нескольких аспектов средства конфигурации архитектурного режима согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 3А изображает еще один пример вычислительного окружения для охвата и использования одного или нескольких аспектов средства конфигурации архитектурного режима согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 3Б изображает более подробную информацию по памяти на фиг. 3А согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 4А изображает один вариант осуществления логики для включения вычислительного окружения в одном архитектурном режиме согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 4В изображает один вариант осуществления последующей обработки, связанной с процессом включения на фиг. 4А согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 5 изображает один вариант осуществления формата слова состояния программы согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 6А изображает один вариант осуществления логики для включения вычислительного окружения в архитектурном режиме, отличающемся от одного архитектурного режима, включение для которого показано на фиг. 4А, согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 6Б изображает один вариант осуществления последующей обработки, связанной с процессом включения на фиг. 6А согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 7 изображает один пример формата команды загрузки слова состояния программы согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 8А изображает один пример формата команды процессора обработки сигналов согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 8Б изображает один вариант осуществления обработки, связанный с командой процессора обработки сигналов на фиг. 8А, согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 9 изображает один вариант осуществления логики для включения вычислительного окружения в реконфигурированной конфигурации согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 10 изображает другие необходимые изменения при реконфигурировании вычислительного окружения согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 11 изображает один вариант осуществления логики для сброса вычислительного окружения,

Фиг. 12 изображает один вариант осуществления логики для конфигурирования вычислительного окружения согласно предпочтительному варианту осуществления настоящего изобретения,

Фиг. 13 изображает один вариант осуществления компьютерного программного продукта согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения,

Фиг. 14 изображает один вариант осуществления системы хост-компьютера согласно известному уровню техники, в котором может быть реализован предпочтительный

вариант осуществления настоящего изобретения,

Фиг. 15 изображает другой пример компьютерной системы согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения,

Фиг. 16 изображает другой пример содержащей компьютерную сеть компьютерной системы согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения,

Фиг. 17 изображает один вариант осуществления различных элементов компьютерной системы согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения,

Фиг. 18А изображают один вариант осуществления устройства выполнения компьютерной системы на фиг. 17 согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения,

Фиг. 18Б изображает один вариант осуществления устройства обработки ветвлений компьютерной системы на фиг. 17 согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения,

Фиг. 18В изображает один вариант осуществления устройства загрузки и хранения компьютерной системы на фиг. 17 согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения,

Фиг. 19 изображает один вариант осуществления эмулированной системы хост-компьютера согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения,

Фиг.20 изображает один вариант осуществления узла облачных вычислений согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения,

Фиг. 21 изображает один вариант осуществления окружения облачных вычислений согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения, и

Фиг. 22 изображает один пример уровней модельной абстракции согласно известному уровню техники, в котором может быть реализован предпочтительный вариант осуществления настоящего изобретения.

Подробное описание

Согласно одному аспекту предоставляется инструмент, который ограничивает использование конфигурации посредством сконфигурированного для поддержки множественных конфигураций вычислительного окружения таким образом, что один или несколько аспектов ограниченной конфигурации являются недоступными для использования. В качестве примера, процессор конфигурируется в конфигурации архитектурного режима (CAM). В CAM, вычислительное окружение (например, процессор, логический раздел, гость), которое первоначально является сконфигурированным для нескольких архитектур, например устаревшей архитектуры и расширенной архитектуры, реконфигурируется таким образом, что оно более не поддерживает один или несколько аспектов по меньшей мере одной архитектуры, такой как устаревшая архитектура. В такой конфигурации, неподдерживаемые аспекты архитектуры не являются доступными.

В качестве одного конкретного примера, в вычислительных окружениях,

поддерживающих множественные архитектуры, такие как ESA/390 и z/Архитектура, предоставляется средство конфигурации архитектурного режима (CZAM) для z/Архитектуры, которое удаляет способность использования аспектов ESA/390. Вместо этого используется z/Архитектура (и/или, в других вариантах осуществления, другая архитектура, отличная от ESA/390). CZAM может быть применено к предназначенной для исходной среды машине, логическому разделу и/или виртуальному гостю, в качестве примеров.

Один пример вычислительного окружения для охвата и использования одного или нескольких аспектов средства конфигурации архитектурного режима описан со ссылками на фиг. 1A. Как показано на фиг. 1A, в одном примере вычислительное окружение 100 основано на z/Архитектуре, предлагаемой International Business Machines (IBM®) Corporation, Армонк, Нью-Йорк), z/Архитектура описана в публикации патента IBM под названием «z/Архитектура, принципы работы» (z/Architecture, Principles of Operation), публикация №SA22-7932-09, 10-й выпуск, сентябрь 2012. Хотя вычислительное окружение основано на z/Архитектуре, в одном предпочтительном варианте осуществления оно также поддерживает одну или несколько других архитектурных конфигураций, таких как ESA/390.

В качестве примера, вычислительное окружение 100 включает в себя центральный вычислительный комплекс (CPC) 102, присоединенный к одному или нескольким устройствам 106 ввода/вывода (I/O) через одно или несколько устройств 108 управления. Центральный вычислительный комплекс 102 включает в себя, например, память 104 процессора (известную также под названием оперативная память, основная память, центральная память), соединенную с одним или несколькими центральными процессорами (известными также под названием центральных вычислительных устройств (ЦП)) 110, и с подсистемой 111 ввода/вывода, каждый из указанных элементов описан ниже.

Память 104 процессора включает в себя, например, один или несколько разделов 112 (например, логических разделов), а также встроенное программное обеспечение 113 процессора, которое включает в себя гипервизор 114 логического раздела и другое встроенное программное обеспечение 115 процессора. Один пример гипервизора 114 логического раздела представлен администратором ресурсов процессора/системы Processor Resource/Systems Manager™ (PR/SM), предлагаемым International Business Machines Corporation, Армонк, Нью-Йорк.

Логический раздел функционирует как отдельная система и имеет в себе одно или несколько приложений 120 и, опционально, резидентную операционную систему 122, которая может отличаться для каждого логического раздела. В одном предпочтительном варианте осуществления операционная система является z/OS операционной системой, z/VM операционной системой, z/Linux операционной системой или операционной системой TPF, предлагаемой International Business Machines Corporation, Армонк, Нью-Йорк. Логическими разделами 112 управляет гипервизор 114 логических разделов, который реализован посредством встроенного программного обеспечения, функционирующего на процессорах 110. При рассмотрении в настоящем документе, встроенное программное обеспечение включает в себя, например, микрокод и/или милликод процессора. Он включает в себя, например, команды аппаратного уровня и/или структуры данных, используемые в реализации высокоуровневого машинного кода. В одном предпочтительном варианте он включает в себя, например, проприетарный код, обычно поставляемый как микрокод, который включает в себя выверенное программное обеспечение или микрокод, специфичный для используемого

оборудования и управляющий доступом операционной системы к оборудованию системы.

Центральные процессоры 110 являются физическими процессорными ресурсами, выделенными логическим разделам. Конкретно, каждый логический раздел 112 имеет один или несколько логических процессоров, каждый из которых представляет собой, полностью или частично, выделенный разделу физический процессор 110. Логические процессоры конкретного раздела 112 могут быть либо выделены разделу таким образом, что базовый процессорный ресурс 110 резервируется для данного раздела, либо быть используемыми совместно с другим разделом таким образом, что базовый процессорный ресурс является потенциально доступным другому разделу. В одном из вариантов, один или несколько из ЦП включают в себя описанные в настоящем документе аспекты средства 130 конфигурации архитектурного режима.

Подсистема 111 ввода/вывода направляет поток информации между устройствами 106 ввода-вывода и основной памятью 104. Эта подсистема соединена с центральным вычислительным комплексом в том плане, что она может быть как частью центрального вычислительного комплекса, так и быть выполненной отдельной от него. Подсистема I/O освобождает центральные процессоры от задач сообщения непосредственно с устройствами ввода-вывода и позволяет обработке данных продолжаться одновременно с обработкой ввода/вывода.

Для обеспечения связи подсистема I/O использует коммуникационные адаптеры I/O.

Существуют различные типы коммуникационных адаптеров, в том числе, например, каналы, адаптеры I/O, платы протокольной управляющей информации (PCI), платы Ethernet, платы интерфейса хранения малых вычислительных машин (SCSI) и т.д. В описанном здесь конкретном примере коммуникационные адаптеры I/O являются каналами, и поэтому, подсистема I/O в настоящем документе называется канальной подсистемой. Тем не менее, данный пример является только одним из многих. Также могут использоваться и другие типы подсистем I/O.

Подсистема I/O использует один или несколько трактов ввода/вывода в качестве коммуникационных каналов при управлении потоком информации к устройствам 106 ввода-вывода или от них. В этом конкретном примере эти тракты называются канальными трактами, поскольку коммуникационные адаптеры являются каналами.

Другой пример вычислительного окружения для охвата и использования одного или нескольких аспектов средства CAM описан со ссылками на фиг. 1Б. В этом примере вычислительное окружение 150 включает в себя центральный вычислительный комплекс 152, предоставляющий поддержку виртуальной машине. CPC 152 соединен с одним или несколькими устройствами 106 ввода/вывода (I/O) через одно или несколько устройств 108 управления. Центральный вычислительный комплекс 152 включает в себя, например, память 154 процессора (известную также под названием оперативная память, основная память, центральная память), соединенную с одним или несколькими центральными процессорами (известными также под названием центральных вычислительных устройств (ЦП)) 110, и с подсистемой 111 ввода/вывода.

Память процессора 154 включает в себя, например, одну или несколько виртуальных машин 162, а также встроенное программное обеспечение 163 процессора, который включает в себя хост-гипервизор 164 и другое встроенное программное обеспечение 165 процессора. Один пример хост-гипервизора 164 представлен z/VM®, предлагаемым International Business Machines Corporation, Армонк, Нью-Йорк.

Поддержка виртуальной машины со стороны CPC предоставляет возможность

управления большим количеством виртуальных машин 162, каждая из которых способна к хостингу гостевой операционной системы 172, такой как Linux®. Каждая виртуальная машина 162 способна к функционированию в качестве отдельной системы. Это означает, что каждая виртуальная машина может быть независимо сброшена, может служить хостом для гостевой операционной системы и может работать с различными программами 120. Операционная система или прикладная программа, работающая в виртуальной машине, представляется как имеющая доступ к полной системе, но в действительности, только ее часть является доступной. Linux является зарегистрированной торговой маркой Linus Torvalds в Соединенных Штатах, в других странах, или в обеих юрисдикциях.

В этом конкретном примере модель виртуальных машин является моделью V=V, в которой абсолютная или реальная память виртуальной машины поддерживается виртуальной памятью хоста вместо реальной или абсолютной памяти. Каждая виртуальная машина имеет пространство виртуальной линейной памяти. Физические ресурсы принадлежат хосту 164, и совместно используемые физические ресурсы по мере необходимости диспетчеризируются посредством хоста к гостевым операционным системам для удовлетворения их вычислительных потребностей.

Такая модель виртуальной машины V=V (то есть, гость со страничной организацией) подразумевает, что взаимодействиями между гостевыми операционными системами и физическими совместно используемыми ресурсами машины управляет хост, поскольку большое количество гостей обычно исключает для хоста возможность простого разделения и присвоения аппаратных ресурсов сконфигурированным гостям. Один или несколько аспектов модели V=V, кроме того, описаны в публикации IBM® под названием «z/VM: Выполнение гостевых операционных систем» (Running Guest Operating Systems), IBM® публикация № SC24-5997-02, октябрь 2001.

Центральные процессоры 110 являются физическими процессорными ресурсами, которые присваиваются виртуальной машине. Например, виртуальная машина 162 включает в себя один или несколько логических процессоров, каждый из которых представляет, полностью или частично, физический процессорный ресурс 110, который может быть динамическим образом выделен виртуальной машине. Виртуальными машинами 162 управляет хост 164.

В одном предпочтительном варианте осуществления аппаратное оборудование/встроенное программное обеспечение хоста (например, z/VM®) и процессора (например, System z) взаимодействуют друг с другом управляемым совместным способом с целью обработки операций гостевой операционной системы V=V без необходимости в передаче управления между гостевой операционной системой и хостом. Гостевые операции могут быть выполнены непосредственно, без вмешательства хоста, с помощью средства, позволяющего командам выполняться в режиме интерпретации для гостя с режимом записи в память со страничной организацией. Это средство предоставляет команду запуска выполнения в режиме интерпретации (Start Interpretive Execution) (SIE), которую хост может выдавать путем назначения блока управления, называемого описанием состояния, который содержит состояние и управляющие воздействия гостя (виртуальной машины), такие как управляющие элементы выполнения и управляющие элементы режима. Команда переводит машину в режим интерпретационного выполнения, в котором гостевые команды и прерывания обрабатываются непосредственно до тех пор, пока не возникает состояние, требующее вмешательства хоста. Когда такое состояние возникает, выполнение в режиме интерпретации заканчивается и, либо реализуется прерывание хоста, либо команда SIE завершает сохранение деталей

возникшего состояния, это последнее действие вызывают перехватом. Один пример выполнения в режиме интерпретации описан в работе «Система/370 - Расширенная архитектура/Выполнение в режиме интерпретации» (System/370 Extended Architecture/ Interpretive Execution), IBM публикация №SA22-7095-01, сентябрь 1985.

Конкретно, в одном предпочтительном варианте осуществления средство выполнения в режиме интерпретации предоставляет команду для выполнения виртуальных машин. Эта команда, называемая запуском выполнения в режиме интерпретации (Start Interpretative Execution) (SIE), выпускается хостом, создающим гостевое окружение выполнения. Хост является управляющей программой, которая непосредственно управляет реальной машиной, а гость относится к любой виртуальной или интерпретируемой машине. Машина переводится в режим интерпретационного выполнения посредством хоста, который выдает команду SIE. В этом режиме машина предоставляет функции выбранной архитектуры (например, z/Архитектуры, ESA/390). Функции включают в себя, например, среди прочего, выполнение привилегированных и проблемных программных команд, трансляцию адресов, обработку прерываний и согласование по времени. Машина, как говорят, интерпретирует функции, которые она выполняет в контексте виртуальной машины.

Команда SIE имеет операнд, называемый описанием состояния, который включает в себя информацию, относящуюся к текущему состоянию гостя. Когда выполнение SIE заканчивается, представляющая состояние гостя информация, включая сюда PSW гостя, сохраняется в описании состояния прежде, чем управление будет возвращено хосту.

Архитектура выполнения в режиме интерпретации предоставляет режим записи в память для абсолютной памяти, называемый режимом записи в память со страничной организацией. В режиме записи в память со страничной организацией динамическая трансляция адресов на уровне хоста используется для отображения гостевой основной памяти. Хост имеет способность к распределению реальной памяти гостей с режимом записи в память со страничной организацией по применимым пакетам в произвольных местоположениях в основной памяти хоста при помощи DAT хоста, и к разбиению гостевых данных на страницы для вспомогательной памяти. Такая техника обеспечивает гибкость при выделении реальных машинных ресурсов и, в то же время, сохраняет для гостя ожидаемое внешнее представление непрерывного диапазона абсолютной памяти.

Окружение виртуальной машины может дважды вызывать DAT для применения: сначала на гостевом уровне, для трансляции гостевого виртуального адреса посредством управляемых гостем трансляционных таблиц в гостевой действительный адрес, а затем, для гостя со страничной организацией, на уровне хоста, для трансляции соответствующего виртуального адреса хоста в действительный адрес хоста.

В определенных случаях, хосту приходится вмешиваться в обычно делегируемые машине операции. С этой целью описание состояния включает в себя управляющие элементы, задаваемые хостом для «прерывания» или перехвата особых состояний. Биты управления перехватом запрашивают возвращение машиной управления к имитации хоста при встрече с конкретными гостевыми командами. Интервенционные управляющие элементы захватывают основные сведения по активированному состоянию в PSW таким образом, что хост может предоставить для гостя прерывание, которое он удерживает отложенным. В то время как интерпретация продолжается, интервенционные управляющие элементы могут быть асинхронным образом заданы посредством хоста на другом реальном процессоре. Машина периодически повторно выбирает управляющие элементы из памяти таким образом, что распознаются обновленные значения. Гостевые прерывания, таким образом, могут быть сделаны отложенными

без преждевременного нарушения интерпретации.

В одном предпочтительном варианте настоящего изобретения, управляющие элементы режима в описании состояния задают выполнение гостя в режиме ESA/390 или z/Архитектуры, а также выбирают один из нескольких способов представления гостевой основной памяти гостевой виртуальной машины в памяти хоста. Согласно одному предпочтительному варианту настоящего изобретения, в управляющем элементе состояния предоставляется бит управления для выбора между пребыванием гостя в первом или втором архитектурном режиме (например, z/Архитектуре или ESA/390, соответственно). Согласно другому предпочтительному варианту настоящего изобретения, две различные команды могут предоставлять хосту способность по созданию первой и второй гостевой виртуальной машины, например, могут быть предоставлены различные команды SIEz и SIEe для запуска гостевых машин в режиме z/Архитектуры и ESA/390, соответственно.

Команда SIE прогоняет диспетчеризированный управляющей программой виртуальный сервер до тех пор, пока либо интервал времени сервера не оказывается использованным, либо сервер не попытается выполнить операцию, которую не могут виртуализировать аппаратные средства, или для выполнения которой управляющая программа должна восстановить свое управление. На этом этапе выполнение команды SIE заканчивается и управление возвращается к управляющей программе, которая либо имитирует команду, либо переводит виртуальный сервер в состояние принудительного ожидания. По завершении, управляющая программа вновь вводит виртуальный сервер в план для выполнения, и цикл вновь запускается. Таким образом, полные возможности и скорость ЦП являются доступными для виртуального сервера. Перехвату подвергаются только те привилегированные команды, которые требуют помощи со стороны управляющей программы или проверки ею их правильности. Такие перехваты SIE, как они известны, также используются управляющей программой для накладывания ограничений на операции, которые виртуальный сервер может выполнять на физическом устройстве.

Более подробная информация относительно SIE изложена в работе «Интерпретирующе-выполняющая архитектура ESA/390, основания для VM/ESA» (ESA/390 interpretive-execution architecture, foundation for VM/ESA), Осисек и др. (Osisek et al), IBM Systems Journal, том 30, №1, январь 1991, стр. 34-51.

Другой пример вычислительного окружения для охвата и использования одного или нескольких аспектов средства конфигурации архитектурного режима описан со ссылками на фиг. 2. В этом примере вычислительное окружение 200 включает в себя неразбитое на разделы окружение, которое может быть сконфигурировано для нескольких архитектурных режимов, в том числе, z/Архитектуры и ESA/390. Это окружение включает в себя, например, процессор (центральное вычислительное устройство - ЦП) 202, который включает в себя, например, средство 204 конфигурации архитектурного режима, а также один или несколько кэшей 206. Процессор 202 коммуникативно соединен с имеющим один или несколько кэшей 210 участком 208 памяти и с подсистемой 212 ввода/вывода (I/O). Подсистема I/O 212 коммуникативно соединена с внешними устройствами 214 ввода-вывода, которые могут включать в себя, например, устройства ввода данных, датчики и/или устройства вывода, такие как дисплеи. Другой предпочтительный вариант настоящего изобретения вычислительного окружения для охвата и использования одного или нескольких аспектов средства конфигурации архитектурного режима описан со ссылками на фиг. 3А. В этом примере вычислительное окружение 300 включает в себя, например, предназначенное для исходной среды

центральное вычислительное устройство (ЦП) 302, память 304, а также одно или несколько устройств ввода-вывода и/или интерфейсов 306, соединенных друг с другом, например, через одну или несколько шин 308 и/или других присоединений. Например, вычислительное окружение 300 может включать в себя процессор PowerPC или сервер Power Systems, предлагаемые International Business Machines Corporation, Армонк, Нью-Йорк, HP Superdome с процессорами Intel Itanium II, предлагаемые Hewlett Packard, Пало-Альто, Калифорния, и/или другие машины, основанные на архитектурах, предлагаемых International Business Machines Corporation, Hewlett Packard, Intel, Oracle или другими. Intel и Itanium являются торговыми марками или зарегистрированными торговыми марками Intel Corporation или ее филиалов в Соединенных Штатах и других странах.

Предназначенное для исходной среды центральное вычислительное устройство 302 включает в себя один или несколько предназначенных для исходной среды регистров 310, таких как один или несколько регистров общего назначения и/или один или несколько регистров особого назначения, используемых в процессе обработки в пределах окружения, а также средство 311 конфигурации архитектурного режима. Эти регистры включают в себя информацию, представляющую состояние окружения на какой-либо конкретный момент времени.

Кроме того, предназначенное для исходной среды центральное вычислительное устройство 302 выполняет команды и код, которые сохраняются в памяти 304. В одном конкретном примере центральное вычислительное устройство выполняет код 312 эмулятора, сохраняемый в памяти 304. Этот код активирует вычислительное окружение, сконфигурированное в одной архитектуре, для эмуляции одной или нескольких других архитектур. Например, код 312 эмулятора позволяет основанным на отличных от z/Архитектуры архитектурах машинам, таким как процессоры PowerPC, серверы Power Systems, серверы HP Superdome или другие, эмулировать z/Архитектуру (и/или ESA/390) и выполнять программное обеспечение и команды, разработанные на основе z/Архитектуры.

Более подробная информация относительно кода 312 эмулятора приведена со ссылками на фиг. 3Б. Сохраняемые в памяти 304 гостевые команды 350 содержат команды программного обеспечения (например, коррелирующие с машинными командами), которые были разработаны для выполнения в архитектуре, отличной от таковой для предназначенного для исходной среды ЦП 302. Например, гостевые команды 350 могут быть разработаны для выполнения на процессоре 202 z/Архитектуры, но вместо этого, эмулируются на предназначенном для исходной среды ЦП 302, который может быть представлен, например, процессором Intel Itanium II. В одном примере, код 312 эмулятора включает в себя подпрограмму 352 выборки команд для получения одной или нескольких гостевых команд 350 из памяти 304 и, опционально, для предоставления локальной буферизации для полученных команд. Данный код также включает в себя подпрограмму 354 трансляции команд для выявления типа полученной гостевой команды и для трансляции гостевой команды в одну или несколько соответствующих команд 356 исходной среды. Такая трансляция включает в себя, например, идентификацию функции, которая подлежит выполнению посредством гостевой команды, и выбор предназначенной для исходной среды команды (команд) для выполнения этой функции.

Кроме того, код 312 эмулятора включает в себя подпрограмму 360 управления эмуляцией для принуждения команд исходной среды к выполнению. Подпрограмма 360 управления эмуляцией может принудить предназначенное для исходной среды ЦП 302 к выполнению подпрограммы команд исходной среды, которые эмулируют одну

или несколько ранее полученных гостевых команд и, в конце такого выполнения, возвращают управление подпрограмме выборки команд для эмуляции получения следующей гостевой команды или группы гостевых команд. Выполнение команд 356 исходной среды может включать в себя загрузку данных в регистр из памяти 304, сохранение данных обратно в память из регистра, или выполнение арифметической или логической операции некоторого типа, как задано подпрограммой трансляции.

Каждая подпрограмма, например, реализуется в программном обеспечении, сохраняемом в памяти и выполняемом посредством предназначенного для исходной среды центрального вычислительного устройства 302. В других примерах, одна или несколько из числа подпрограмм или операций, реализуются во встроенном программном обеспечении, аппаратных средствах, программном обеспечении или в некоторой комбинации из этих средств. Регистры эмулированного процессора могут быть эмулированы с помощью регистров 310 предназначенного для исходной среды ЦП или при помощи местоположений в памяти 304. В предпочтительных вариантах осуществления гостевые команды 350, команды 356 исходной среды и код 312 эмулятора могут находиться в той же самой памяти или могут быть рассредоточены среди различных запоминающих устройств.

Описанные выше вычислительные окружения являются только примерами пригодных к использованию вычислительных окружений. Могут быть использованы и другие окружения, в том числе, но не ограничиваясь, другие неразбитые на разделы окружения, другие разбитые на разделы окружения и/или другие эмулированные окружения, варианты осуществления не ограничиваются каким-либо окружением.

Согласно одному или несколькими аспектам средство конфигурации архитектурного режима (CAM) инсталлируется в одном или нескольких процессорах (например, центральных вычислительных устройствах) вычислительного окружения для управления реконфигурированием окружения. Например, когда средство CAM инсталлировано в вычислительном окружении, поддерживающем несколько архитектурных режимов, вычислительное окружение реконфигурируется таким образом, что ограничивается использование одного или нескольких аспектов по меньшей мере одного из архитектурных режимов.

Один конкретный пример средства конфигурации архитектурного режима представлен средством конфигурации архитектурного режима (CZAM) для z/Архитектуры. Инсталляция CZAM обозначается, например, индикатором инсталляции средства, например битом 138, которому задается значение, например, единица. В одном конкретном примере, когда биту 138 задано значение единица, средство CZAM инсталлировано, и, когда инсталлировано, нормальный сброс и чистый сброс переводят конфигурацию в архитектурный режим z/Архитектуры. Таким образом, биту функции, например биту 2, указывающему на активность архитектурного режима z/Архитектуры, также задается значение единица, в данном примере.

За счет инсталляции CZAM вычислительное окружение (например, одиночный процессор, логический раздел, виртуальный гость и т.д.) реконфигурируется таким образом, что более не поддерживаются один или несколько аспектов выбранной архитектуры, например ESA/390. Различные более не поддерживающиеся аспекты и/или некоторые затронутые установкой CZAM процессы описаны ниже.

Хотя в описанных в настоящем документе вариантах осуществления совокупность архитектурных режимов включает в себя устаревшую архитектуру (например, ESA/390) и расширенную архитектуру (например, z/Архитектуру), и аспекты устаревшей архитектуры ESA/390 более не поддерживаются, другие варианты осуществления могут

включать в себя другие архитектуры. ESA/390 и z/Архитектура являются только примерами.

Один из затронутых инсталляцией CZAM процессов представлен процессом включения. Для описания того, как этот процесс затрагивается, первоначально описан процесс включения для окружения, поддерживающего множественные архитектурные конфигурации и не включающего в себя функцию CZAM, со ссылками на фиг. 4А-4Б, а затем процесс включения для окружения, сконфигурированного для множественных архитектурных конфигураций, и включающего в себя средство CZAM, описан со ссылками на фиг. 6А-6Б. Включение для системы содержит, например, этапы запуска системы и инициации последовательности загрузки или других средств инициирования операций в системе. Это может соответствовать физическому включению, аппаратному сбросу и/или виртуальному включению (например, в эмулированной системе, виртуальной машине или гостевом окружении).

При обращении первоначально к фиг. 4А, при условии, что процессор вычислительного окружения был включен, а кнопка оператора, например кнопка нормальной загрузки или кнопка чистой загрузки, была активирована, процессор входит в состояние загрузки и задает вычислительному окружению конкретный архитектурный режим, например режим ESA/390, ЭТАП 400. Например, выполняется начальная загрузка программы (IPL), такая как начальная загрузка программы (IPL) управляющего слова (CCW) канала, ЭТАП 402. Начальная загрузка программы предоставляет ручные средства для принуждения к считыванию программы из заданного устройства и для инициирования выполнения этой программы. IPL типа CCW инициируется вручную посредством задания средствам управления адресом загрузочного устройства значения в виде четырехзначного числа для обозначения устройства ввода данных, и посредством последующей активации кнопки чистой загрузки или нормальной загрузки для конкретного ЦП.

Активация кнопки чистой загрузки вызывает выполнение чистого сброса на конфигурации, а активация кнопки нормальной загрузки вызывает выполнение начального сброса ЦП на данном ЦП (ЦП, на котором кнопка была активирована), сброс ЦП подлежит распространению на все другие ЦП в конфигурации, а сброс подсистемы выполняется на остаточной части конфигурации. Активация кнопки чистой загрузки или кнопки нормальной загрузки задает архитектурный режим (например, ESA/390).

В загрузочной части операции, после выполнения сброса, этот ЦП входит в состояние загрузки. Данный ЦП не обязательно входит в остановленное состояние в процессе выполнение операций по сбросу. Индикатор загрузки включен, когда ЦП находится в состоянии загрузки.

Впоследствии, инициируется операция считывания из устройства ввода-вывода, указанного посредством средств управления адресом загрузочного устройства. Эффект выполнения программы управления работой канала ввода-вывода состоит в том, как если формат 0 управляющего слова CCW канала, начинающийся в абсолютной местоположении 0 памяти, задает команду считывания с нулевыми значениями битов модификатора, адресом данных ноль, отсчетом байтов 24, с метками команды с признаком цепочки и SLI (интерпретатора логики служб) равными единицам, и всеми другими метками равными нолям.

Когда операция ввода-вывода IPL завершается успешно, идентификационное слово подсистемы для устройства IPL сохраняется в выбранных абсолютных местоположениях памяти (например, ячейках 184-187), нули сохраняются в других выбранных абсолютных

местоположениях памяти (например, местоположениях 188-191), а новое слово состояния программы (PSW) загружается из выбранных абсолютных местоположений памяти (например, местоположений 0-7), ЭТАП 404. Слово состояния программы управляет операциями вычислительного окружения.

Если загрузка PSW является успешной, и не обнаружено каких-либо сбоев машины, данный ЦП покидает состояние загрузки, и индикатор нагрузки выключается. Если управление производительностью задано в положение обработки, ЦП входит в рабочее состояние, и функционирование вычислительного окружения переходит под управление нового слова (PSW) состояния программы, ЭТАП 406. Загруженное вычислительное окружение затем функционирует, ЭТАП 408, как далее описано со ссылками на фиг.4Б.

Согласно фиг. 4Б, загруженное вычислительное окружение инициируется в режиме ESA/390, ЭТАП 420 и, таким образом, операции выполняются в режиме ESA/390, ЭТАП 422. В некоторый момент может быть выполнен запрос на изменение архитектурного режима от ESA/390 к z/Архитектуре. Конкретно, программа отправляет код распоряжения (например, код, обозначающий задание архитектуры) к процессору, который выдает команду процессора (SIGP) обработки сигналов с кодом распоряжения для переключения от режима ESA/390 к режиму z/Архитектуры, ЭТАП 424. Например, используется средство подачи сигнала и ответа ЦП, который включает в себя команду процессора обработки сигналов (описанную ниже) и механизм для интерпретирования кодов распоряжений сервера и реагирования на них, в том числе, на код задания архитектуры. Средство предоставляет возможности связи между ЦП, включая сюда передачу, получение и декодирование набора присвоенных кодов распоряжений, инициирование указанной операции, а также отправку ответа к ЦП обработки сигналов. Посредством использования задания архитектуры архитектурному режиму задается требуемая конфигурация, например z/Архитектура. Более подробная информация по этой обработке приведена ниже.

После этого, производится выявление относительно того, была ли операция SIGP принята, ИНФОРМАЦИОННЫЙ ЗАПРОС 426. На основании кода возврата могут быть диагностированы несколько состояний ошибки, включая сюда указание на «недопустимый параметр», когда было выявлено, что ЦП уже находится в заданном посредством кода архитектурном режиме (то есть, что задание архитектуры представляет собой переключение к текущему режиму как таковому, в отличие от переключения с одного режима к другому режиму). Если SIGP принимается, а задание архитектуры представляет собой допустимую операцию по переключению режима, то все получившие операцию SIGP процессоры вычислительного окружения переходят в режим z/Архитектуры с использованием, например, описанной в настоящем документе обработки задания архитектуры, ЭТАП 428. Однако, если операция SIGP не является допустимой, обозначается ошибка, ЭТАП 430.

Как описано выше, операция включения загружает слово состояния программы. Один предпочтительный вариант настоящего изобретения формата слова состояния программы (PSW) описан со ссылками на фиг. 5. Согласно фиг. 5, в этом примере формат слова состояния программы является форматом ESA/390, за исключением того, что бит 31 показан как EA, как обозначено ниже. В одном предпочтительном варианте осуществления настоящего изобретения слово 500 состояния программы включает в себя, в качестве примера, следующие поля:

Маска PER (R) 502: Бит 1 управляет активацией в ЦП прерываний, связанных с регистрацией (PER) программных событий. Когда бит представлен нолем, никакое событие PER не может вызвать прерывание. Когда бит представлен единицей,

прерывания разрешены согласно битам маски события PER в регистре 9 управления,

Режим DAT (T) 504: Бит 5 управляет наличием скрытой динамической трансляции (DAT) адресов для логических адресов и адресов команд, используемых для получения доступа к памяти. Когда бит представлен нолем, DAT выключена, и логические адреса и адреса команд обрабатываются как действительные адреса. Когда бит представлен единицей, DAT включена, а механизм динамической трансляции адресов вызван.

Маска ввода/вывода (10) 506: Бит 6 управляет активацией в ЦП прерываний для I/O. Когда бит представлен нолем, прерывание I/O не может произойти. Когда бит представлен единицей, прерывания I/O подвергаются воздействию битов маски подкласса прерывания I/O в регистре 6 управления. Когда бит маски подкласса прерывания I/O представлен нолем, прерывание I/O для данного подкласса прерывания I/O не может произойти, когда бит маски подкласса прерывания I/O представлен единицей, прерывание I/O для данного подкласса прерывания I/O может произойти.

Внешняя маска (EX) 508: Бит 7 управляет активацией в ЦП прерываний посредством условий, включаемых во внешний класс. Когда бит представлен нолем, внешнее прерывание не может произойти. Когда бит представлен единицей, внешнее прерывание подвергается воздействию соответствующих битов подкласса внешней маски в регистре 0 управления. Когда бит подкласса представлен нолем, связанные с подклассом условия не могут вызвать прерывание. Когда бит подкласса представлен единицей, прерывание в этом подклассе может произойти.

Ключ PSW (Key) 510: Биты 9-11 формируют ключ доступа для обращения к памяти посредством ЦП. Если обращение является объектом управляемой ключом защиты, ключ PSW согласовывается с ключом защиты памяти при сохранении информации или при выборке информации из защищенного от выборки местоположения. Тем не менее, для каждого из числа операндов Перейти к первичному выражению, Перейти ко вторичному выражению, Перейти с ключом, Перейти с исходным ключом и Перейти с целевым ключом, а также для одного или для обоих операндов Перейти с необязательными спецификациями, вместо ключа PSW используется ключ доступа, заданный как операнд.

Бит 12 512: Этот бит указывает на текущий архитектурный режим. Он имеет значение единица для формата PSW для ESA/390. Для формата PSW для z/Архитектуры это значение бита задается нолем. При пребывании в режиме z/Архитектуры, загрузка расширенной PSW команды (LPSWE) задана для загрузки истинного PSW для z/Архитектуры (которое имеет другой формат, отличный от описанного в настоящем документе формата, и включающий в себя наличие адреса команды в битах 64-127). Тем не менее, загрузка PSW (LPSW) в ESA/390 все еще поддерживается и может быть использована для загрузки формата PSW для ESA/390. Когда LPSW выполняется, а вычислительное окружение находится в режиме z/Архитектуры, процессор расширяет формат PSW для ESA/390 до формата для z/Архитектуры, включая сюда инвертирующий бит 12. Эта операция является обратной к сворачиванию формата PSW для z/Архитектуры, которое операционная система выполняет для создания формата PSW для ESA/390. Таким образом, в вычислительных окружениях, поддерживающих и ESA/390 и z/Архитектуру, когда копия PSW размещается в памяти, операционная система сворачивает полное PSW для z/Архитектуры к размеру и формату PSW для ESA/390. Таким образом другое программное обеспечение с зависимостями от формата PSW может не знать о Z/Архитектуре PSW.

Маска машинного контроля (M) 514: Бит 13 управляет активацией в ЦП прерываний посредством условий машинного контроля. Когда бит представлен нолем, прерывание

машинного контроля не может произойти. Когда бит представлен единицей, прерывания машинного контроля вследствие повреждения системы и повреждения обработки команд являются разрешенными, но прерывания вследствие других условий подкласса машинного контроля подвергаются воздействию битов маски подкласса в регистре 14 управления.

Состояние ожидания (W) 516: Когда бит 14 представлен единицей, ЦП ожидает, то есть, посредством ЦП никакие команды не обрабатываются, но могут иметь место прерывания. Когда бит 14 представлен нолем, выборка и выполнение команды происходят нормальным способом. Когда бит представлен единицей, индикатор ожидания представлен единицей.

Заданный режим (P) 518: Когда бит 15 представлен единицей, ЦП находится в заданном режиме. Когда бит 15 представлен нолем, ЦП находится в супервизорном режиме. В супервизорном режиме все команды являются допустимыми. В заданном режиме являются допустимыми только те команды, которые предоставляют значащую информацию заданной программе, и которые не могут затронуть целостность системы, такие команды называют непривилегированными командами. Команды, которые не являются допустимыми в заданном режиме, называют привилегированными командами. Когда ЦП в заданном режиме пытается выполнить привилегированную команду, распознается исключение по привилегированной операции. Другая группа команд, называемых полупривилегированными командами, выполняется посредством ЦП в заданном режиме только в том случае, когда выполняются специальные проверки полномочий, в противном случае, распознается исключение по привилегированной операции или какое-либо другое программное исключение, в зависимости от того конкретного требования, которое было нарушено.

Управление адресным пространством (AS) 520: Биты 16 и 17 совместно с битом 5 PSW управляют режимом трансляции.

Условный код (CC) 522: Биты 18 и 19 представляют собой два бита условного кода.

Условному коду присваивают значения 0, 1, 2 или 3 в зависимости от результата, полученного при выполнении конкретных команд. Большинство арифметических и логических операций, а также некоторые другие операции присваивают значения условного кода. Команда BRANCH ON CONDITION (перехода по условию) может задавать любой выбор значений условного кода в качестве критерия ветвления.

Программная маска 524: Биты 20-23 представляют собой четыре бита программной маски. Каждый бит связан с программным исключением нижеследующим образом:

Бит программной маски	Программное исключение
20	Переполнение с фиксированной точкой
21	Десятичное переполнение
22	Исчезновение порядка HFP
23	Значимость разряда HFP

Когда бит маски представлен единицей, результатом исключения является прерывание. Когда бит маски представлен нолем, не происходит никакого прерывания. Задание бита маски исчезновения порядка HFP для бита маски значимости разряда HFP также задает способ завершения операции при осуществлении соответствующего исключения.

Расширенный режим адресации (EA) 526: Бит 31 управляет размером исполнительных адресов и образованием исполнительных адресов совместно с битом 32, основным

битом режима адресации. Когда бит 31 представлен нолем, режимом адресации управляет бит 32. Когда оба бита 31 и 32 представлены единицами, задается 64-битовая адресация.

Основной режим адресации (BA) 528: Биты 31 и 32 управляют размером исполнительных адресов и образованием исполнительных адресов. Когда оба бита 31 и 32 представлены нолями, задается 24-битовая адресация. Когда бит 31 представлен нолем, а бит 32 представлен единицей, задается 31-битовая адресация. Когда оба бита 31 и 32 представлены единицами, задается 64-битовая адресация. Бит 31 равный единице и бит 32 равный нолю представляют собой недопустимую комбинацию, вызывающую распознавание исключения по спецификации. Режим адресации не управляет размером адресов PER, или адресов, используемых для получения доступа к DAT, ASN, вызываемому управлению блоком, таблицам связей, входных записей и слежения или спискам доступа или стеку соединений. Управление режимом адресации посредством битов 31 и 32 PSW можно подытожить следующим образом:

PSW:31	PSW:32	Режим адресации
0	0	24-битовый
0	1	31-битовый
1	1	64-битовый

Адрес 530 команды: Биты 33-63 PSW представляют собой адрес команды. Адрес обозначает местоположение крайнего левого байта следующей подлежащей выполнению команды, если ЦП не находится в состоянии ожидания (бит 14 PSW равен единице).

Согласно аспекту изобретения, когда средство конфигурации архитектурного режима, такое как средство конфигурации архитектурного режима (CZAM) для z/Архитектуры инсталлировано и активировано в вычислительном окружении, процесс включения изменяется. Один предпочтительный вариант осуществления настоящего изобретения процесса включения CZAM описан со ссылками на фиг. 6A.

Согласно фиг. 6A, на основании включения процессора вычислительного окружения, вычислительное окружение пребывает в конкретном архитектурном режиме, заданном средством конфигурации архитектурного режима, например режима z/Архитектуры (также называемом ESAME), когда инсталлировано CZAM, ЭТАП 600. Например, выполняется, как описано выше, начальная загрузка (IPL) программы, такая как начальная загрузка (IPL) программы управляющего слова (CCW) канала, ЭТАП 602, и когда операция ввода-вывода IPL завершается успешно, идентификационное слово подсистемы для устройства IPL сохраняется в выбранных местоположениях абсолютной памяти (например, местоположениях 184-187), ноли сохраняются в других выбранных местоположениях абсолютной памяти (например, местоположениях 188-191), и в этом варианте осуществления 16-байтовое новое слово (PSW) состояния программы создается из выбранных местоположений абсолютной памяти (например, местоположений 0-7), ЭТАП 604. Новое 16-байтовое PSW образуется, например, из содержимого выбранного двойного слова памяти (например, местоположений 0-7). Бит 12 двойного слова должен иметь значение единица, в противном случае, может быть обозначена ошибка. (Ошибка может быть представлена распознанным исключением по спецификации, по машинному контролю или индикацией какой-либо другой ошибки.) Биты 0-32 вновь созданного PSW задаются по битам 0-32 выбранного двойного слова, кроме бита 12, значение которого инвертируется. Битам 33-96 вновь созданного PSW задаются нулевые значения. Позиции битов 97-127 вновь созданного PSW инициализируются по битам 33-63

выбранного двойного слова.

В одном предпочтительном варианте осуществления настоящего изобретения поля PSW, которые должны быть загружены командой, не проверяются на допустимость перед их загрузкой. В одном предпочтительном варианте осуществления настоящего изобретения бит 12 PSW проверяется на допустимость. В еще предпочтительном варианте осуществления настоящего изобретения все поля проверяются на допустимость. В другом предпочтительном варианте осуществления любые биты, не проверенные до загрузки PSW, проверяются на допустимость после того, как PSW было инициализировано, и процессор может указать на ошибку (например, путем активации распознанного исключения по спецификации, машинному контролю или индикации какой-либо другой ошибки.)

Вычислительное окружение входит в рабочее состояние, и функционирование вычислительного окружения переходит под управление нового слова (PSW) состояния программы, ЭТАП 606. Загруженное вычислительное окружение затем функционирует, ЭТАП 608, как далее описано со ссылками на фиг. 6Б.

Согласно фиг. 6Б, загруженное вычислительное окружение инициируется в режиме z/Архитектуры, ЭТАП 620 и, таким образом, операции выполняются в режиме z/Архитектуры, ЭТАП 622. Не является необходимым какое-либо переключение режима, и обработка непосредственно продолжается обработкой в режиме z/Архитектуры. Тем самым, в одном предпочтительном варианте осуществления настоящего изобретения следующие этапы не являются необходимыми: Операция процессора (SIGP) обработки сигналов по переключению из режима ESA/390 к режиму z/Архитектуры, выявление того, является ли операция SIGP допустимой операцией, переход к z/Архитектуре, в случае допустимости операции, или индикация ошибки в случае недопустимости операции SIGP.

Все процессоры вычислительного окружения (то есть, конфигурируемого окружения, например одиночный процессор, логический раздел, гостевая VM), находятся в режиме z/Архитектуры без выполнения вышеупомянутых этапов. Таким образом, как описано в настоящем документе, согласно одному аспекту способность к загрузке или к включению в режиме ESA/390 удаляется из вычислительного окружения, сконфигурированного для обеих ESA/390 и z/Архитектуры. Конкретно, хотя вычислительное окружение сконфигурировано для поддержки множественных архитектур, предоставляется инструмент ограничения конкретных аспектов по меньшей мере одной из сконфигурированных архитектур, одним из аспектов которого является способность к включению в этой архитектуре.

В одном или нескольких предпочтительных вариантах осуществления включение в режиме z/Архитектуры предоставляет механизм для задания одного элемента из числа: (1) логического раздела (гость-1), и (2) логического раздела и гостя-2, в качестве загружаемого и сбрасываемого в режиме z/Архитектуры без потребности в загрузке в режиме ESA/390. Этот признак может быть инсталлирован безусловным образом или под управлением переключателя конфигураций.

Последовательность загрузки относительно инициализации PSW изменяется. Например, в конце IPL, загружается PSW IPL в абсолютные местоположения 0-7. Как делается в настоящее время, когда условием сброса является ESA/390, бит 12 представлен единицей, что обеспечивает допустимость ESA/390 IPL PSW, и программа продолжает выполнять команды в архитектурном режиме ESA/390. При инсталлированном CZAM условием сброса является z/Архитектура, бит 12 по прежнему равен единице, что делает допустимым ESA/390 IPL PSW, но бит 12 инвертируется в процессе формирования 16-

байтового текущего PSW z/Архитектуры, как задано выше.

В дополнение к процессу включения, инсталляция средства конфигурации архитектурного режима также может изменить или затронуть и другие процессы, логики работы и/или операции. Эти затронутые процессы, логики работы и/или операции являются специфичными для режимов ESA/390 и z/Архитектуры. Однако подобные и/или различные процессы могут быть затронуты и при других типах архитектуры. Типовые процессы, логики работы и/или операции, которые могут быть затронуты в одном или нескольких вариантах осуществления, включают в себя, например:

(1) Предоставление возможности переключения из режима в тот же режим (например, из режима z/Архитектуры в режим z/Архитектуры) без генерации ошибки (или с игнорированием ошибки). Таким образом, процессор может дать команду SIGP для переключения в режим z/Архитектуры и, если он уже пребывает в этом режиме, то не генерируется какой-либо ошибки. Ранее, попытка переключения к режиму, соответствующему текущему режиму, генерировала ошибку.

(2) Отключение переключения к режиму ESA/390. На основе инсталлирования и активации CZAM переключение к ESA/390 отключается, и теперь генерирует ошибку. Переключение назад к ESA/390 предотвращается путем проверки бита 12 PSW и принятия исключения, если бит 12 не задан для указания на режим z/Архитектуры (что представлено битом 12 со значением «1» в памяти, инвертированном к значению «0» для представления z/Архитектуры в PSW, когда PSW для ESA/390 преобразуют к допустимому для z/Архитектуры PSW).

(3) Изменение функционирования команды загрузки PSW для ограничения обработки бита 12. Если инсталлировано средство конфигурации архитектурного режима для z/Архитектуры, команда загрузки PSW распознает исключение по спецификации, если бит 12 ее второго операнда не является единицей. Команда загрузки PSW загружает биты 0-32 своего второго операнда, за исключением бита 12, который имеет инвертированное значение, а биты 33-63 операнда в качестве, соответственно, битов 0-32 и 97-127 текущего PSW, что задает битам 33-96 текущего PSW нулевые значения.

Более подробная информация относительно команды загрузки PSW описана со ссылками на фиг. 7. В одном предпочтительном варианте осуществления настоящего изобретения команда 700 загрузки PSW включает в себя поле 702 кода операции, включающее в себя код операции для указания на операцию загрузки PSW, основное поле (B2) 704, и поле (D2) 706 смещения. Содержимое общего регистра, на которое указывает поле B2, добавляется к содержимому поля D2 для формирования адреса второго операнда в памяти (называемого вторым адресом операнда).

При выполнении команды загрузки PSW текущее PSW заменяется 16-байтовым PSW, образованным из содержимого двойного слова в местоположении, на которое указывает второй адрес операнда.

Бит 12 двойного слова должен иметь значение единица, в противном случае, в зависимости от модели, может быть распознано исключение по спецификации. Если инсталлировано средство конфигурации архитектурного режима для z/Архитектуры, то исключение по спецификации распознается в том случае, если бит 12 двойного слова не является единицей. Биты 0-32 двойного слова, за исключением бита 12, который имеет инвертированное значение, размещаются в положениях 0-32 текущего PSW. Биты 33-63 двойного слова размещаются в положениях 97-127 текущего PSW. Битам 33-96 текущего PSW задаются нулевые значения.

Функция преобразования в последовательную форму и синхронизации контрольной точки выполняется прежде или после выборки операнда, и вновь после завершения

операции.

Операнд должен быть назначен на границе двойного слова, в противном случае, распознается исключение по спецификации. Если бит 12 операнда представлен нолем, в зависимости от модели, может быть распознано исключение по спецификации.

Поля PSW, которые должны быть загружены посредством команды, не проверяются на допустимость перед их загрузкой, за исключением проверки бита 12. Однако сразу после загрузки распознается исключение по спецификации и происходит прерывание программы, когда для недавно загруженного PSW является истинным любое из последующих условий:

- Любой из битов 0, 2-4, 12 или 24-30 представлен единицей.
- Биты 31 и 32 оба равны нолю, но не все биты 97-103 представлены нолями.
- Биты 31 и 32 представлены единицей и нолем, соответственно.

В этих случаях операция завершается, а получающийся код длины команды равен 0.

Операция подавляется на всех исключениях адресации и защиты.

Получающийся условный код: код задается, как указано в новом загруженном PSW.

Программные исключения:

- Доступ (выборка, операнд 2)
- Привилегированная операция
- Спецификация

Примечание по программированию: Второй операнд должен иметь формат ESA/390 PSW. Если бит 12 операнда представлен нолем, исключение по спецификации распознается в процессе или после выполнения команды LOAD PSW.

Более подробная информация относительно PSW приведена в статье «Разработка и реквизиты z/Архитектуры» (Development and Attributes of z/Architecture), Пламбек и др. (Plambeck et al), IBM J. Res. & Dev., Vol. 46, №4/5, июль/сентябрь 2002.

В дополнение к вышеупомянутым процессам, операциям и/или логикам работы, которые могут быть изменены вследствие инсталляции средства конфигурации архитектурного режима, в одном или нескольких вариантах осуществления также может быть изменен режим сброса, как объяснено ниже. (4) Изменение режима сброса (например, для сброса, чистого сброса и других действий для сброса). Когда инсталлировано средство CZAM, сброс ЦП задает архитектурному режиму режим z/Архитектуры, если это вызывается посредством активации, например посредством кнопки нормальной загрузки.

Имеется несколько функций сброса, которые включены в качестве составной части в режимы ESA/390 и z/Архитектуры, включая сюда, например, сброс ЦП, начальный сброс ЦП, сброс подсистемы, чистый сброс и сброс включения, каждая из которых описана ниже.

Сброс ЦП

Сброс ЦП предоставляет средства для очистки указателей проверки оборудования, а также какой-либо проистекающей непредсказуемости в состоянии ЦП с наименьшим возможным объемом уничтожаемой информации. Конкретно, такой сброс используют для очистки условий проверки, когда состояние ЦП должно быть сохранено для анализа или возобновления операции. Если средство конфигурации архитектурного режима (CZAM) для z/Архитектуры не инсталлировано, сброс ЦП задает архитектурному режиму режим ESA/390, если это вызывается активацией кнопки нормальной загрузки (средство оператора). Когда средство CZAM инсталлировано, сброс ЦП задает архитектурному режиму режим z/Архитектуры, если это вызывается активацией кнопки

нормальной загрузки. Когда сброс ЦП задает режим ESA/390, он сохраняет текущее PSW таким образом, что PSW может быть восстановлено посредством распоряжения на задание архитектуры от процессора обработки сигналов, которое изменяет архитектурный режим обратно к z/Архитектуре.

В одном предпочтительном варианте осуществления настоящего изобретения сброс ЦП вызывает следующие действия:

1. Выполнение текущей команды или другой последовательности обработки, такой как прерывание, завершается, и все условия прерывания программы и прерывания обслуживания вызова супервизора очищаются.

2. Очищаются какие бы то ни было незаконченные условия внешнего прерывания, которые являются локальными для данного ЦП. Перемещаемые условия внешнего прерывания не очищаются.

3. Очищаются какие бы то ни было незаконченные условия прерывания машинного контроля и индикации ошибки, которые являются локальными для данного ЦП, а также любые состояния контрольного останова. Перемещаемые условия прерывания машинного контроля не очищаются. Любое условие машинного контроля, о котором сообщается всем ЦП в конфигурации, и которое было сделано отложенным для ЦП, считается для данного ЦП локальным.

4. Очищаются все копии выбранных с упреждением команд или операндов. Кроме того, очищаются любые результаты, которые подлежат сохранению вследствие выполнения команд в текущем интервале контрольной точки.

5. Ассоциативный буфер ART (трансляции регистра доступа) и ассоциативный буфер трансляции очищаются от записей.

6. Если сброс вызывается активацией кнопки нормальной загрузки на любом ЦП в конфигурации, происходят следующие действия:

А. Когда средство CZAM не инсталлировано, архитектурный режим ЦП (и всех других ЦП в конфигурации вследствие выполненных ими начального сброса ЦП или сбросов ЦП) изменяется от режима z/Архитектуры к режиму ESA/390. Если средство CZAM инсталлировано, архитектурному режиму ЦП (и всех других ЦП в конфигурации вследствие выполненных ими начального сброса ЦП или сбросов ЦП) задается режим z/Архитектуры.

Б. Когда средство CZAM не инсталлировано, текущее PSW сохраняется для последующего использования посредством распоряжения на задание архитектуры от процессора обработки сигналов, восстанавливающего режим z/Архитектуры.

В. Когда средство CZAM не инсталлировано, текущее PSW изменяется с 16 байтов до восьми байтов. Биты восьмибайтового PSW задаются следующим образом: биты 0-11 и 13-32 задаются равными тем же самым битам 16-байтового PSW, бит 12 задается равным единице, а биты 33-63 задаются равными битам 97-127 16-байтового PSW.

Сброс ЦП, вызванный активацией системы кнопкой нормального сброса или распоряжением на сброс ЦП от процессора обработки сигналов, а также любой сброс ЦП в режиме ESA/390, не затрагивают захваченный словом PSW z/Архитектуры регистр (то есть, слово PSW, сохраненное при последнем переходе ЦП из режима z/Архитектуры к режиму ESA/390 вследствие распоряжения на задание архитектуры с кодом 0, или сброса ЦП вследствие активации кнопкой нормальной загрузки).

7. ЦП пребывает в остановленном состоянии по завершении действий 1-6. Когда за функцией сброса на таком ЦП следует последовательность IPL типа CCW, по завершении функции сброса ЦП входит в состояние загрузки, и не обязательно входит в остановленное состояние в процессе выполнения операции по сбросу. Когда за функцией

сброса на таком ЦП следует управляемая списком последовательность IPL, ЦП входит в рабочее состояние и не обязательно входит в остановленное состояние в процессе выполнения операции по сбросу.

Регистры, содержание памяти и состояние внешних по отношению к ЦП условий остаются в результате сброса ЦП неизменными. Однако последующее содержимое регистра, местоположения или состояния являются непредсказуемыми, если происходит операция, изменяющая содержимое во время сброса. Удерживаемая посредством ЦП при выполнении команды PERFORM LOCKED OPERATION (выполнения заблокированной операции) блокировка не разблокируется при сбросе ЦП.

Когда функция сброса в ЦП инициируется во время выполнения ЦП команды I/O или выполнения прерывания I/O, текущая операция между ЦП и канальной подсистемой может быть или может не быть завершена, и результирующее состояние ассоциированной функции канальной подсистемы может быть непредсказуемым.

Примечания по программированию:

1. Большинство операций, которые могут изменить состояние, условие или содержимое поля, не могут произойти, когда ЦП находится в остановленном состоянии. Однако некоторые функции процессора обработки сигналов и некоторые функции оператора могут изменить эти поля. Для устранения возможности потери поля при выпуске сброса ЦП, ЦП должен быть остановлен, и никакие функции оператора не должны происходить.

2. Если архитектурный режим изменяется на режим ESA/390, и бит 31 текущего PSW равен единице, PSW является недопустимым.

Начальный сброс ЦП

Начальный сброс ЦП предоставляет функции сброса ЦП совместно с инициализацией программируемых регистров текущего PSW, захваченного PSW z/Архитектуры, таймера ЦП, компаратора часов, префикса, управления прерыванием адресов событий, управления с плавающей точкой и времени (TOD) суток. Если средство CZAM не инсталлировано, начальный сброс ЦП задает архитектурному режиму режим ESA/390, если это вызывается активацией кнопки нормальной загрузки. Когда средство CZAM инсталлировано, начальный сброс ЦП задает архитектурному режиму режим z/Архитектуры, если это вызывается активацией кнопки нормальной загрузки.

Начальный сброс ЦП сочетает функции сброса ЦП со следующими функциями очистки и инициализации:

1. Когда средство CZAM не инсталлировано, если сброс вызывается активацией кнопки нормальной загрузки, архитектурному режиму ЦП (и всех других ЦП в конфигурации) задается режим ESA/390. В противном случае, если средство CZAM инсталлировано, архитектурному режиму ЦП (и всех других ЦП в конфигурации) задается режим z/Архитектуры.

2. Содержимому программируемых регистров текущего PSW, захваченного PSW z/Архитектуры, префикса, таймера ЦП, компаратора часов и TOD задаются нулевые значения. Когда последовательность IPL следует за функцией сброса на данном ЦП, содержимое PSW не обязательно задается нулевым.

Содержимому регистров управления задаются их начальные для z/Архитектуры значения. Значения всех 64 битов регистров управления задаются независимо от пребывания ЦП в архитектурном режиме ESA/390 или z/Архитектуры.

4. Содержимому регистра управления с плавающей точкой задается нулевое значение.

5. Содержимое регистра адреса события прерывания инициализируется к шестнадцатеричному числу.

Эти функции очистки и инициализация включают в себя проверку правильности.

Задание текущему PSW нулевого значения при нахождении ЦП в архитектурном режиме ESA/390 в конце операции приводит к недопустимому PSW, поскольку бит PSW 12 в этом режиме должен быть равен единице. Таким образом, в этом случае, если ЦП пребывает в рабочем состоянии после сброса без предварительного введения нового PSW, распознается исключение по спецификации.

Сброс подсистемы

Сброс подсистемы предоставляет средства для очистки перемещаемых условий прерывания, а также для вызова сброса системы I/O.

Чистый сброс

Чистый сброс вызывает выполнение начального сброса ЦП и сброса подсистемы, а также, дополнительно, очищает или инициализирует все местоположения памяти и регистры во всех ЦП в конфигурации, за исключением часов TOD. Такая очистка является полезной при отладке программ и при обеспечении пользовательской конфиденциальности. Чистый сброс также разблокирует все блокировки, используемые командой PERFORM LOCKED OPERATION (выполнения заблокированной операции). Если средство CZAM не инсталлировано, чистый сброс задает архитектурному режиму режим ESA/390. Когда средство CZAM инсталлировано, чистый сброс задает архитектурному режиму режим z/Архитектуры. Очистка не затрагивает внешнюю память, такую как запоминающие устройства с прямым доступом, используемые управляющей программой для сохранения содержимого неадресуемых страниц.

Чистый сброс сочетает функцию начального сброса ЦП с функцией инициализации, вызывающей следующие действия:

1. Когда средство CZAM не инсталлировано, архитектурному режиму всех ЦП в конфигурации задается режим ESA/390. Если средство CZAM инсталлировано, архитектурному режиму всех ЦП в конфигурации задается режим z/Архитектуры.

2. Регистрам доступа, общему и с плавающей точкой всех ЦП в конфигурации задаются нулевые значения. Всем 64 битам общих регистров задаются нулевые значения независимо от пребывания ЦП в архитектурном режиме ESA/390 или z/Архитектуры на момент инициирования функции чистого сброса.

3. Содержимому основной памяти в конфигурации и ассоциированных ключей защиты памяти задаются нулевые значения с допустимым кодом проверки блокировок.

4. Используемые каким-либо ЦП в конфигурации блокировки разблокируются при выполнении команды PERFORM LOCKED OPERATION.

5. Выполняется сброс подсистемы.

Проверка правильности входит в состав процессов задания значений регистрам, а также очистки памяти и ключей защиты памяти. Примечания по программированию:

1. Архитектурный режим не изменяется посредством активации кнопки нормального сброса системы или выполнения распоряжения процессора обработки сигналов на сброс ЦП или основной сброс ЦП. Все ЦП в конфигурации находятся в том же самом архитектурном режиме.

2. Для предотвращения воздействия операции сброса ЦП на содержимое полей, которые должны оставаться неизменными, ЦП во время сброса не должен выполнять команды и должен быть деактивирован для всех прерываний. За исключением функционирования таймера ЦП и для возможности появления прерывания машинного контроля, вся активность ЦП может быть остановлена путем перевода ЦП в состоянии ожидания и путем отключения его для I/O и внешних прерываний. Во избежание возможности вызывания сброса в то время, когда обновляется таймер ЦП или

происходит прерывание машинного контроля, ЦП должен пребывать в остановленном состоянии.

3. Сброс ЦП, начальный сброс ЦП, сброс подсистемы и чистый сброс не затрагивают значение и состояние часов TOD.

4. Условия, при которых ЦП входит в состояние контрольного останова, являются модельно-зависимыми и включают в себя сбои, которые исключают завершение текущей операции. Следовательно, если сброс ЦП или начальный сброс ЦП выполняется в то время, когда ЦП находится в состоянии контрольного останова, содержимое PSW, регистров и местоположений памяти, включая сюда ключи защиты памяти и местоположения памяти, к которым получают доступ во время ошибки, может иметь непредсказуемые значения и, в некоторых случаях, содержимое может все еще пребывать в состоянии ошибки после того, как состояние контрольного останова очищено посредством этих сбросов. В этой ситуации, для очистки ошибки требуется чистый сброс.

Сброс по включении

Функция сброса по включении для компонента машины выполняется как часть последовательности включения для этого компонента.

Последовательности включения для часов TOD, основной памяти, дополнительной памяти и канальной подсистемы могут составлять собой часть последовательности включения ЦП, либо последовательность включения для этих модулей может быть инициирована отдельно.

Сброс по включении ЦП: Сброс по включении вызывает выполнение начального сброса ЦП, и может вызывать или может не вызывать выполнение сброса системы I/O в канальной подсистеме. Содержимое общих регистров, регистров доступа и регистров с плавающей точкой очищается к нулевым значениям с допустимым кодом проверки блокировок. Используемые командой PERFORM LOCKED OPERATION и связанные с ЦП блокировки разблокируются, если только они не удерживаются посредством уже включенного ЦП. Если средство CZAM не инсталлировано, а сброс связан с созданием конфигурации, ЦП переводится в режим ESA/390, в противном случае, ЦП переводится в тот же архитектурный режим, что и уже сконфигурированные ЦП. Если средство CZAM инсталлировано, ЦП переводится в режим z/Архитектуры.

Сброс ЦП, начальный сброс ЦП, сброс подсистемы и чистый сброс могут быть инициированы вручную при помощи средств оператора. Начальный сброс ЦП является частью функции начальной загрузки программы. Сброс по включении выполняется как часть процесса включения.

Когда средство CZAM не инсталлировано, если сброс инициируется посредством кнопки чистого сброса системы, нормальной загрузки, или чистой загрузки, либо сбросом по включении ЦП, что образует конфигурацию, архитектурному режиму задается режим ESA/390, в противном случае, архитектурный режим остается неизменным, за исключением того, что сброс по включении задает тот же режим, в котором находятся уже сконфигурированные ЦП. Если средство CZAM инсталлировано, архитектурному режиму задается режим z/Архитектуры.

Другие процессы, операции и/или логики работы, которые могут быть изменены вследствие инсталляции функции конфигурации архитектурного режима, описаны ниже:

(5) Подавляет другие связанные со сбросом действия, предпринимаемые для облегчения перехода между режимами ESA/390 и z/Архитектуры по выполнении сброса. Когда средство CZAM не инсталлировано, текущее PSW сохраняется для последующего использования посредством распоряжения на задание архитектуры от процессора

обработки сигналов, восстанавливающего режим z/Архитектуры. Когда средство CZAM не инсталлировано, текущее PSW изменяется с 16 байтов до восьми байтов. Биты восьмибайтового PSW задаются следующим образом: биты 0-11 и 13-32 задаются равными тем же самым битам 16-байтового PSW, бит 12 задается равным единице, а биты 33-63 задаются равными битам 97-127 16-байтового PSW. Когда средство CZAM инсталлировано, PSW не сохраняется для последующего использования посредством распоряжения на задание архитектуры от процессора обработки сигналов, восстанавливающего режим z/Архитектуры, а текущее PSW не изменяется с 16 байтов до 8 байтов.

(6) Изменяет процесс конфигурирования ЦП с помощью команды конфигурирования ЦП SCLP (служебный вызов логического процессора) и операций кнопки загрузки. Вместо конфигурирования в ESA/390, конфигурирует в заданном сбросом режиме. Команда конфигурирования ЦП SCLP переводит подчиненный ЦП в тот же архитектурный режим, в котором находятся уже сконфигурированные ЦП. По меньшей мере первый ЦП, переведенный в конфигурацию, переводится в нее в связи со сбросом ЦП по включении, и как часть этого сброса, переводится в архитектурный режим, заданный в сбросе ЦП по включении. Альтернативно, модель может задавать режим ЦП, которые находятся в резервном состоянии, когда она задает режим сконфигурированных ЦП.

Включение кнопки чистой загрузки или кнопки нормальной загрузки задает архитектурный режим, как это, соответственно, предписано для чистого сброса или начального сброса ЦП.

(7) Изменяет SIGP таким образом, что он не позволяет распоряжению на задание архитектуры изменять архитектурный режим на ESA/390.

Один предпочтительный вариант осуществления настоящего изобретения команды процессора (SIGP) обработки сигналов описан со ссылками на фиг.8A. В одном предпочтительном варианте осуществления настоящего изобретения команда 800 процессора обработки сигналов имеет совокупность полей, включая сюда, например, поле 802 кода операции (opcode), содержащее код операции, указывающий на операцию процессора обработки сигналов, первое поле (R_1) 804 регистра, второе поле (R3) 806 регистра, основное поле (B2) 808, и поле (D2) 810 смещения. R_1 обозначает общий регистр, содержимое которого является первым операндом, R3 обозначает общий регистр, содержимое которого является третьим операндом, и содержимое регистра, обозначаемого R2, добавляется к смещению в D2 для получения адреса второго операнда.

При функционировании восьмибитовый код распоряжения и, если запрошен, 32-битовый параметр передаются к ЦП, обозначаемому адресом ЦП, содержащимися в третьем операнде. Результат обозначается посредством условного кода и может быть детализирован посредством состояния, собранного в позициях битов 32-63 из местоположения первого операнда.

Адрес второго операнда не используется для обращения к данным, вместо этого, биты 56-63 адреса содержат восьмибитовый код распоряжения. Биты 0-55 адреса второго операнда игнорируются. Код распоряжения задает подлежащую выполнению адресуемым ЦП функцию. Присвоение и описание кодов распоряжения включает в себя, например, последующее, в одном примере:

Код (Десятичное число)	(Шестнадцатеричное число)	Распоряжение
0	00	Неприсвоенный
1	01	Значение
2	02	Внешний вызов
3	03	Аварийный сигнал
4	04	Запуск
5	05	Останов
6	06	Перезапуск
7	07	Неприсвоенный
8	08	Неприсвоенный
9	09	Останов и сохранение состояния
10	0A	Неприсвоенный
11	0B	Начальный сброс ЦП
12	0C	Сброс ЦП
13	0D	Задание префикса
14	0E	Сохранение состояния по адресу
15-17	0F-11	Неприсвоенный
18	12	Задание архитектуры
19	13	Условный аварийный сигнал
14	14	Неприсвоенный
21	15	Текущее состояние значения
22-255	16-FF	Неприсвоенный

Содержащееся в позициях 48-63 битов общего регистра R3 16-битовое двоичное число образует адрес ЦП. Биты 0-47 регистра игнорируются. Когда заданное распоряжение является распоряжением на задание архитектуры, адрес ЦП игнорируется, все другие ЦП в конфигурации считаются адресованными.

Общий регистр, содержащий 32-битовый параметр в позициях 32-63 битов, является тем регистром из числа R_1 или R_1+1, который имеет нечетный номер. От кода распоряжения зависит предоставление параметра и цель его использования.

Непосредственно ранее описанные операнды имеют следующие форматы, в одном примере:

Общий регистр, на который указывает R_1: Биты 0-31 не используются, биты 32-63 включают в себя состояние, Общий регистр, на который указывает R_1 или R_1+1,, который имеет нечетный номер: Биты 0-31 не используются, биты 32-63 включают в себя параметр,

Общий регистр, на который указывает R3: Биты 0-48 не используются, биты 49-63 включают в себя адрес ЦП,

Адрес второго операнда: Биты 0-55 не используются, биты 56-63 включают в себя код распоряжения.

Функция преобразования в последовательную форму выполняется перед началом операции, и вновь после завершения операции.

Когда код распоряжения принят, и никакого ненулевого состояния не возвращается, задается значение условного кода 0. Когда информация о состоянии генерируется данным ЦП (ЦП, выполняющим роль SIGP), или возвращена адресуемым ЦП, состояние размещается в позициях 32-63 битов общего регистра R_1, биты 0-31 регистра остаются неизменными, а условному коду задается значение 1.

Когда путь доступа к адресуемому ЦП занят, или адресуемый ЦП хотя и находится в рабочем состоянии, но его состояние не позволяет ему ответить на код распоряжения, условному коду задается значение 2.

Когда адресуемый ЦП не находится в рабочем состоянии (то есть, он не предоставлен в установке, он не входит в состав конфигурации, он находится в каком-либо из конкретных тестовых наладочных режимов, или он отключен от питания), условному коду задается значение 3.

Получающийся условный код

1 Код распоряжения принят

2 Состояние сохранено

3 Занят

4 Не находится в рабочем состоянии

Программные исключения:

- Привилегированная операция
- Транзакционное ограничение

Когда распоряжение на задание архитектуры процессора обработки сигналов задано в позициях 56-63 битов адреса второго операнда команды процессора обработки сигналов, содержимое позиций 56-63 битов регистра параметра используется в качестве кода, задающего архитектурный режим, который должен быть задан для всех ЦП в конфигурации: код 0 задает режим ESA/390, а коды 1 и 2 задают режим z/Архитектуры. Код 1 задает, что для каждого из всех ЦП в конфигурации текущее PSW для ESA/390 должно быть преобразовано к PSW для z/Архитектуры. Код 2 задает, что PSW для ЦП, выполняющего роль процессора обработки сигналов, должно быть преобразовано к PSW для z/Архитектуры и, что для каждого из всех других ЦП в конфигурации PSW должно быть задано со значением регистра захваченного PSW для z/Архитектуры для этого ЦП. Задание PSW со значением регистра захваченного PSW для z/Архитектуры позволяет восстановить PSW, существовавшее, когда ЦП последний раз находился в режиме z/Архитектуры, при условии, что регистру захваченного PSW для z/Архитектуры не были заданы все нули в результате сброса.

Биты 0-55 регистра параметра игнорируются. Содержимое регистра адреса ЦП процессора обработки сигналов игнорируется, все другие ЦП в конфигурации считаются адресованными.

Когда средство CZAM не инсталлировано, распоряжение принимается только в том случае, если код равен 0, 1 или 2, ЦП уже не находится в режиме, задаваемом кодом, каждый из всех других ЦП находится или в остановленном состоянии или в состоянии контрольного останова, и никакое другое условие не исключает прием распоряжения.

Когда средство CZAM инсталлировано, код 0 не принимается вследствие того, что возврат к режиму ESA/390 не является разрешенным, и поскольку ЦП уже находится в архитектурном режиме z/Архитектуры, специфицирование кодов 1 и 2 имеет

результатом завершение, указывающее на недопустимый параметр, и условный код 1. Другие предварительные условия, обычно проверяемые посредством распоряжения на задание архитектуры, могут как быть, так и не быть проверены.

В случае принятия, распоряжение завершается всеми ЦП во время выполнения процессора обработки сигналов. В этом варианте осуществления различные ЦП ни в коем случае не могут пребывать в различных архитектурных режимах.

Распоряжение на задание архитектуры завершается следующим образом, в одном примере:

- Если код в регистре параметра не равен 0, 1 или 2, или если ЦП уже находится в заданном кодом архитектурном режиме, распоряжение не принимается. Вместо этого биту 55 (недопустимый параметр) общего регистра, обозначаемому полем R] команды процессора обработки сигналов, задается значение единица, и задается условный код 1.

- Если не является истинным, что все другие ЦП в конфигурации находятся в остановленном состоянии или состоянии контрольного останова, распоряжение не принимается. Вместо этого биту 54 (недопустимый параметр) общего регистра, обозначаемому полем R_1 команды процессора обработки сигналов, задается значение единица, и задается условный код 1.

- Архитектурный режим всех ЦП в конфигурации задается, как это указано кодом (например, бит 12 PSW для управления операциями задается для указанного архитектурного режима, и/или другой указатель в вычислительном окружении задается с индикацией указанного архитектурного режима).

- Если распоряжение изменяет архитектурный режим от ESA/390 к z/Архитектуре, и код равняется 1, то для каждого ЦП в конфигурации восьмибайтовое текущее PSW изменяется на 16-байтовое PSW, и биты 16-байтового PSW задаются следующим образом: биты 0-11 и 13-32 задаются равными тем же самым битам восьмибайтового PSW, биту 12 и битам 33-96 задаются нулевые значения, а биты 97-127 задаются равными битам 33-63 восьмибайтового PSW. Кроме того, биту 19 префикса ESA/390, который становится битом 51 префикса z/Архитектуры, задается нулевое значение.

Если код равняется 2, PSW выполняющего роль процессора обработки сигналов ЦП и значения префиксов всех ЦП задаются как в случае кода-1. Для каждого из всех других ЦП в конфигурации PSW задается со значением регистра захваченного PSW для z/Архитектуры. Тем не менее, регистру захваченного PSW для z/Архитектуры задаются все нулевые значения в том случае, если ЦП выполнил сброс, отличный от сброса ЦП, либо во время перемены архитектурного режима, либо впоследствии.

- Если распоряжение изменяет архитектурный режим от z/Архитектуры к ESA/390, то для каждого ЦП в конфигурации: (1) текущее PSW, которое является обновленным PSW в случае ЦП, выполняющего роль процессора обработки сигналов, сохраняется в регистре захваченного PSW для z/Архитектуры, и (2) 16-байтовое текущее PSW изменяется на восьмибайтовое PSW путем задания битов восьмибайтового PSW следующим образом: биты 0-11 и 13-32 задаются равными тем же самым битам 16-байтового PSW, бит 12 задается равным единице, а биты 33-63 задаются равными битам 97-127 16-байтового PSW. Бит 51 префикса z/Архитектуры, становящийся битом 19 префикса ESA/390, остается неизменным.

- ALB (ассоциативные буфера трансляции регистров доступа) и TLB (ассоциативные буфера трансляции) всех ЦП в конфигурации очищаются от их содержимого.

- Функция преобразования в последовательную форму и синхронизации контрольной точки выполняется на всех ЦП в конфигурации.

Если распоряжение изменяет архитектурный режим от z/Архитектуры к ESA/390, и команда процессора обработки сигналов вызывает возникновение выбирающего команду события PER, только самые правые 31 бит адреса команды сохраняются в адресном поле PER для ESA/390.

В одном предпочтительном варианте настоящего изобретения с CZAM последующее является предварительными условиями: Каждый из всех других ЦП находится или в остановленном состоянии или в состоянии контрольного останова, и никакое другое условие не исключает прием распоряжения. Когда средство CZAM инсталлировано, код 0 не принимается вследствие того, что возврат к режиму ESA/390 не является разрешенным, и поскольку ЦП уже находится в архитектурном режиме z/Архитектуры, специфицирование кодов 1 и 2 имеет результатом завершение, указывающее на недопустимый параметр, и условный код 1. Другие предварительные условия, обычно проверяемые посредством распоряжения на задание архитектуры, могут как быть, так и не быть проверены. В еще одном предпочтительном варианте настоящего изобретения SIGP с помощью кода 1 и 2 указывает на успешное завершение без дополнительной индикации.

Один предпочтительный вариант настоящего изобретения обработки, связанной с выполнением команды SIGP для кода распоряжения на задание архитектуры, описан со ссылками на фиг. 8Б. Согласно фиг. 8Б, процессор вычислительного окружения выполняет команду SIGP и получает код распоряжения, указывающий на операцию задания архитектуры, ЭТАП 850. В одном примере, код распоряжения входит в состав адреса второго операнда команды SIGP.

Кроме того, запрашиваемый архитектурный режим, на который должно быть произведено переключение, получают, например, из регистра параметра, заданного командой SIGP, ЭТАП 852. Кроме того, производится выявление относительно того, инсталлировано ли средство конфигурации архитектурного режима, такое как CZAM, ИНФОРМАЦИОННЫЙ ЗАПРОС 854. Это выявляется, в одном примере, посредством индикатора средства.

Если CZAM не инсталлировано, дальнейшее выявление производится относительно того, пребывает ли уже ЦП в требуемом архитектурном режиме, ИНФОРМАЦИОННЫЙ ЗАПРОС 856. В таком случае, состояние предоставляется, например, в регистр, на который указывает команда SIGP, ЭТАП 858, и состояние обрабатывается как ошибка, ЭТАП 860. Однако, если ЦП не находится в требуемом режиме, ИНФОРМАЦИОННЫЙ ЗАПРОС 856, производится выявление относительно того, удовлетворяются ли другие задаваемые командой условия, такие как пребывание других сконфигурированных ЦП вычислительного окружения в остановленном состоянии, и т.д., ИНФОРМАЦИОННЫЙ ЗАПРОС 862. Если условия не удовлетворяются, обработка продолжается к ЭТАПУ 858. В противном случае, распоряжения принимается, ЭТАП 864, и архитектурный режим должен быть изменен. Таким образом, PSW задается, как описано выше, ЭТАП 866, и обработка для этого аспекта команды заканчивается, ЭТАП 868.

Возвращаясь к ИНФОРМАЦИОННОМУ ЗАПРОСУ 854, если CZAM инсталлировано, то производится выявление относительно того, пребывает ли ЦП в требуемом режиме, ИНФОРМАЦИОННЫЙ ЗАПРОС 870. Если ЦП уже находится в требуемом режиме, то в одном примере предоставляется состояние того, что ЦП уже находится в требуемом архитектурном режиме (например, z/Архитектуры), ЭТАП 872. В этом варианте осуществления, однако, это состояние является приемлемым и не обрабатывается как ошибка, ЭТАП 874. Оно может быть либо проигнорировано, либо,

в другом варианте осуществления, может быть предоставлен условный код, который не является кодом ошибки. В еще одном другом варианте осуществления состояние просто указывает на успешное завершение. Существуют также и другие возможности для указания на отсутствие ошибки, несмотря на то, что ЦП уже находится в требуемом архитектурном режиме.

Возвращаясь к ИНФОРМАЦИОННОМУ ЗАПРОСУ 870, если, однако, ЦП не находится в требуемом режиме, то распоряжение не принимается, поскольку является недопустимым возвращение к тому или иному архитектурному режиму (например, ESA/390), ЭТАП 876. Предоставляется состояние, ЭТАП 878, которое считается ошибкой, ЭТАП 880.

В одном предпочтительном варианте осуществления настоящего изобретения, когда CZAM находится в системе в качестве невыбираемого средства, ИНФОРМАЦИОННЫЙ ЗАПРОС 854 может быть исключен, и управление может перейти от ЭТАПА 852 непосредственно к ЭТАПУ 870. В таком варианте осуществления не могут быть реализованы ЭТАПЫ 854-868.

В другом предпочтительном варианте осуществления настоящего изобретения, при получении распоряжения на переключение на текущий режим архитектуры, распоряжение не может быть принято, и ошибка может быть обозначена на ЭТАПЕ 874.

Другие логики работы, процессы и/или операции, которые могут измениться в результате инсталляции CAM, включают в себя:

(8) Изменения битов средства: Новый бит, например, бит 138 добавляется к битам средства для указания на средство конфигурации архитектурного режима для z/Архитектуры, а биту 2, который указывает на активность архитектурного режима z/Архитектуры, должно быть задано значение единица (указание на активность).

По меньшей мере в одном предпочтительном варианте настоящего изобретения, средство CZAM инсталлировано для LPAR и гостя-1 (гости первого уровня являются гостями, инициируемыми посредством гипервизора (например, путем выпуска команды запуска выполнения в режиме интерпретации (SIE)), но не для гостя-2 (гости второго уровня являются гостями, запускаемыми посредством другого гостя (например, путем выпуска команды SIE).

По меньшей мере в одном предпочтительном варианте настоящего изобретения, когда CZAM инсталлировано, и инициируется гость-2 для z/Архитектуры, гость инициируется в режиме z/Архитектуры согласно методике на фиг. 6A. Однако, когда CZAM инсталлировано, и инициируется гость-2 для ESA/390, он инициируется в режиме ESA/390, согласно методике на фиг. 4A, поскольку, в этом варианте осуществления, он не подвергается воздействию CZAM. Таким образом, хост и гости первого уровня управляется посредством CZAM, при этом они инициируются/сбрасываются и т.д. в z/Архитектуре независимо от предпочтения для архитектурного режима (например, они принуждаются к пребыванию в z/Архитектуре, поскольку ESA/390 не поддерживается), но гости ESA/390 второго уровня не подвергаются воздействию CZAM и продолжают инициироваться/сбрасываться и т.д. в ESA/390.

Как описано в настоящем документе, на основании инсталляции средства конфигурации архитектурного режима, такого как средство конфигурации архитектурного режима для z/Архитектуры, изменяются конкретные процессы, операции и/или логики работы сконфигурированного для множественных архитектурных режимов вычислительного окружения. Один такой процесс представлен процессом включения.

Другие аспекты связанной с процессом включения обработки при инсталлированном

средстве конфигурации архитектурного режима описаны со ссылками на фиг. 9.

Согласно фиг. 9, первоначально производится выявление относительно того, инсталлировано ли средство конфигурации архитектурного режима в вычислительном окружении, сконфигурированном для нескольких архитектурных режимов, и имеется ли заданная последовательность включения для включения вычислительного окружения в одном архитектурном режиме (например, устаревшем режиме, таком как ESA/390), ЭТАП 900. Один архитектурный режим включает в себя первую архитектуру системы команд и имеет первый набор поддерживаемых сервисов, таких как 31-битовая адресация, использование 32-битовых регистров общего назначения, а также различные функции. Если выявлено, что средство конфигурации архитектурного режима не инсталлировано, ИНФОРМАЦИОННЫЙ ЗАПРОС 902, тогда выполняется текущая последовательность включения, ЭТАП 904, как описано со ссылками на фиг. 4А-4Б. В противном случае, вычислительное окружение реконфигурируется для ограничения использования одного архитектурного режима (например, устаревшего режима ESA/390), ЭТАП 906. Реконфигурирование включает в себя, например, выбор отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме (например, в более поздней или расширенной версии архитектурного режима, например в z/Архитектуре), ЭТАП 908. Другой архитектурный режим включает в себя вторую архитектуру системы команд и имеет второй набор поддерживаемых сервисов, таких как 64-битовая адресация, использование 64-битовых регистров общего назначения, а также различные функции, такие как динамическая трансляция адресов и/или другие функции. Последовательность включения в этом случае выполняется для включения вычислительного окружения в другом архитектурном режиме, ограничивающем использование одного архитектурного режима, ЭТАП 910, как описано, например, со ссылками на фиг. 6А-6Б. В одном примере, такое выполнение включает в себя загрузку PSW и инвертирование бита 12. После этого, вычислительное окружение выполняется в другом архитектурном режиме (например, z/Архитектуры), ЭТАП 912.

В другом варианте осуществления, как показано на фиг. 10, реконфигурирование включает в себя отключение одной или нескольких поддерживаемых одним архитектурным режимом операций, включая сюда отключение операции переключения, ЭТАП 1000. Например, команда процессора обработки сигналов изменяется для предоставления ошибки на основании запроса на переключение обратно к одному архитектурному режиму, например к ESA/390.

Кроме того, один или несколько других процессов, операций и/или логик работы изменяются для поддержки включения в другом архитектурном режиме вместо одного архитектурного режима, и использование одного архитектурного режима ограничивается, ЭТАП 1002. Один или несколько других процессов, операций и/или логик работы включают в себя, например, выполнение команды ЦП SCLP, которая переводит ЦП в тот же архитектурный режим, в котором находятся уже сконфигурированные ЦП, 1004, кнопка чистой загрузки и кнопка нормальной загрузки, которые являются средствами оператора, которые задают архитектурный режим, как он назначен для чистого сброса или начального сброса ЦП, соответственно, 1006, команда процессора обработки сигналов, изменяющаяся для принятия переключения от архитектурного режима к тому же архитектурному режиму таким образом, что состояние предоставляется и не обрабатывается в качестве ошибки, 1008, и добавление битов средства к индикаторам средства для указания на средство конфигурации архитектурного режима, 1010.

Как описано в настоящем документе, другая операция, затрагиваемая инсталляцией средства конфигурации архитектурного режима, представлена операцией сброса. Один предпочтительный вариант осуществления настоящего изобретения связанной со сбросом обработки описан со ссылками на фиг. 11. Первоначально, процессор получает (например принимает, обеспечивает или иным образом приобретает) операцию сброса, ЭТАП 1100, и операция сброса выполняется для сброса вычислительного окружения к другому архитектурному режиму (например, z/Архитектуры), ЭТАП 1102, как описано в настоящем документе. Этот процесс включает в себя, например, использование PSW, которое находится в соответствующем архитектуре формате, и задание биту 12 в PSW нулевого значения.

В настоящем документе подробно описывается средство конфигурации архитектурного режима, ограничивающее использование конкретных архитектурных аспектов архитектуры, поддерживаемой вычислительным окружением, сконфигурированным для нескольких архитектур. В одном примере, средство конфигурации архитектурного режима инсталлировано, а поддерживающее множественные архитектурные конфигурации вычислительное окружение может быть реконфигурировано таким образом, что аспекты одного из архитектурных режимов (например, устаревшего режима) более не поддерживаются, но другой архитектурный режим (например, расширенный архитектурный режим) остается поддерживаемым. Когда вычислительное окружение сконфигурировано таким образом, предотвращается обратное реконфигурирование вычислительного окружения к неподдерживаемому архитектурному режиму.

В другом варианте осуществления вычислительное окружение динамически конфигурируется в выбранном архитектурном режиме, таком как z/Архитектура. В этом варианте осуществления не может быть осуществлена проверка относительно инсталляции средства CZAM и/или возможности выполнения заданного в явном виде распоряжения SIGP на задание архитектуры. Один предпочтительный вариант осуществления настоящего изобретения логики для выполнения этой конфигурации описан со ссылками на фиг.12.

Согласно фиг. 12, в одном предпочтительном варианте настоящего изобретения, процессор конфигурирует вычислительное окружение для выполнения операций в выбранном архитектурном режиме (например, z/Архитектуре), ЭТАП 1200. Конфигурирование включает в себя, например, начало инициализации вычислительного окружения с использованием сохраненного слова состояния программы, ЭТАП 1202. В одном примере, сохраненное слово состояния программы имеет формат архитектурного режима, отличающегося от выбранного архитектурного режима. Таким образом, производится выявление того, что сохраненное слово состояния программы имеет формат архитектурного режима, отличающегося от выбранного архитектурного режима, ЭТАП 1204. На основании этого выявления, сохраненное слово состояния программы автоматически изменяется для приобретения формата выбранного архитектурного режима, ЭТАП 1206. Автоматическое изменение выполняется в отсутствие заданного в явном виде запроса на переключение к выбранному архитектурному режиму. Инициализация вычислительного окружения с помощью измененного слова состояния программы затем завершается для конфигурирования вычислительного окружения в выбранном архитектурном режиме, ЭТАП 1208.

В одном предпочтительном варианте осуществления настоящего изобретения средство CZAM может использоваться совместно с одним или несколькими другими средствами, включая сюда, например, средство No-DAT и/или средство начальной

загрузки управляющей сервисной программы, описанные в следующих связанных и принадлежащих одному и тому же правообладателю заявках, под названием «Администрирование связанной с выбранными архитектурными средствами обработки» (Managing Processing Associated with Selected Architectural Facilities), Гэйни и др., (Gainey et al)(номер досье IBM: POU 92020 US1), и «Общая последовательность загрузки для управляющей сервисной программы, способной к инициализации в множественных архитектурах» (Common Boot Sequence for Control Utility Able to be Initialized in Multiple Architectures), Майкл К. Гшвинд (Michael K. Gschwind) (номер досье IBM: POU92019US1), соответственно, каждая из которых включена в настоящий документ путем данной отсылки в полном объеме.

Согласно фиг. 13, в одном примере компьютерный программный продукт 1300 включает в себя, например, один или несколько энергонезависимых машиночитаемых информационных носителей 1302 для хранения на них машиночитаемых средств программного кода, логики и/или команд 1304 для предоставления и оказания содействия в использовании одного или нескольких вариантов осуществления.

Настоящее изобретение может быть представлено системой, способом и/или компьютерным программным продуктом. Компьютерный программный продукт может включать в себя машиночитаемый информационный носитель (или носители), имеющий на нем машиночитаемые программные команды для принуждения процессора к выполнению аспектов настоящего изобретения.

Машиночитаемый информационный носитель может быть представлен материальным устройством, которое способно удерживать и сохранять команды для использования посредством устройства выполнения команд. Машиночитаемый информационный носитель может быть представлен, например, в том числе, но не ограничиваясь, устройством электронной памяти, магнитным запоминающим устройством, оптическим запоминающим устройством, электромагнитным запоминающим устройством, полупроводниковым запоминающим устройством или любой подходящей комбинацией из вышеупомянутого. Неисчерпывающий список более конкретных примеров машиночитаемого информационного носителя включает в себя следующее: портативная компьютерная дискета, жесткий диск, оперативная память (RAM), постоянная память (ROM), стираемая программируемая постоянная память (EPROM или флеш-память), статическая оперативная память (SRAM), переносной компакт-диск для однократной записи данных (CD-ROM), цифровой универсальный диск (DVD), карта памяти, гибкий диск, механически закодированное устройство, такое как перфокарты или выступающие структуры в канавке с записанными на них командами, а также любая подходящая комбинация из вышеупомянутого. Машиночитаемый информационный носитель, как он рассматривается в настоящем документе, не подлежит истолкованию в качестве представленного преходящими сигналами как таковыми, такими как радиоволны или другие свободно распространяющиеся электромагнитные волны, электромагнитные волны, распространяющиеся через волновод или другие среды передачи (например, проходящие через волоконно-оптический кабель световые импульсы), или передаваемые через провода электрические сигналы.

Описанные в настоящем документе машиночитаемые программные команды могут быть загружены в соответствующие устройства вычисления/обработки с машиночитаемого информационного носителя или во внешний компьютер или во внешнее устройство хранения через сеть, например Интернет, локальную сеть, глобальную сеть и/или беспроводную сеть. Сеть может содержать медные кабели передачи, волокна оптической передачи, беспроводную передачу, маршрутизаторы,

брандмауэры, переключения, шлюзы и/или граничные серверы. Карта сетевого адаптера или сетевой интерфейс в каждом устройстве вычисления/обработки получает машиночитаемые программные команды из сети и направляет машиночитаемые программные команды для хранения в машиночитаемый информационный носитель в пределах соответствующего устройства вычисления/обработки.

Машиночитаемые программные команды для выполнения операций настоящего изобретения могут быть представлены командами ассемблера, командами архитектуры системы команд (ISA), машинными командами, машинно-зависимыми командами, микрокодом, командами встроенного программного обеспечения, присваивающими значение состоянию данными, или иным исходным кодом или объектным кодом, записанным на любой комбинации из одного или нескольких языков программирования, включая сюда объектно-ориентированные языки программирования, такие как Smalltalk, C++ и т.п., а также обычные языки процедурного программирования, такие как язык программирования «C» или подобные языки программирования. Машиночитаемые программные команды могут выполняться полностью на компьютере пользователя, частично на компьютере пользователя, как автономный пакет программного обеспечения, частично на компьютере пользователя и частично на удаленном компьютере или полностью на удаленном компьютере или сервере. В последнем сценарии удаленный компьютер может быть присоединен к компьютеру пользователя через любой тип сети, включая сюда локальную сеть (LAN) или глобальную сеть (WAN), или присоединение может быть сделано к внешнему компьютеру (например, через Интернет с использованием Интернет-провайдера). В некоторых вариантах осуществления электронные схемы, включающие в себя, например, программируемые логические схемы, программируемые на месте вентильные матрицы (FPGA) или программируемые логические матрицы (PLA) могут выполнять машиночитаемые программные команды посредством использования информации о состоянии машиночитаемых программных команд для настройки электронной схемы с целью выполнения аспектов настоящего изобретения.

Аспекты настоящего изобретения описаны в настоящем документе с отсылками на иллюстрации в виде блок-схем и/или блок-диаграмм для способов, устройств (систем) и компьютерных программных продуктов согласно вариантам осуществления изобретения. Подразумевается, что каждый блок иллюстраций в виде блок-схем и/или блок-диаграмм, а также комбинации блоков на иллюстрациях в виде блок-схем и/или блок-диаграмм, может быть реализован посредством машиночитаемых программных команд.

Такие машиночитаемые программные команды могут быть предоставлены процессору универсального компьютера, специализированного компьютера или другого программируемого устройства обработки данных для образования машины таким образом, что выполняющиеся посредством процессора компьютера или другого программируемого устройства обработки данных команды создают средства для реализации функций/действий, заданных в блоке или блоках блок-схемы и/или блок-диаграммы. Такие машиночитаемые программные команды также могут быть сохранены в машиночитаемом информационном носителе, который может управлять компьютером, программируемым устройством обработки данных и/или другими устройствами для их функционирования особым способом таким образом, что сохраняющий на нем команды машиночитаемый информационный носитель представляет собой изделие, содержащее команды, которые реализуют аспекты функций/действий, заданных в блоке или блоках блок-схемы и/или блок-диаграммы.

Машиночитаемые программные команды могут также быть загружены в компьютер, другое программируемое устройство обработки данных или другое устройство для принуждения к выполнению на компьютере, другом программируемом устройстве или другом устройстве серии эксплуатационных этапов для получения такого компьютерно-реализованного процесса, что выполняемые на компьютере, другом программируемом устройстве или другом устройстве инструкции реализуют функции/действия, заданные в блоке или блоках блок-схемы и/или блок-диаграммы.

Блок-схемы и блок-диаграммы на чертежах показывают архитектуру, функциональность и функционирование возможных реализаций систем, способов и компьютерных программных продуктов согласно различным вариантам осуществления настоящего изобретения. В этом отношении каждый блок в блок-схемах или блок-диаграммах может представлять модуль, сегмент или участок команд, который содержит одну или несколько исполнимых команд для реализации указанной логической функции (функций). В некоторых альтернативных реализациях отмеченные в блоке функции могут осуществляться в порядке, отличном от указанного на чертежах. Например, два блока, показанные следующими друг за другом, фактически могут быть выполнены по существу одновременно, или блоки могут иногда выполняться в обратном порядке, в зависимости от предусмотренной к выполнению функциональности. Необходимо также отметить, что каждый блок на иллюстрациях в виде блок-схем и/или блок-диаграмм, а также в комбинациях блоков на иллюстрациях в виде блок-схем и/или блок-диаграмм, может быть реализован посредством основанных на аппаратных средствах систем особого назначения, которые выполняют указанные функции или действия или выполняют комбинации аппаратных и компьютерных команд особого назначения.

В дополнение к вышеупомянутому, один или несколько аспектов могут быть предоставлены, предложены, развернуты, управляемы, обслуживаемы и т.д. поставщиком услуг, предлагающим управление окружениями заказчика. Например, поставщик услуг может создавать, обслуживать, поддерживать и т.д. машинный код и/или компьютерную инфраструктуру, выполняющую один или несколько аспектов для одного или нескольких клиентов. В свою очередь, поставщик услуг может получать оплату от клиента в рамках соглашения о подписке и/или о сборах, в качестве примеров. Дополнительно или альтернативно, поставщик услуг может получать оплату от продажи рекламного содержимого одному или нескольким третьим лицам.

В одном аспекте приложение может быть развернуто для выполнения одного или нескольких предпочтительных вариантов осуществления. В качестве примера, развертывание приложения предусматривает обеспечение компьютерной инфраструктуры, действующей для выполнения одного или нескольких вариантов осуществления.

В качестве другого аспекта, может быть развернута вычислительная инфраструктура, содержащая в вычислительной системе интегрирующий машиночитаемый код, в которой код, в сочетании с вычислительной системой, способен к выполнению одного или нескольких вариантов осуществления.

В качестве еще одного аспекта, может быть предоставлен способ интеграции содержащей интегрирующий машиночитаемый код вычислительной инфраструктуры в компьютерную систему.

Компьютерная система содержит машиночитаемый носитель, причем компьютерный носитель содержит один или несколько вариантов осуществления. Код в сочетании с компьютерной системой способен к выполнению одного или нескольких вариантов осуществления.

Хотя различные варианты осуществления описаны выше, они являются только примерами. Например, вычислительные окружения другой архитектуры могут быть использованы для охвата и использования одного или нескольких вариантов осуществления. Кроме того, могут быть использованы различные команды, форматы команд, поля команд и/или значения команд. Также и другие, неупомянутые типы процессов, операций и/или логик работы могут быть затронуты в результате инсталляции CAM. Являются возможными самые разнообразные вариации.

Кроме того, могут быть использованы и оказаться выгодными другие типы вычислительных окружений. В качестве примера, является применимой подходящая для сохранения и/или выполнения программного кода система обработки данных, которая включает в себя по меньшей мере два процессора, соединенных прямо или косвенно через системную шину с элементами памяти. Элементы памяти включают в себя, например, локальную память, используемую в процессе фактического выполнения программного кода, запоминающее устройство большого объема и кэш-память, которая предоставляет временное сохранение по меньшей мере части программного кода с целью уменьшения числа выборок из запоминающего устройства большого объема в процессе выполнения.

Устройства ввода-вывода или устройства I/O (в том числе, но не ограничиваясь, клавиатуры, дисплеи, позиционирующие устройства, DASD (запоминающее устройство прямого доступа), устройство записи на ленту, CD, DVD, карты флеш-памяти и другие носители памяти и т.д.) могут быть соединены с системой или непосредственно или через переходные контроллеры I/O. С системой также могут быть соединены сетевые адаптеры для предоставления системе обработки данных возможности установления соединения с другими системами обработки данных или с удаленными принтерами или с запоминающими устройствами посредством переходных частных сетей или сетей общего пользования. Модемы, кабельные модемы и платы Ethernet являются всего несколькими примерами из числа доступных типов сетевых адаптеров.

Согласно фиг. 14, изображены характерные компоненты системы 5000 хост-компьютера для реализации одного или нескольких вариантов осуществления. Характерная система 5000 хост-компьютера содержит один или несколько ЦП 5001, находящихся в связи с памятью 5002 компьютера (то есть, центральной памятью), а также интерфейсы I/O к устройствам 5011 хранения данных и сетям 5010 для сообщения с другими компьютерами или SAN (системными сетями) и т.п. ЦП 5001 является совместимым с архитектурой, имеющей архитектурно спроектированную систему команд и архитектурно спроектированную функциональность. ЦП 5001 может иметь трансляцию (ART) 5012 регистров доступа, что включает в себя ассоциативный буфер (ALB) 5013 ART для выбора адресного пространства, которое используются динамической трансляцией (DAT) 5003 адресов для преобразования адресов программы (виртуальных адресов) в действительные адреса памяти. DAT обычно включает в себя ассоциативный буфер (TLB) 5007 трансляции для кэширования трансляций таким образом, что более поздние доступы к блоку памяти 5002 компьютера не вызывают задержек при трансляции адресов. Как правило, кэш 5009 используется между памятью 5002 компьютера и процессором 5001. Кэш 5009 может быть построен иерархическим образом, то есть, иметь большой кэш, доступный для более чем одного ЦП, и меньшие, более быстрые (нижнего уровня) кэши, размещенные между большим кэшем и каждым из ЦП. В некоторых реализациях кэши нижнего уровня разделены для предоставления отдельных низкоуровневых кэшей для выборки команд и доступов к данным.

В одном предпочтительном варианте осуществления настоящего изобретения команда

выбирается из памяти 5002 посредством устройства 5004 выборки команд через кэш 5009. Команда декодируется в устройстве 5006 декодирования команд и диспетчеризируется (совместно с другими командами в некоторых вариантах осуществления) к устройству или устройствам 5008 выполнения команд. Обычно используют несколько устройств 5008 выполнения, например арифметическое устройство выполнения, устройство выполнения для операций с плавающей точкой и устройство выполнения команд ветвления. Команда выполняется посредством устройства выполнения, которое по мере необходимости получает доступ к операндам из заданных командой регистров или к памяти. Если нужно получить доступ к операнду (загруженному или сохраненному) из памяти 5002, устройство 5005 загрузки и хранения, как правило, осуществляет доступ под управлением выполняемой команды. Команды могут быть выполнены в аппаратных схемах или во внутреннем микрокоде (встроенном программном обеспечении) или в комбинации обоих способов.

Как отмечено, компьютерная система включает в себя информацию в локальной (или главной) памяти, а также адресацию, защиту, а также запись изменений и обращений. Некоторые аспекты адресации включают в себя формат адресов, концепцию адресных пространств, различные типы адресов, а также способ, посредством которого один тип адреса транслируется в другой тип адреса. Часть основной памяти включает в себя постоянно назначенные местоположения памяти. Основная память предоставляет системе прямо адресуемое запоминающее устройство с быстрой выборкой для данных. Как данные, так и программы должны быть загружены в основную память (от устройств ввода данных) прежде, чем они могут быть обработаны.

Основная память может включать в себя один или несколько меньших, буферных накопителей с более быстрым доступом, иногда называемых кэшами. Кэш обычно физически соединен с ЦП или с процессором I/O. Обусловленные физической конструкцией и использованием различных информационных носителей эффекты функционирования, за исключением производительности, в большинстве случаев не могут быть замечены программой.

Для операндов команд и данных могут содержаться отдельные кэши. Информация в пределах кэша сохраняется в непрерывных байтах на целочисленной границе, называемой блоком кэша или строкой кэша (или строкой, если коротко). Модель может предоставить команду EXTRACT CACHE ATTRIBUTE (извлечь реквизит кэша), возвращающую размер строки кэша в байтах. Модель может также предоставить команды PREFETCH DATA (предварительной выборки данных) и PREFETCH DATA RELATIVE LONG (предварительной выборки данных относительно большого объема), которые производят упреждающую выборку памяти в кэше данных или команд или разблокирование данных из кэша.

Память рассматривается как длинная горизонтальная строка битов. Для большинства операций доступы к памяти продолжаются в последовательности слева направо. Строка битов подразделена на модули длиной восемь битов. Восьмибитовый модуль называют байтом, который является основой всех информационных форматов. Каждое местоположение байта в памяти идентифицируется уникальным неотрицательным целым числом, которое является адресом этого местоположения байта или, просто, адресом байта. Смежные местоположения байта имеют последовательные адреса, начинающиеся с 0 слева и продолжающиеся в последовательности слева направо. Адреса являются двоичными целыми числами без знака и равняются 24, 31 или 64 битам.

Информация передается между памятью и ЦП или канальной подсистемой по одному байту или по группе байтов за один раз. Если иного не указано, например, в z/

Архитектуре, группа байтов в памяти адресуется по крайнему левому байту группы.

Число байтов в группе либо подразумевается, либо явно задается подлежащей выполнению операцией. При использовании в операции ЦП, группу байтов называют полем. В пределах каждой группы байтов, например, в z/Архитектуре, биты пронумерованы в последовательности слева направо. В z/Архитектуре крайние левые биты иногда упоминаются как биты «старшего разряда», а самые правые биты - как биты «младшего разряда». Тем не менее, номера битов не являются адресами памяти. Только байты могут адресоваться. Для действия с отдельными битами байта в памяти доступ получают ко всему байту. Биты в байте пронумерованы от 0 до 7, слева направо (например, в z/Архитектуре). Биты в адресе могут быть пронумерованы 8-31 или 40-63 для 24-битовых адресов, или 1-31 или 33-63 для 31-битовых адресов, они пронумерованы 0-63 для 64-битовых адресов. В одном примере, биты 8-31 и 1-31 относятся к адресам, которые находятся в местоположении (например, регистре), который имеет ширину в 32 бита, тогда как биты 40-63 и 33-63 относятся к адресам, которые находятся в местоположении шириной 64 бита. В пределах любого другого формата фиксированной длины для множественных байтов, составляющие формат биты пронумерованы последовательно, начиная от 0. В целях обнаружения ошибок и, предпочтительно, для их исправления, с каждым байтом или с группой байтов могут быть переданы один или несколько контрольных битов. Такие контрольные биты генерируются автоматически посредством машины, и не могут непосредственно управляться посредством программы. Емкости памяти выражаются в числе байтов. Когда длина поля операнда хранения неявно задается посредством операционного кода команды, считается, что поле имеет фиксированную длину, которая может составлять один, два, четыре, восемь или шестнадцать байтов. Для некоторых команд могут неявно задаваться более крупные поля. Когда длина поля операнда хранения задается не неявно, но заявляется явным образом, считается, что поле имеет переменную длину. Операнды переменной длины могут изменяться по длине посредством приращений в один байт (или с некоторыми командами, с кратностью в два байта или с другой кратностью). Когда информация размещается в памяти, заменяется содержимое только тех местоположений байтов, которые входят в состав указанного поля, также и в том случае, когда ширина физического тракта к памяти может быть больше, чем длина сохраняемого поля.

Некоторые единицы информации должны располагаться в памяти на целочисленной границе. Границу называют целочисленной для единицы информации, когда ее адрес в памяти является кратным длине модуля в байтах. Полям величиной в 2, 4, 8, 16 и 32 байта на целочисленной границе дают специальные наименования. Полуслово является группой из двух последовательных байтов на двухбайтовой границе и является основным конструкционным блоком команд. Слово является группой из четырех последовательных байтов на четырехбайтовой границе. Двойное слово является группой из восьми последовательных байтов на восьмибайтовой границе. Учетверенное слово является группой из 16 последовательных байтов на 16-байтовой границе. Восьмикратное слово является группой из 32 последовательных байтов на 32-байтовой границе. Когда адреса в памяти обозначают полуслова, слова, двойные слова, учетверенные слова и восьмикратные слова, двоичное представление адреса содержит, соответственно, один, два, три, четыре, или пять самых правых нулевых битов. Команды должны располагаться на двухбайтовых целочисленных границах. Операнды хранения большинства команд не имеют требований по выравниванию границ.

На устройствах, реализующих отдельные кэши для операндов команд и данных,

может возникать значительная задержка, если программа сохраняет в строку кэша, из которой команды впоследствии выбираются, независимо от того, изменяет ли сохранение подлежащие последующей выборке команды.

В одном примере, вариант осуществления может быть реализован посредством программного обеспечения (иногда называемого лицензированным внутренним кодом, встроенным программным обеспечением, микрокодом, милликодом, пикокодом и т.п., каждый из которых должен быть совместимым с одним или несколькими вариантами осуществления). Согласно фиг. 14, к программному коду программного обеспечения, воплощающему один или несколько аспектов, доступ может быть получен процессором 5001 хост-системы 5000 в долговременных запоминающих устройствах 5011, таких как дисковод для компакт-дисков, устройство записи на ленту или жесткий диск. Программный код программного обеспечения может быть воплощен на любом из ряда известных носителей для использования с системой обработки данных, таких как дискета, жесткий диск или компакт-диск. Код может распространяться на упомянутых носителях или может распространяться к пользователям из памяти 5002 компьютера или памяти одной компьютерной системы по сети 5010 к другим компьютерным системам для использования пользователями таких других систем.

Программный код программного обеспечения включает в себя операционную систему, которая управляет функционированием и взаимодействием различных компьютерных компонентов, а также одну или несколько прикладных программ. Программный код обычно поставляется с разбиением на страницы от устройства 5011 хранения данных к относительно более высокоскоростной компьютерной памяти 5002, где он является доступным для обработки посредством процессора 5001. Методы и способы воплощения программного кода программного обеспечения в памяти, на физических носителях и/или распространения программного кода через сети являются хорошо известными и далее не обсуждаются в настоящем документе. Программный код, когда он создан и сохранен на материальном носителе (в том числе, но не ограничиваясь, модулях электронной памяти (RAM), флеш-памяти, компакт-дисках (CD), DVD, устройствах записи на магнитная ленту и т.п.), зачастую называют «компьютерным программным продуктом». Носитель компьютерного программного продукта обычно является считываемым посредством устройства обработки данных, предпочтительно, в составе компьютерной системы, для выполнения посредством устройства обработки данных.

Фиг. 15 показывает характерные аппаратные средства системы рабочей станции или сервера, в которых могут быть осуществлены один или несколько вариантов осуществления. Система 5020 на фиг. 15 содержит характерную основную компьютерную систему 5021, такую как персональный компьютер, рабочая станция или сервер, включающую в себя опциональные периферийные устройства. Основная компьютерная система 5021 включает в себя один или несколько процессоров 5026 и шину, используемую для соединения и обеспечения коммуникации между процессором (процессорами) 5026 и другими компонентами системы 5021 согласно известным методам. Шина соединяет процессор 5026 с памятью 5025 и с долговременным запоминающим устройством 5027, которое может включать в себя жесткий диск (включающий в себя, например, любой из магнитных носителей, CD, DVD и флеш-память) или, например, устройство записи на ленту. Система 5021 может также включать в себя адаптер пользовательского интерфейса, соединяющий микропроцессор 5026 через шину с одним или несколькими устройствами интерфейса, такими как клавиатура 5024, мышь 5023, принтер/сканер 5030 и/или с другими устройствами интерфейса,

которые могут быть представлены любыми устройствами пользовательского интерфейса, такими как сенсорный экран, цифровая контактная панель и т.д. Шина также соединяет через дисплейный адаптер дисплей 5022, такой как жидкокристаллический экран или монитор с микропроцессором 5026.

Система 5021 может сообщаться с другими компьютерами или компьютерными сетями посредством сетевого адаптера, способного к сообщению 5028 с сетью 5029. Типовые сетевые адаптеры представлены коммуникационными каналами, кольцом с маркерным доступом, сетью Ethernet или модемами. Альтернативно, система 5021 способна к сообщению с использованием беспроводного интерфейса, такого как карта CDPD (сотовые цифровые пакетные данные). Система 5021 может быть связана с другими подобными компьютерами по локальной сети (LAN) или глобальной сети (WAN), или система 5021 может быть клиентом в клиент-серверной структуре с другим компьютером и т.д. Все эти конфигурации, а также соответствующее коммуникационное аппаратное и программное обеспечение являются известными из уровня техники.

Фиг. 16 показывает сеть 5040 обработки данных, в которой могут быть осуществлены один или несколько вариантов осуществления. Сеть обработки данных 5040 может включать в себя несколько отдельных сетей, таких как беспроводная сеть и проводная сеть, каждая из которых может включать в себя несколько отдельных рабочих станций 5041, 5042, 5043, 5044. Кроме того, как осведомлены специалисты в данной области техники, в состав такой сети могут входить одна или несколько LAN, причем LAN может содержать несколько интеллектуальных рабочих станций, соединенных с хост-процессором.

Продолжая рассмотрение фиг. 16, сети могут также включать в себя мэйнфреймовые компьютеры или серверы, такие как шлюзовой компьютер (клиент-сервер 5046) или сервер приложений (удаленный сервер 5048, который может получить доступ к репозиторию данных, и к которому доступ также может быть получен непосредственно с рабочей станции 5045). Шлюзовой компьютер 5046 служит точкой входа в каждую отдельную сеть. Шлюз является необходимым при присоединении одного сетевого протокола к другому. Шлюзовой компьютер 5046 может быть, предпочтительно, соединен с другой сетью (например, с Интернетом 5047) посредством линии связи. Шлюзовой компьютер 5046 может также быть непосредственно соединен с одной или несколькими рабочими станциями 5041, 5042, 5043, 5044 с использованием линии связи. Шлюзовой компьютер может быть реализован с использованием сервера IBM eServer System z, поставляемого International Business Machines Corporation.

При одновременном рассмотрении фиг. 15 и фиг. 16, к программному коду 5031 программного обеспечения, который может воплощать один или несколько аспектов, процессор 5026 системы 5020 может получить доступ в долговременных запоминающих устройствах 5027, таких как дисковод для компакт-дисков или жесткий диск.

Программный код программного обеспечения может быть воплощен на любом из ряда известных носителей для использования с системой обработки данных, таких как дискета, жесткий диск или компакт-диск. Код может распространяться на упомянутых носителях или может распространяться к пользователям 5050, 5051 из памяти или устройства хранения одной компьютерной системы по сети к другим компьютерным системам для использования пользователями таких других систем.

Альтернативно, программный код может быть воплощен в памяти 5025, и к нему может быть получен доступ процессором 5026 с помощью шины процессора. Такой программный код включает в себя операционную систему, которая управляет функционированием и взаимодействием различных компьютерных компонентов, а

также одну или несколько прикладных программ 5032. Программный код обычно поставляется с разбиением на страницы от устройства 5027 хранения данных к относительно более высокоскоростной компьютерной памяти 5025, где он является доступным для обработки посредством процессора 5026. Методы и способы воплощения программного кода программного обеспечения в памяти, на физических носителях и/или распространения программного кода через сети являются хорошо известными и далее не обсуждаются в настоящем документе. Программный код, когда он создан и сохранен на материальном носителе (в том числе, но не ограничиваясь, модулях электронной памяти (RAM), флеш-памяти, компакт-дисках (CD), DVD, устройствах записи на магнитная ленту и т.п.), зачастую называют «компьютерным программным продуктом». Носитель компьютерного программного продукта обычно является считываемым посредством устройства обработки данных, предпочтительно, в составе компьютерной системы, для выполнения посредством устройства обработки данных.

Кэш, который является самым легкодоступным для процессора (обычно является более быстрым и малым, чем другие кэши процессора), является самый низким по уровню (L1 или первого уровня) кэшем, а основное запоминающее устройство (оперативная память) является кэшем высшего уровня (L3, если имеется 3 уровня). Кэш самого низкого уровня зачастую делится на кэш команд (I-кэш), содержащий подлежащие выполнению машинные команды, и кэш данных (D-кэш), содержащий операнды данных.

Согласно фиг. 17, типовой вариант осуществления процессора изображен для процессора 5026. Обычно один или несколько уровней кэша 5053 используются для буферизации блоков данных в памяти с целью повышения производительности процессора. Кэш 5053 является высокоскоростным буфером, содержащим строки кэша данных памяти, использование которых является вероятным. Типичные строки кэша вмещают 64, 128 или 256 байтов данных памяти. Отдельные кэши чаще используются для кэширования команд, а не для кэширования данных. Согласованность кэша (синхронизация копий строк в памяти и кэшах) зачастую предоставляется посредством различных отслеживающих алгоритмов, известных из уровня техники. Основную память 5025 процессорной системы зачастую называют кэшем. В процессорной системе, имеющей 4 уровня кэша 5053, основную память 5025 иногда называют кэшем уровня 5 (L5), поскольку она обычно является более быстрой и удерживает только часть энергонезависимой памяти (DASD, устройство записи на ленту и т.д.), которая является доступной для компьютерной системы. Основная память 5025 «кэширует» страницы данных, подкачиваемые постранично в основную память 5025 и из нее посредством операционной системы.

Счетчик программы (счетчик команд) 5061 отслеживает адрес текущей подлежащей выполнению команды. Счетчик программы в процессоре z/Архитектуры составляет 64 бита и может быть усечен до 31 или 24 битов для поддержки прежних пределов адресации. Счетчик программы обычно воплощается в PSW (слово состояния программы) компьютера таким образом, что он сохраняется в процессе переключения контекста. Таким образом, выполняющаяся программа, имеющая значение счетчика программы, может быть прервана, например, посредством операционной системы (переключение контекста от окружения программы к среде операционной системы). PSW программы поддерживает значение счетчика программы во время неактивности программы, и счетчик программы (в PSW) операционной системы используется во время выполнения операционной системы. Как правило, счетчик программы увеличивает отсчет на величину, равную числу байтов текущей команды. Команды RISC (компьютера

с сокращенной системой команд) обычно имеют фиксированную длину, в то время как команды CISC (компьютера со сложной системой команд) обычно имеют переменную длину. Команды z/Архитектуры IBM являются командами CISC, имеющими длину 2, 4 или 6 байтов. Счетчик 5061 программы изменяется посредством либо операции по переключению контекста, либо, например, операции принятия ветвления команды ветвления. В операции по переключению контекста текущее значение счетчика программы сохраняется в слове состояния программы наряду с другой информацией о состоянии относительно выполняемой программы (такой как условные коды), и загружается новое значение счетчика программы, указывающее на подлежащую выполнению команду нового программного модуля. Операция принятия ветвления выполняется с целью разрешения программе принятия решений или реализации в рамках программы оператора цикла путем загрузки результата команды ветвления в счетчик 5061 программы.

Обычно устройство 5055 выборки команд используется для выборки команд в интересах процессора 5026. Устройство выборки либо выбирает «следующие последовательные команды», целевые команды команд принятия ветвления или первые команды программы после переключения контекста. Современные устройства выборки команд зачастую используют методы упреждающей выборки для предположительной упреждающей выборки команд на основании вероятности использования выбранных с упреждением команд. Например, устройство выборки может выбрать 16 байтов команды, включающей в себя следующую последовательную команду и дополнительные байты других последовательных команд.

Выбранные команды затем выполняются посредством процессора 5026. В варианте осуществления выбранная команда (команды) передаются к устройству 5056 диспетчера устройства выборки. Устройство диспетчера декодирует команду (команды) и передает информацию относительно декодируемой команды (команд) в соответствующие устройства 5057, 5058, 5060. Устройство 5057 выполнения обычно получает информацию относительно декодированных арифметических команд от устройства 5055 выборки команд, и выполняет арифметические операции на операндах согласно коду операции команды. Операнды предоставляются устройству 5057 выполнения, предпочтительно, либо из памяти 5025, архитектурно спроектированных регистров 5059, либо из поля непосредственной адресации выполняемой команды. Результаты выполнения, при сохранении, сохраняются либо в памяти 5025, регистрах 5059, либо в других аппаратных средствах машины (таких как регистры управления, регистры PSW и т.п.).

Виртуальные адреса преобразовываются в действительные адреса с помощью динамической трансляции 5062 адресов и, опционально, с помощью трансляции 5063 регистров доступа.

Процессор 5026 обычно имеет одно или несколько устройств 5057, 5058, 5060 для выполнения функции команды. Согласно фиг. 18A, устройство 5057 выполнения может сообщаться 5071 с архитектурно спроектированными общими регистрами 5059, с устройством 5056 декодирования/диспетчеризации, с устройством 5060 загрузки и хранения, а также с другими процессорами 5065 путем взаимодействия посредством интерфейсной логики 5071. Устройство 5057 выполнения может использовать несколько контуров 5067, 5068, 5069 регистров для содержания информации, которой оперирует арифметико-логическое устройство (ALU) 5066. ALU выполняет арифметические операции, такие как сложение, вычитание, умножение и деление, а также логические функции, такие как И, Или, а также Исключающего Или (XOR), Циклического Сдвига и Смещения. Предпочтительно, ALU поддерживает специализированные операции,

которые являются зависимыми от проектного решения. Другие контуры могут предоставлять другие архитектурно спроектированные функции 5072, включающие в себя, например, условные коды и логику поддержки восстановления. Обычно результат операции ALU содержится в контуре 5070 выходного регистра, который может передавать результат к ряду других функций обработки. Имеются самые разнообразные структуры процессорных устройств, настоящее описание предназначается только для обеспечения понимания одного характерного предпочтительного варианта осуществления настоящего изобретения.

Команда ADD, например, может выполняться в устройстве 5057 выполнения, имеющем арифметическую и логическую функциональности, в то время как команда с плавающей точкой, например, может выполняться в устройстве выполнения с плавающей точкой, имеющем специализированный инструмент работы с плавающей точкой. Предпочтительно, устройство выполнения воздействует на операнды, идентифицируемые командой путем выполнения на операндах функции, задаваемой посредством кода операции. Например, команда ADD может быть выполнена посредством устройства 5057 выполнения на операндах, найденных в двух регистрах 5059, идентифицируемых полями регистра в команде.

Устройство 5057 выполнения выполняет арифметическое сложение на двух операндах и сохраняет результат в третьем операнде, причем третий операнд может быть представлен третьим регистром или одним из двух исходных регистров. Устройство выполнения предпочтительно использует арифметико-логическое устройство (ALU) 5066, которое является способным к выполнению ряда логических функций, таких как Смещение, Циклический сдвиг, И, Или и Исключающее или, а также ряда алгебраических функций, включающих в себя любую операцию из числа сложения, вычитания, умножения, деления. Некоторые ALU 5066 разработаны для скалярных операций, а некоторые - для операций с плавающей точкой. Данные могут быть представлены, в зависимости от архитектуры, в порядке следования, начиная со старшего (где наименее значащий байт располагается по адресу старшего байта), или в порядке следования, начиная с младшего (где наименее значащий байт располагается по адресу младшего байта), zl Архитектура IBM имеет порядок следования, начиная со старшего. Поля значения со знаком, в зависимости от архитектуры, могут быть представлены знаком и порядком величины, представленным в обратном двоичном коде числом или представленным в дополнительном двоичном коде числом. Представленное в дополнительном двоичном коде число является выгодным в том плане, что для ALU не требуется проектировать инструмент для вычитания, поскольку как отрицательная величина, так и положительная величина в дополнительном двоичном коде требуют только сложения в пределах ALU. Числа обычно описываются в сокращенном виде, когда 12 битовое поле задает адрес 4 096-байтового блока и обычно описывается, например, как 4 кбайтовый (килобайтовый) блок.

Согласно фиг. 18Б, информация о команде ветвления для выполнения команды ветвления обычно отправляется в устройство 5058 обработки ветвлений, которое зачастую использует алгоритм предсказания ветвлений, такой как таблица 5082 истории ветвлений (BHT) для предсказания результата ветвления прежде завершения других условных операций. Цель текущей команды ветвления выбирается и упреждающим образом выполняется прежде завершения условных операций. Когда условные операции завершаются, выполняемые упреждающим образом команды ветвления или завершаются или сбрасываются на основании условий условной операции и упреждающего результата. Типичная команда ветвления может проверять условные

коды и выполнять ветвление к целевому адресу, если условные коды отвечают требованию ветвления команды ветвления, целевой адрес может быть вычислен на основании нескольких чисел, включая сюда, например, найденные в полях регистра или в поле непосредственной адресации команды. Устройство 5058 обработки ветвлений может использовать ALU 5074, имеющий несколько входных контуров 5075, 5076, 5077 регистров и контур 5080 выходного регистра. Устройство 5058 обработки ветвлений может сообщаться 5081 с общими регистрами 5059, с устройством 5056 декодирования/диспетчеризации или, например, с другими контурами 5073.

Выполнение группы команд может быть прервано по ряду причин, включающему в себя переключение контекста, инициируемое операционной системой, исключением программы или ошибкой, вызывающей переключение контекста, сигналом прерывания I/O, вызывающим переключение контекста или, например, многопоточной активностью нескольких программ (в многопоточном окружении). Предпочтительно, действие переключения контекста сохраняет информацию о состоянии относительно выполняющейся в настоящее время программы, а затем загружает информацию о состоянии относительно другой вызываемой программы. Информация о состоянии может быть сохранена, например, в аппаратных регистрах или в памяти. Информация о состоянии предпочтительно содержит значение счетчика программы, указывающее на следующую подлежащую выполнению команду, условные коды, информацию о трансляции памяти и содержимое архитектурно спроектированного регистра. Действие переключения контекста может быть осуществлено аппаратными схемами, прикладными программами, программами операционной системы или микропрограммным кодом (микрокодом, пикокодом или лицензированным внутренним кодом (LIC)), поодиночке или в комбинации.

Процессор получает доступ к операндам согласно заданным командой методам. Команда может предоставить операнд непосредственной адресации с помощью значения участка команды, может предоставить одно или несколько полей регистра, явно указывающих либо на регистры общего назначения, либо на регистры особого назначения (например, на регистры с плавающей точкой). Команда может использовать в качестве операндов неявно задаваемые регистры, идентифицируемые посредством поля кода операции. Команда может использовать для операндов местоположения памяти. Местоположение памяти операнда может быть предоставлено регистром, полем непосредственной адресации или комбинацией регистров и полей непосредственной адресации, образцом чего является средство длинного смещения z/Архитектуры, где команда задает базовый регистр, индексный регистр и поле непосредственной адресации (поле смещения), которые складываются вместе для предоставления, например, адреса операнда в памяти. Местоположение в настоящем документе обычно подразумевает местоположение в оперативной памяти (основной памяти), если не указано иное.

Согласно фиг. 18B, процессор получает доступ к памяти с помощью устройства 5060 загрузки и хранения. Устройство 5060 загрузки и хранения может выполнять операцию загрузки путем получения адреса целевого операнда в памяти 5053 и загрузки операнда в регистре 5059 или в другом местоположении памяти 5053, или может выполнять операцию сохранения путем получения адреса целевого операнда в памяти 5053 и сохранения данных, полученных из регистра 5059 или другого местоположения памяти 5053, в целевом местоположении операнда в памяти 5053. Устройство 5060 загрузки и хранения может быть упреждающим, и может получать доступ к памяти в последовательности, порядок которой не соответствует таковому для

последовательности команды, однако устройство 5060 загрузки и хранения должно поддерживать внешнее представление по отношению к программам, команды которых выполняются по порядку. Устройство 5060 загрузки и хранения может сообщаться 5084 с общими регистрами 5059, с устройством 5056 декодирования/диспетчеризации, с интерфейсом 5053 кэша/памяти или с другими элементами 5083, и содержит различные контуры 5086, 5087, 5088 и 5089 регистра, ALU 5085 и управляющую логику (CLT) 5090 для вычисления адресов памяти и для предоставления конвейерного упорядочивания для поддержания операций упорядоченными. Некоторые операции могут располагаться не по порядку, но устройство загрузки и хранения предоставляет функциональность для предложения находящихся вне порядка операций программе в виде выполненных по порядку, как это хорошо известно из уровня техники.

Предпочтительно, «видимые» прикладной программой адреса, зачастую называют виртуальными адресами. Виртуальные адреса иногда называют «логическими адресами» и «исполнительными адресами». Эти виртуальные адреса являются виртуальными в том смысле, что они перенаправляются к местоположению физической памяти посредством одной из ряда технологий динамической трансляции адресов (DAT), в том числе, но не ограничиваясь, путем простого снабжения виртуального адреса префиксом в виде значения смещения, транслирующим виртуальный адрес с помощью одной или нескольких трансляционных таблиц, причем трансляционные таблицы предпочтительно содержат по меньшей мере таблицу сегментов и таблицу страниц, по одиночке или в комбинации, и причем таблица сегментов предпочтительно имеет запись, указывающую на таблицу страниц. В z/Архитектуре предоставляется иерархия трансляции, включающая в себя таблицу первой области, таблицу второй области, таблицу третьей области, таблицу сегментов и опциональную таблицу страниц. Производительность трансляции адресов зачастую улучшают путем использования ассоциативного буфера (TLB) трансляции, содержащего записи, отображающие виртуальный адрес на ассоциированное местоположение физической памяти. Записи создаются, когда DAT транслирует виртуальный адрес с помощью трансляционных таблиц. В этом случае, при последующем использовании виртуального адреса может быть применена запись быстрого TLB, а не медленных последовательных доступов трансляционной таблицы. Содержимым TLB можно управлять посредством ряда алгоритмов замены, включающих в себя LRU (наиболее давно не используемый).

В случае, когда процессор является процессором многопроцессорной системы, каждый процессор несет ответственность за поддержание совместно используемых ресурсов, таких как I/O, кэши, TLB и памяти взаимно блокированными для обеспечения согласованности. Как правило, для поддержания согласованности кэша используются отслеживающие технологии. С целью облегчения совместного использования, в отслеживающем окружении каждая строка кэша может быть отмечена как пребывающая в любом из следующих состояний: совместно используемое, исключительное, измененное, недопустимое и т.п.

Устройства I/O 5054 (фиг. 17) предоставляют процессору средства для присоединения к периферийным устройствам, включающим в себя, например, устройства записи на ленту, дисководы, принтеры, дисплеи и сети. Устройства I/O зачастую предоставляются компьютерной программе посредством программных драйверов. В мэйнфреймах, таких как System z от IBM®, канальные адаптеры и адаптеры открытой системы являются устройствами I/O мэйнфрейма, предоставляющими сообщение между операционной системой и периферийными устройствами.

Кроме того, и другие типы вычислительных окружений могут извлечь выгоду из

одного или нескольких аспектов. Как пример, окружение может включать в себя эмулятор (например, программное обеспечение или другие механизмы эмуляции), в котором конкретная архитектура (включая сюда, например, выполнение команд, архитектурно спроектированные функции, такие как трансляция адресов и архитектурно спроектированные регистры) или подмножество этих элементов, эмулируется (например, на собственной компьютерной системе, имеющей процессор и память). В таком окружении одна или несколько функций эмуляции эмулятора могут реализовать один или несколько вариантов осуществления, также и в том случае, когда выполняющий эмуляцию компьютер может иметь архитектуру, отличную от таковой эмулируемых инструментов. В качестве примера, в режиме эмуляции, конкретная эмулируемая команда или операция декодируется, и создается соответствующая функция эмуляции для реализации отдельной команды или операции.

В среде эмуляции хост-компьютер включает в себя, например, память для хранения команд и данных, устройство выборки команд для выборки команд из памяти и, опционально, для предоставления локальной буферизации выбранной команды, устройство декодирования команд для получения выбранных команд и для выявления типа выбранных команд, и устройство выполнения команд для выполнения команд. Выполнение может включать в себя загрузку данных в регистр из памяти, сохранение данных обратно в памяти из регистра, или выполнение некоторого типа арифметической или логической операции, как выявлено модулем декодирования.

В одном примере, каждое устройство реализовано в программном обеспечении. Например, выполняемые устройствами операции реализуются в виде одной или нескольких подпрограмм в рамках программного обеспечения эмулятора.

Прежде всего, в мэйнфрейме архитектурно спроектированные машинные команды используются программистами, в настоящее время обычно программистами на «C», зачастую посредством компиляционного приложения.

Эти сохраненные в информационном носителе команды могут быть выполнены в исходной среде z/Архитектуры в IBM® Server или, альтернативно, в выполняющих другие архитектуры машинах. Они могут быть эмулированы в существующих и в будущих мэйнфреймовых серверах IBM®, а также на других машинах IBM® (например, серверах Power Systems и System x). Они могут быть выполнены в использующих Linux машинах на большом разнообразии машин с помощью аппаратных средств, изготовленных IBM®, Intel®, AMD и другими компаниями. Помимо выполнения на аппаратных средствах под архитектурой z/Архитектурой, может использоваться Linux, равно как и машины, использующие эмуляцию посредством Hercules, UMX или FSI (корпорация Fundamental Software), где в основном выполнение производится в режиме эмуляции. В режиме эмуляции программное обеспечение эмуляции выполняется собственным процессором для эмуляции архитектуры эмулируемого процессора.

Собственный процессор обычно выполняет программное обеспечение эмуляции, содержащее либо встроенное программное обеспечение, либо собственную операционную систему для выполнения эмуляции эмулируемого процессора. Программное обеспечение эмуляции ответственно за выборку и выполнение команд архитектуры эмулируемого процессора. Программное обеспечение эмуляции поддерживает эмулированный счетчик программы для отслеживания границ команд. Программное обеспечение эмуляции может одновременно выбирать одну или несколько эмулируемых машинных команд и преобразовывать одну или несколько эмулируемых машинных команд в соответствующую группу собственных машинных команд для выполнения собственным процессором. Эти преобразованные команды могут

кэшироваться таким образом, что может быть достигнуто более быстрое преобразование. Несмотря на это, программное обеспечение эмуляции должно поддерживать архитектурные правила архитектуры эмулируемого процессора таким образом, что обеспечивается правильное функционирование операционных систем и приложений, записанных для эмулируемого процессора. Кроме того, программное обеспечение эмуляции должно предоставлять ресурсы, идентифицируемые посредством архитектуры эмулируемого процессора, в том числе, но не ограничиваясь, регистры управления, регистры общего назначения, регистры с плавающей точкой, динамическую трансляцию адресов, включающую в себя, например, таблицы сегментов и таблицы страниц, механизмы прерывания, механизмы переключения контекста, часы времени суток (TOD), а также архитектурно спроектированные интерфейсы к подсистемам I/O таким образом, что операционная система или прикладная программа, разработанная для выполнения на эмулируемом процессоре, могут быть выполнены на собственном процессоре, имеющем программное обеспечение эмуляции.

Эмулируемая конкретная команда декодируется, и вызывается подпрограмма для выполнения функции отдельной команды. Функция эмуляционного программного обеспечения, эмулирующая функцию эмулируемого процессора реализуется, например, в подпрограмме «С» или в драйвере или некоторым другим способом обеспечения драйвера для специальных аппаратных средств, как должно быть ясно специалистам уровня техники после понимания описания предпочтительного варианта осуществления. Различные патенты по программной и аппаратной эмуляции, в том числе, но не ограничиваясь, патент на изобретение США №5 551 013, под названием «Многопроцессорная система для аппаратной эмуляции» (Multiprocessor for Hardware Emulation), Босолей и др. (Beausoleil et al), и патент на изобретение США №6 009 261, под названием «Предварительная обработка сохраненных целевых подпрограмм для эмуляции несовместимых команд на целевом процессоре» (Preprocessing of Stored Target Routines for Emulating Incompatible Instructions on a Target Processo), Скальци и др. (Scalzi et al), и патент на изобретение США №5 574 873, под названием «Гостевая команда декодирования для непосредственного доступа к подпрограммам эмуляции, которые эмулируют гостевые команды» (Decoding Guest Instruction to Directly Access Emulation Routines that Emulate the Guest Instructions), Давидян и др. (Davidian et al), и патент на изобретение США №6 308 255, под названием «Симметричная многопроцессорная шина и чипсет, используемые для поддержки сопроцессора и позволяющие выполнение в системе несобственного кода» (Symmetrical Multiprocessing Bus and Chipset Used for Coprocessor Support Allowing Non-Native Code to Run in a System), Горишек и др. (Gorishek et al), и патент на изобретение США №6 463 582, под названием «Динамический оптимизирующий транслятор объектного кода для эмуляции архитектуры и способ динамической оптимизирующей трансляции объектного кода» (Dynamic Optimizing Object Code Translator for Architecture Emulation and Dynamic Optimizing Object Code Translation Method), Летин и др. (Lethin et al), и патент на изобретение США №5 790 825, под названием «Способ эмуляции гостевых команд на главном компьютере посредством динамической рекомпиляции команд главного компьютера» (Method for Emulating Guest Instructions on a Host Computer Through Dynamic Recompilation of Host Instructions), Эрик Траут (Eric Traut), а также многие другие, демонстрируют ряд известных и доступных специалистам в данной области техники способов достижения эмуляции для целевой машины формата архитектурно спроектированных для различных машин команд.

На фиг. 19 предоставлен пример эмулированной системы 5092 хост-компьютера, которая эмулирует систему 5000' хост-компьютера хост-архитектуры. В эмулированной

системе 5092 хост-компьютера хост-процессор (ЦП) 5091 является эмулированным хост-процессором (или виртуальным хост-процессором) и содержит процессор 5093 эмуляции, имеющий собственную архитектуру системы команд, отличную от таковой процессора 5091 хост-компьютера 5000'. Эмулированная система 5092 хост-компьютера имеет память 5094, доступную для процессора 5093 эмуляции. В типовом варианте осуществления настоящего изобретения, память 5094 подразделяется на участок памяти 5096 хост-компьютера и участок подпрограмм 5097 эмуляции. Память хост-компьютера 5096 является доступной для программ эмулированного хост-компьютера 5092 согласно архитектуре хост-компьютера. Процессор 5093 эмуляции выполняет команды исходной среды архитектурно спроектированной системы команд архитектуры, отличной от таковой эмулированного процессора 5091, команды исходной среды, полученные из памяти 5097 подпрограмм эмуляции, а также может получать доступ к подлежащей выполнению команде хост-компьютера из программы в памяти 5096 хост-компьютера путем использования одной команды или нескольких команд, полученных в подпрограмме контроля последовательности команд и доступа/декодирования, которая может декодировать команду (команды) хост-компьютера, к которой получают доступ для выявления подпрограммы выполнения команды в исходной среде для эмуляции функции команды хост-компьютера, к которой получен доступ. Другие заданные для архитектуры системы 5000' хост-компьютера функции могут быть эмулированы посредством архитектурно спроектированных функциональных подпрограмм, включая сюда такие функции, например, как регистры общего назначения, регистры управления, динамическая трансляция адресов и поддержка подсистемы I/O и кэша процессора. Подпрограммы эмуляции могут также использовать в своих интересах доступные в процессоре 5093 эмуляции функции (такие как общие регистры и динамическая трансляция виртуальных адресов) для повышения производительности подпрограмм эмуляции. Специальное оборудование и разгрузочные подсистемы также могут быть предоставлены в помощь процессору 5093 при эмуляции функционирования хост-компьютера 5000'.

В другом варианте осуществления настоящего изобретения один или несколько аспектов относятся к облачным вычислениям. Заранее подразумевается, что хотя настоящее раскрытие включает в себя подробное описание облачных вычислений, реализация изложенных в настоящем документе идей не ограничивается окружением облачных вычислений. Скорее, варианты осуществления настоящего изобретения могут быть реализованы совместно с любым другим типом вычислительного окружения, известного в настоящее время или разработанным позднее.

Облачные вычисления являются моделью предоставления услуг для обеспечения возможности удобного, в нужный момент времени, сетевого доступа к совместно используемому комплексу конфигурируемых вычислительных ресурсов (например, сетей, сетевых пропускных способностей, серверов, вычислительных мощностей, памяти, запоминающих устройств, приложений, виртуальных машин и услуг), который может быть быстро предоставлен и передан с минимальными управленческими затратами или взаимодействием с поставщиком услуг. Эта облачная модель может включать в себя по меньшей мере пять характеристик, по меньшей мере три модели услуг и по меньшей мере четыре модели развертывания.

Характеристики представлены следующим образом.

Самообслуживание в нужный момент времени: потребитель облачных услуг может в одностороннем порядке обеспечивать вычислительные возможности, такие как время сервера и ресурсы сетевого хранения, по мере необходимости автоматически, без

необходимости в человеческом взаимодействии с провайдером услуги.

Широкий доступ к сети: возможности являются доступными по сети, а получение доступа производится посредством стандартных механизмов, что способствует использованию разнородных платформ тонкого или толстого клиента (например, мобильных телефонов, ноутбуков и PDA (личных электронных секретарей)).

Объединение ресурсов: вычислительные ресурсы провайдера объединены для обслуживания множественных потребителей, использующих модель мультиарендатора, причем различные физические и виртуальные ресурсы динамично присваиваются и повторно присваиваются согласно потребности. Смысл независимости от местоположения состоит в том, что потребитель, в большинстве случаев, не имеет никакого контроля или знания относительно точного местоположения предоставляемых ресурсов, но может быть в состоянии определить местоположение на более высоком уровне абстракции (например, страна, состояние или центр обработки данных).

Быстрая адаптационная способность: инструменты могут быть предоставлены быстрым и адаптивным способом, в некоторых случаях автоматически, для быстрого горизонтального масштабирования с увеличением производительности и быстро возвращены системе для быстрого горизонтального масштабирования с уменьшением производительности. Для потребителя доступные к предоставлению инструменты зачастую представляются неограниченными и могут быть приобретены в любом объеме в любое время.

Измеренная услуга: облачные системы автоматически управляют использованием ресурса и оптимизируют его путем усиления измерительного инструментария на некотором уровне абстракции, соответствующем типу услуги (например, хранения, обработки, пропускной способности и учетным записям активного пользователя). Использование ресурсов может быть подвергнуто мониторингу, управлению и протоколированию, что обеспечивает прозрачность как для провайдера, так и для потребителя используемой услуги.

Модели услуги представлены следующим образом.

Программное обеспечение как услуга (SaaS): потребителю предоставляется инструмент для использования приложений провайдера, работающих на облачной инфраструктуре. Приложения являются доступными с различных клиентских устройств через интерфейс тонкого клиента, такой как веб-браузер (например, интернет-почта). Потребитель не администрирует и не управляет базовой облачной инфраструктурой, содержащей сеть, серверы, операционные системы, запоминающие устройства или даже индивидуальные прикладные инструменты, за возможным исключением ограничений по настройке параметров конфигурации специфических для пользователя приложений.

Платформа как услуга (PaaS): потребителю предоставляется инструмент для развертывания на облачной инфраструктуре созданных или полученных потребителем приложений, созданных с помощью поддерживаемых провайдером языков программирования и программных средств. Потребитель не администрирует и не управляет базовой облачной инфраструктурой, содержащей сеть, серверы, операционные системы или запоминающие устройства, но управляет развертываемыми приложениями и, возможно, приложением, размещающим настройки среды.

Инфраструктура как услуга (IaaS): потребителю предоставляется инструмент для обеспечения обработки, хранения, сетей и других фундаментальных вычислительных ресурсов, с помощью которого потребитель может разворачивать и выполнять произвольное программное обеспечение, которое может включать в себя операционные системы и приложения. Потребитель не администрирует и не управляет базовой

облачной инфраструктурой, но управляет операционными системами, запоминающими устройствами, развертываемыми приложениями и, возможно, имеет ограниченное управление выбором сетевых компонентов (например, межсетевых защитных экранов хоста).

Модели развертывания представлены следующим образом.

Частное облако: облачная инфраструктура действует исключительно для организации. Оно может управляться посредством организации или третьей стороны, и может быть реализовано внутри или вовне организации.

Коллективное облако: облачную инфраструктуру совместно используют несколько организаций и поддерживает особое сообщество, которое имеет совместные интересы (например, главная цель работы, требования к защите, политика и соображения по контролю за соблюдением установленных требований). Оно может управляться посредством организации или третьей стороны, и может быть реализовано внутри или вовне организации.

Открытое облако: облачная инфраструктура сделана доступной для широкой публики или группы крупной отрасли, и принадлежит продающей облачные услуги организации.

Гибридное облако: облачная инфраструктура является сочетанием двух или более облаков (частного, коллективного или отрытого), которые остаются отличными от других объектами, но связаны друг с другом посредством стандартизированной или проприетарной технологии, обеспечивающей для данных и приложений мобильность (например, временное использование ресурсов открытого облака для выравнивания нагрузки между облаками).

Окружение облачных вычислений является услугой, ориентированной с вниманием на бесструктурность, слабую связность, модульный принцип и семантическую функциональную совместимость. В основе облачных вычислений находится инфраструктура, содержащая сеть из связанных узлов.

На фиг. 20 показано схематическое изображение примера узла облачных вычислений. Узел 6010 облачных вычислений является только одним примером подходящего узла облачных вычислений, и не предназначается для предположения какого-либо ограничения относительно объема использования или функциональности предпочтительных вариантов осуществления описанного в настоящем документе изобретения. В любом случае, узел 6010 облачных вычислений способен к своей реализации и/или к выполнению любой из сформулированных выше функциональностей.

В узле 6010 облачных вычислений имеется компьютерная система/сервер 6012, которая способна функционировать совместно с многочисленными другими окружениями или конфигурациями вычислительных систем общего назначения или особого назначения. Примеры известных вычислительных систем, окружений и/или конфигураций, которые могут подойти для использования с компьютерной системой/сервером 6012, включают в себя, но ими не ограничиваются, системы персонального компьютера, системы сервера, тонкие клиенты, толстые клиенты, наладонные или переносные устройства, многопроцессорные системы, основанные на микропроцессорах системы, декодеры каналов кабельного телевидения, программируемую бытовую электронику, сетевые персональные компьютеры, системы миникомпьютера, мэйнфреймовые компьютерные системы и распределенные окружения облачных вычислений, включающие в себя любые из вышеупомянутых систем или устройств, и т.п.

Компьютерная система/сервер 6012 может быть описана в общем контексте исполнимых команд компьютерной системы, таких как программные модули,

выполняемые компьютерной системой. Обычно, программные модули могут включать в себя подпрограммы, программы, объекты, компоненты, логику, структуры данных и так далее, которые выполняют конкретные задачи или реализуют конкретные абстрактные типы данных. Компьютерная система/сервер 6012 может быть осуществлена в распределенных окружениях облачных вычислений, где задачи выполняются отдаленными устройствами обработки, связанными через коммуникационную сеть. В распределенном окружении облачных вычислений программные модули могут быть расположены в информационных носителях как системы локального, так и удаленного компьютера, включая сюда запоминающие устройства.

Как показано на фиг. 20, компьютерная система/сервер 6012 в узле 6010 облачных вычислений показана в виде вычислительного устройства общего назначения. Компоненты компьютерной системы/сервера 6012 могут включать в себя, но не ими ограничиваются, один или несколько процессоров или вычислительных устройств 6016, системную память 6028, и шину 6018, которая соединяет с процессором 6016 различные компоненты системы, включающие в себя системную память 6028.

Шина 6018 представляет собой один или более из числа любых нескольких типов структур шины, включая сюда шину памяти или контроллер памяти, периферийную шину, ускоренный графический порт, и шину процессора или локальную шину, использующую любую из ряда шинных архитектур. В качестве примера, но не ограничения, такая архитектура включает в себя шину Промышленной стандартной архитектуры (ISA), шину Микроканальной архитектуры (MCA), шину Расширенной ISA (EISA), локальную шину Ассоциации по стандартам в области видео-электроники (VESA) и шину Взаимодействия периферийных компонентов (PCI).

Компьютерная система/сервер 6012 обычно включает в себя ряд считываемых компьютерной системой носителей. Такие носители могут быть представлены любыми доступными носителями, который являются доступными посредством компьютерной системы/сервера 6012, и которые включают в себя как энергозависимые, так и энергонезависимые носители, съемные и несъемные носители.

Системная память 6028 может включать в себя считываемые компьютерной системой носители в виде энергозависимой памяти, такие как оперативная память (RAM) 6030 и/или кэш-память 6032. Компьютерная система/сервер 6012, кроме того, может включать в себя и другие съемные и несъемные, энергозависимые и энергонезависимые информационные носители компьютерной системы. Исключительно в качестве примера, может быть предоставлена система 6034 хранения для считывания из несъемных, энергонезависимых магнитных носителей (не показанных и обычно называемых «жестким диском») и для записи в них. Хотя не показаны, могут быть предоставлены магнитный дисковод для считывания из несъемного, энергонезависимого магнитного диска (например, «гибкого диска») и для записи в него, и оптический дисковод для считывания из несъемного, энергонезависимого оптического диска, такого как CD-ROM, DVD-ROM или другие оптические носители, и для записи в них. В таких реализациях каждый дисковод может быть присоединен к шине 6018 посредством одного или нескольких интерфейсов носителей данных. Как будет, кроме того, изображено и описано ниже, память 6028 может включать в себя по меньшей мере один программный продукт, имеющий набор (например, по меньшей мере один) программных модулей, которые сконфигурированы для выполнения функций вариантов осуществления настоящего изобретения.

В качестве примера, но не ограничения, программа/сервисная программа 6040,

имеющая набор (по меньшей мере один) программных модулей 6042, может быть сохранена в памяти 6028, равно как операционная система, одна или несколько прикладных программ, другие программные модули и данные программы. Каждый элемент из числа операционной системы, одной или нескольких прикладных программ, других программных модулей и данных программы или некоторая комбинация из них может включать в себя реализацию сетевого окружения. Программные модули 6042 в основном выполняют функции и/или методологии предпочтительных вариантов осуществления настоящего изобретения, как они описаны в настоящем документе.

Компьютерная система/сервер 6012 может также сообщаться с одним или несколькими внешними устройствами 6014, такими как клавиатура, позиционирующее устройство, дисплей 6024, и т.д., с одним или несколькими устройствами, обеспечивающими пользователю взаимодействие с компьютерной системой/сервером 6012, и/или любыми устройствами (например, сетевой платой, модемом и т.д.), которые обеспечивают компьютерной системе/серверу 6012 сообщение с одним или несколькими другими вычислительными устройствами. Такая коммуникация может происходить через вводные/выводные (I/O) интерфейсы 6022. Кроме того, компьютерная система/сервер 6012 может сообщаться через сетевой адаптер 6020 с одной или несколькими сетями, такими как локальная сеть (LAN), общая глобальная сеть (WAN) и/или сеть общего пользования (например, Интернет). Как изображено, сетевой адаптер 6020 сообщается с другими компонентами компьютерной системы/сервера 6012 через шину 6018. Следует понимать, что совместно с компьютерной системой/сервером 6012 могут быть использованы, хотя и не показаны, другие аппаратные компоненты и/или компоненты программного обеспечения. Примеры, в том числе, но не ограничиваясь: микрокод, драйверы устройств, резервные вычислительные устройства, массивы внешних дисководов, системы RAID (массивы недорогих дисковых накопителей с избыточностью), устройства записи на ленту, системы архивного хранения данных и т.д.

На фиг. 21 изображено показательное окружение 6050 облачных вычислений. Как показано, окружение 6050 облачных вычислений содержит один или несколько узлов 6010 облачных вычислений, с помощью которых могут сообщаться используемые облачными потребителями локальные вычислительные устройства, такие как, например, персональный цифровой секретарь (PDA) или сотовый телефон 6054A, настольный компьютер 6054 B, портативный компьютер 6054C, и/или автомобильная компьютерная система 6054N. Узлы 6010 могут сообщаться друг с другом. Они могут быть сгруппированы (не показано) физически или виртуально, в одну или несколько сетей, таких как частное, коллективное, открытое или гибридное облака, как описано выше, или в комбинацию из них. Это позволяет окружению 6050 облачных вычислений предлагать инфраструктуру, платформы и/или программное обеспечение в качестве услуг, для получения которых облачный потребитель не должен поддерживать ресурсы на локальном вычислительном устройстве. Подразумевается, что показанные на фиг. 28 типы 6054A-N вычислительных устройств предназначаются исключительно для разъяснений и, что вычислительные узлы 6010 и окружение 6050 облачных вычислений могут сообщаться с любым типом компьютеризированного устройства по любому типу сетевого и/или адресуемого через сеть присоединения (например, с помощью веб-браузера).

На фиг. 22 показан ряд функциональных уровней абстракции, предоставляемых окружением 6050 облачных вычислений (фиг. 21). Прежде всего, необходимо понимать, что компоненты, уровни и функции, показанные на фиг. 22, предназначаются исключительно для разъяснений, и ими не ограничиваются варианты осуществления

изобретения. Как изображено, предоставляются следующие уровни и соответствующие функции.

Аппаратный и программный уровень 6060 включает в себя аппаратные и программные компоненты. Примеры аппаратных компонентов включают в себя мэйнфреймы, в одном примере, системы IBM® zSeries®, основанные на архитектуре RISC (компьютер с сокращенной системой команд) серверы, в одном примере, системы IBM pSeries®, системы IBM xSeries®, системы IBM BladeCenter®, запоминающие устройства, сети и сетевые компоненты. Примеры компонентов программного обеспечения включают в себя серверное программное обеспечение сетевых приложений, в одном примере, серверное программное обеспечение приложений IBM WebSphere®, и программное обеспечение базы данных, в одном примере, программное обеспечение базы данных IBM DB2®. IBM, zSeries, pSeries, xSeries, BladeCenter, WebSphere, а также DB2, z/OS, z/VM, Z/Архитектура и Processor Resource/Systems Manager являются торговыми марками International Business Machines Corporation, зарегистрированными в нескольких юрисдикциях по всему миру. Другие используемые здесь наименования могут быть представлены зарегистрированными торговыми марками, торговыми марками или названиями продукта International Business Machines Corporation или других компаний.

Уровень 6062 виртуализации предоставляет уровень абстракции, на основе которого могут быть предоставлены следующие примеры виртуальных объектов: виртуальные серверы, виртуальная память, виртуальные сети, включающие в себя виртуальные частные сети, виртуальные приложения и операционные системы, и виртуальные клиенты.

В одном примере, уровень 6064 управления может предоставлять описанные ниже функции. Функция выделения ресурсов предоставляет динамическую поставку вычислительных ресурсов и других ресурсов, использующихся для выполнения задач в пределах окружения облачных вычислений. Функции измерения и оценки предоставляют отслеживание затрат по мере использования ресурсов в пределах окружения облачных вычислений, а также выписку или выставление счетов за потребление этих ресурсов. В одном примере, эти ресурсы могут содержать лицензии прикладного программного обеспечения. Функция безопасности предоставляет проверку идентификационных данных для облачных потребителей и задач, а также защиту данных и других ресурсов. Функция пользовательского портала предоставляет доступ к окружению облачных вычислений для потребителей и системных администраторов. Функция управления уровнем услуг предоставляет распределение ресурсов облачных вычислений и управление ими таким образом, что удовлетворяются необходимые уровни услуг. Функция планирования и выполнения Соглашения об уровне услуг (SLA) предоставляет предварительную подготовку и приобретение ресурсов облачных вычислений, для которых в будущем ожидается соответствие требованиям согласно SLA.

Уровень 6066 рабочих заданий предоставляет примеры функциональности, для которой может быть использовано окружение облачных вычислений. Примеры рабочих заданий и функций, которые могут быть предоставлены на этом уровне, включают в себя: построение соответствий, а также просмотр и организацию доступа к ресурсам, разработку программного обеспечения и управление его жизненным циклом, предоставление образования в виде виртуальной аудитории, обработку анализа данных, и обработку транзакций.

Используемая в настоящем документе терминология служит исключительно целям описания конкретных вариантов осуществления и не предназначается для ограничения

изобретения. При использовании в настоящем документе, формы единственного числа предназначены для включения в себя также и форм множественного числа, если только контекст не указывает на иное недвусмысленным образом. В последующем изложении подразумевается, что термины «содержит» и/или «содержащий» при их использовании в данном техническом описании задают присутствие заявленных признаков, целочисленных переменных, этапов, операций, элементов и/или компонентов, но не исключают присутствия или добавления одного или нескольких других признаков, целочисленных переменных, этапов, операций, элементов, компонентов и/или образованных ими групп.

Соответствующие структуры, материалы, действия и эквиваленты всех средств или этапов, равно как функциональные элементы в пунктах формулы изобретения ниже, если вообще есть в наличии, предназначаются для включения в себя любой структуры, материала или действия для выполнения функции в сочетании с другими требуемыми элементами, как конкретным образом заявлено. Описание одного или нескольких вариантов осуществления представлено в целях иллюстрации и описания, но не предназначается для полного охвата или ограничения изобретения в заявленном виде. Многие модификации и изменения являются очевидными для средних специалистов в области техники. Вариант осуществления был выбран и описан с целью наилучшего объяснения различных аспектов и практического применения, а также для обеспечения другим средним специалистам в области техники возможности понимания различных вариантов осуществления с различными модификациями, как они подходят для конкретно рассматриваемого использования.

(57) Формула изобретения

1. Способ реконфигурирования вычислительного окружения, причем способ содержит:

- выявление посредством процессора того, что средство конфигурации архитектурного режима инсталлировано в вычислительном окружении, сконфигурированном для нескольких архитектурных режимов и имеющем заданную последовательность включения, которая предназначена для включения вычислительного окружения в одном архитектурном режиме из нескольких архитектурных режимов, причем один архитектурный режим содержит первую архитектуру системы команд и имеет первый набор поддерживаемых сервисов,

- на основании выявления того, что средство конфигурации архитектурного режима инсталлировано, проведение посредством процессора реконфигурирования вычислительного окружения для ограничения использования одного архитектурного режима, причем реконфигурирование включает в себя:

- выборку отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме из нескольких архитектурных режимов, причем другой архитектурный режим отличен от одного архитектурного режима и другой архитектурный режим содержит вторую архитектуру системы команд и имеет второй набор поддерживаемых сервисов, и

- выполнение отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме вместо одного архитектурного режима, ограничивая использование одного архитектурного режима.

2. Способ по п. 1, причем выполнение отличной последовательности включения содержит создание нового слова состояния программы для управления операциями вычислительного окружения в другом архитектурном режиме, и причем создание нового

слова состояния программы содержит инвертирование значения индикатора архитектурного режима в новом слове состояния программы для указания на другой архитектурный режим.

3. Способ по п. 2, причем создание нового слова состояния программы содержит формирование нового слова состояния программы с форматом, указанным посредством другого архитектурного режима, причем формат содержит расширение адресного поля от первого размера до второго размера, а также выполнение инвертирования индикатора архитектурного режима.

4. Способ по одному из предшествующих пунктов, причем выявление того, что средство конфигурации архитектурного режима инсталлировано, содержит проверку индикатора средства, причем индикатор средства задается безусловным образом или под управлением индикатора конфигурации.

5. Способ по одному из предшествующих пунктов, причем реконфигурирование, кроме того, содержит отключение в пределах вычислительного окружения одной или нескольких операций для поддержки одного архитектурного режима, причем одна или несколько операций содержат операцию переключения для переключения от другого архитектурного режима к одному архитектурному режиму, причем переключение обратно к одному архитектурному режиму отключается.

6. Способ по п. 5, причем отключение содержит изменение обработки команды процессора обработки сигналов для предоставления ошибки на основании запроса на переключение обратно к одному архитектурному режиму.

7. Способ по предшествующим пунктам, причем способ, кроме того, содержит выполнение сброса по меньшей мере одного процессора вычислительного окружения, причем выполнение сброса содержит:

- приведение в исходное состояние вычислительного окружения в другом архитектурном режиме, причем приведение в исходное состояние содержит задание архитектурному режиму вычислительного окружения другого архитектурного режима, и

- инвертирование значения индикатора архитектурного режима в слове состояния программы для указания на другой архитектурный режим, причем

слово состояния программы используется для управления операциями вычислительного окружения.

8. Способ по предшествующим пунктам, причем реконфигурирование содержит обработку изменения операции процессора обработки сигналов, причем операция процессора обработки сигналов для задания архитектурному режиму вычислительного окружения того архитектурного режима, в котором оно в настоящее время находится, имеет результатом сохранение состояния, указывающего на нахождение вычислительного окружения в настоящее время в данном архитектурном режиме, причем это состояние обрабатывается посредством выпускающего операции процессора обработки сигналов как приемлемое.

9. Способ по предшествующим пунктам, причем один архитектурный режим является устаревшим режимом, а другой архитектурный режим является расширенным режимом, и причем первый набор поддерживаемых сервисов содержит 31-битовую адресацию и использует 32-битовые регистры общего назначения, а второй набор поддерживаемых сервисов содержит 64-битовую адресацию и использует 64-битовые регистры общего назначения.

10. Способ по предшествующим пунктам, причем вычислительное окружение является виртуальным гостевым окружением, имеющим хост-процессор, первую гостевую

виртуальную машину на первом уровне виртуализации и вторую гостевую виртуальную машину на втором уровне виртуализации, и причем реконфигурирование выполняется для хост-процессора и первой гостевой виртуальной машины, но не для второй гостевой виртуальной машины, и причем вторая гостевая виртуальная машина инициируется и производит обработку в одном архитектурном режиме.

11. Способ конфигурирования вычислительного окружения, причем способ содержит:

- конфигурирование посредством процессора вычислительного окружения для выполнения операций в выбранном архитектурном режиме, причем конфигурирование содержит:

- начало инициализации вычислительного окружения с использованием сохраненного слова состояния программы, причем сохраненное слово состояния программы имеет формат архитектурного режима, отличающегося от выбранного архитектурного режима,

- выявление того, что сохраненное слово состояния программы имеет формат архитектурного режима, отличающегося от выбранного архитектурного режима,

- на основании выявления того, что сохраненное слово состояния программы имеет формат архитектурного режима, отличающегося от выбранного архитектурного режима, проведение автоматического изменения сохраненного слова состояния программы для приобретения им формата выбранного архитектурного режима, причем автоматическое изменение выполняется в отсутствии явного запроса на переключение на выбранный архитектурный режим, и

- завершение инициализации вычислительного окружения с помощью измененного слова состояния программы для конфигурирования вычислительного окружения в выбранном архитектурном режиме.

12. Компьютерная система для реконфигурирования вычислительного окружения, причем компьютерная система содержит:

- память, и

- соединенный с памятью процессор, причем компьютерная система сконфигурирована для осуществления способа, причем способ содержит:

- выявление посредством процессора того, что средство конфигурации архитектурного режима инсталлировано в вычислительном окружении, сконфигурированном для нескольких архитектурных режимов и имеющем заданную последовательность включения, которая предназначена для включения вычислительного окружения в одном архитектурном режиме из нескольких архитектурных режимов, причем один архитектурный режим содержит первую архитектуру системы команд и имеет первый набор поддерживаемых сервисов,

- на основании выявления того, что средство конфигурации архитектурного режима инсталлировано, проведение посредством процессора реконфигурирования вычислительного окружения для ограничения использования одного архитектурного режима, причем реконфигурирование включает в себя:

- выборку отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме из нескольких архитектурных режимов, причем другой архитектурный режим отличен от одного архитектурного режима, и другой архитектурный режим содержит вторую архитектуру системы команд и имеет второй набор поддерживаемых сервисов, и

- выполнение отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме вместо одного архитектурного режима, ограничивая использование одного архитектурного режима.

13. Компьютерная система по п. 12, причем выполнение отличной последовательности включения содержит создание нового слова состояния программы для управления операциями вычислительного окружения в другом архитектурном режиме, причем создание нового слова состояния программы содержит инвертирование значения индикатора архитектурного режима в новом слове состояния программы для указания на другой архитектурный режим.

14. Компьютерная система по п. 13, причем создание нового слова состояния программы содержит формирование нового слова состояния программы с форматом, указанным посредством другого архитектурного режима, причем формат предусматривает расширение адресного поля от первого размера до второго размера, а также выполнение инвертирования индикатора архитектурного режима.

15. Компьютерная система по одному из пп. 12-14, причем реконфигурирование, кроме того, содержит отключение в пределах вычислительного окружения одной или нескольких операций для поддержки одного архитектурного режима, причем одна или несколько операций содержат операцию переключения для переключения от другого архитектурного режима к одному архитектурному режиму, причем переключение обратно к одному архитектурному режиму отключается.

16. Компьютерная система по п. 15, причем отключение содержит изменение обработки команды процессора обработки сигналов для предоставления ошибки на основании запроса на переключение обратно к одному архитектурному режиму.

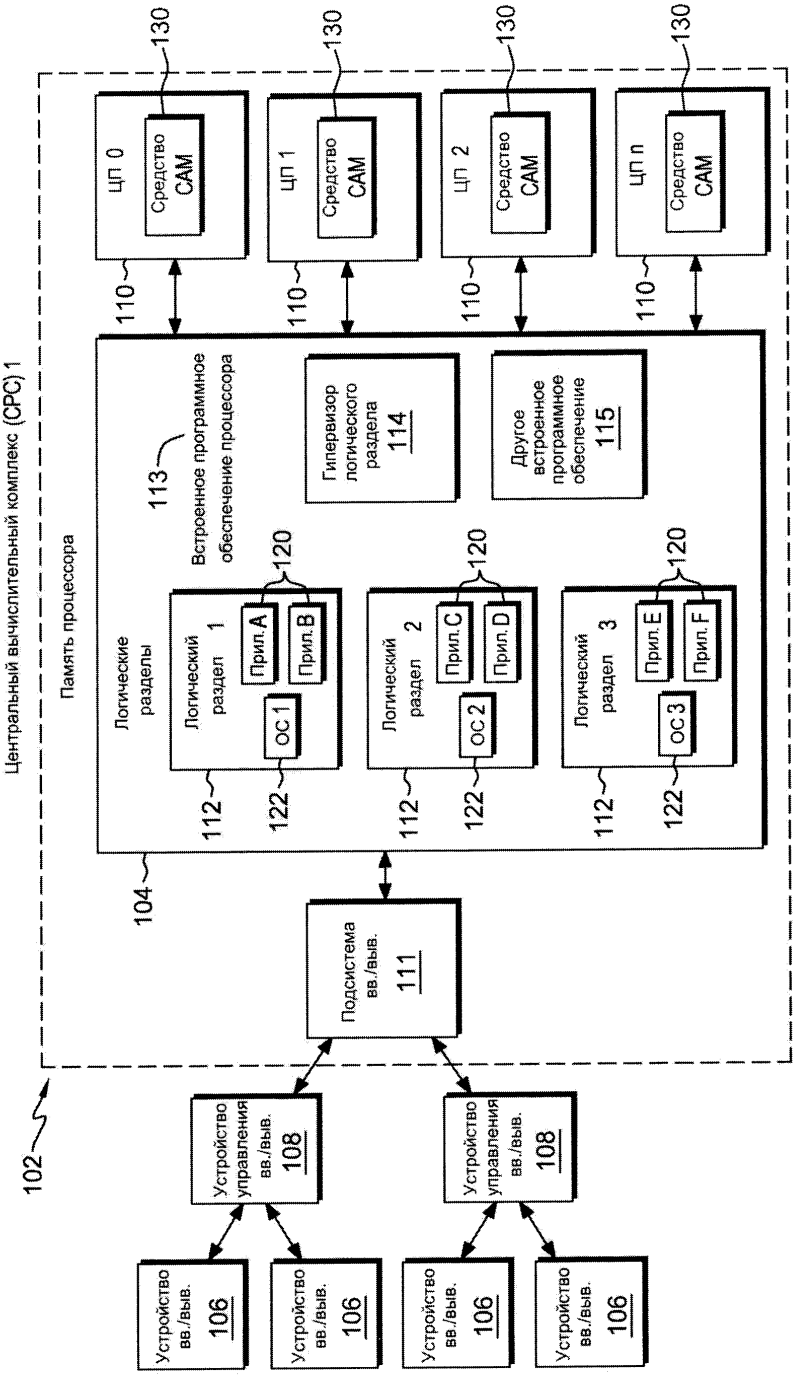
17. Компьютерная система по одному из пп. 12-16, причем реконфигурирование содержит обработку изменения операции процессора обработки сигналов, причем операция процессора обработки сигналов для задания архитектурному режиму вычислительного окружения того архитектурного режима, в котором оно в настоящее время находится, имеет результатом сохранение состояния, указывающего на нахождение вычислительного окружения в настоящее время в данном архитектурном режиме, причем это состояние обрабатывается посредством выпускающего операции процессора обработки сигналов как приемлемое.

18. Машиночитаемый информационный носитель, считываемый посредством устройства обработки данных и содержащий программу для реконфигурирования вычислительного окружения, включающую в себя команды, выполняемые посредством устройства обработки данных для осуществления способа по одному из пп. 1-10.

19. Машиночитаемый информационный носитель, считываемый посредством устройства обработки данных и содержащий программу для конфигурирования вычислительного окружения, включающую в себя команды, выполняемые посредством устройства обработки данных для осуществления способа по п. 11.

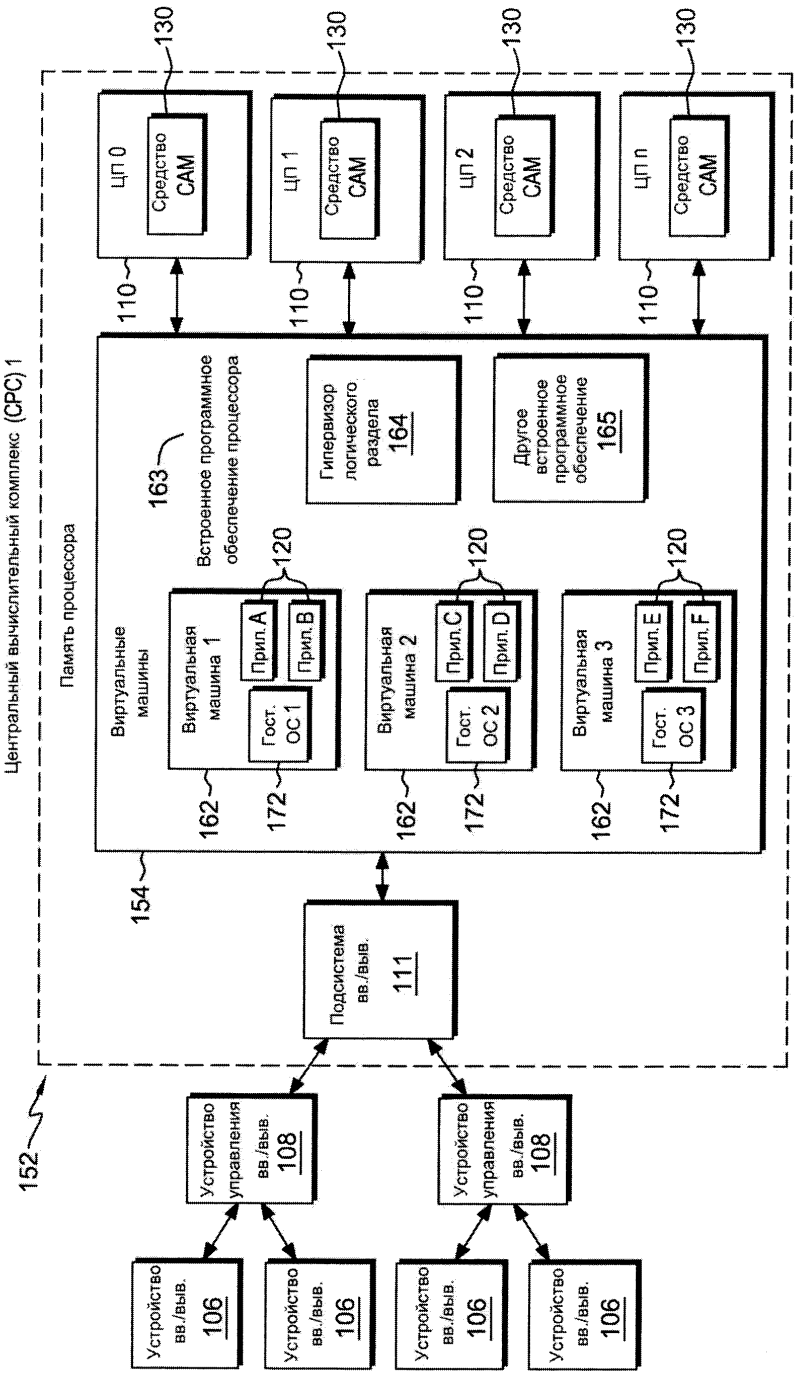
20. Машиночитаемый информационный носитель, в котором сохранена компьютерная программа, содержащая участки программного кода и загружаемая во внутреннюю память цифровой вычислительной машины, когда данная программа выполняется на компьютере для осуществления способа по одному из пп. 1-11.

1

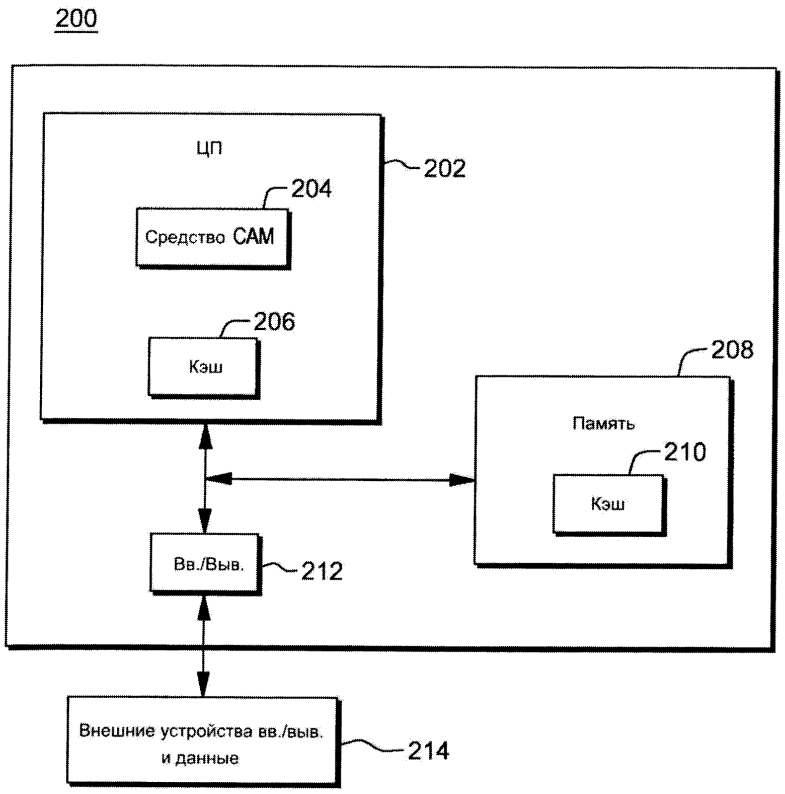

100
Центральный вычислительный комплекс (CPC) 1
Память процессора
Логические разделы
Логический раздел 1
Прил. A
Прил. B
ОС 1
Логический раздел 2
Прил. C
Прил. D
ОС 2
Логический раздел 3
Прил. E
Прил. F
ОС 3
120
112
122
113
Встроенное программное обеспечение процессора
Гипервизор логического раздела 114
Другое встроенное программное обеспечение 115
104
Подсистема вв./выв. 111
ЦП 0
ЦП 1
ЦП 2
ЦП n
Средство CAM
130
110
102
Устройство управления вв./выв. 108
Устройство вв./выв. 106

Фиг. 1A

2

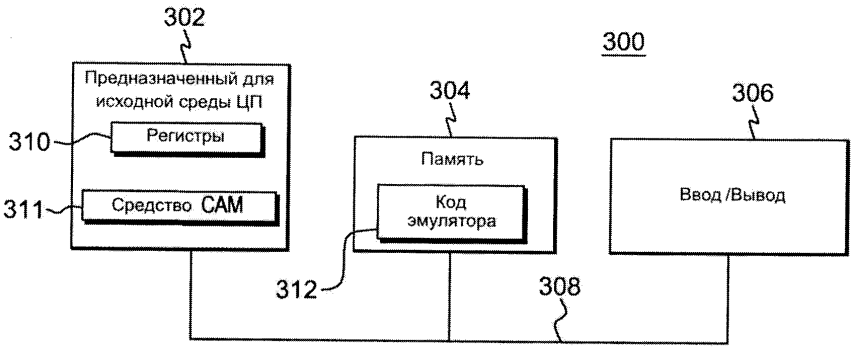

150
Центральный вычислительный комплекс (CPC) 1
Память процессора
Виртуальные машины
Встроенное программное обеспечение процессора
163
Гипервизор логического раздела 164
Другое встроенное программное обеспечение 165
Виртуальная машина 1
Виртуальная машина 2
Виртуальная машина 3
Прил.A
Прил.B
Прил.C
Прил.D
Прил.E
Прил.F
Гост. ОС 1
Гост. ОС 2
Гост. ОС 3
120
162
172
154
Цп 0
Цп 1
Цп 2
Цп n
Средство CAM
130
110
Подсистема вв./выв. 111
152
Устройство управления вв./выв. 108
Устройство вв./выв. 106

Фиг. 1Б

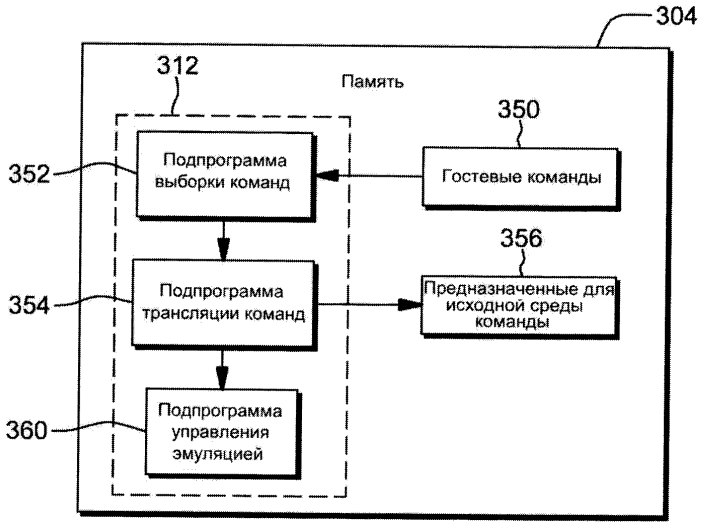

200
ЦП
202
204
Средство CAM
206
Кэш
208
Память
210
Кэш
Вв./Выв.
212
Внешние устройства вв./выв. и данные
214

Фиг. 2

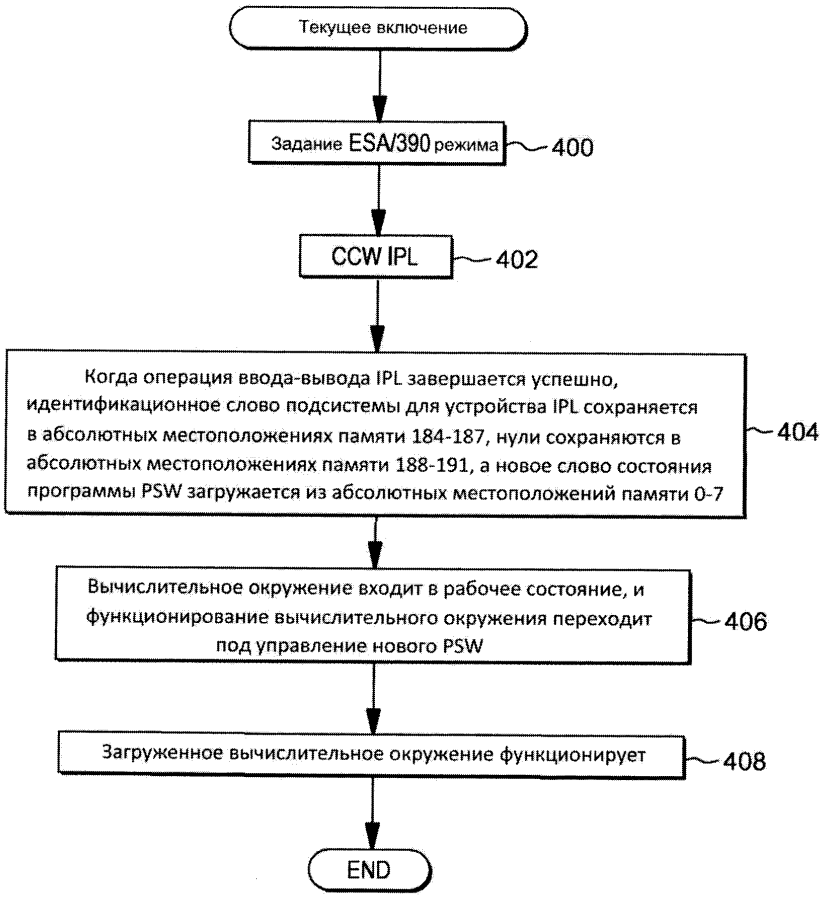
4


302
300
Предназначенный для исходной среды ЦП
Регистры
310
Средство CAM
311
304
Память
Код эмулятора
306
Ввод /Вывод
312
308

Фиг. 3А

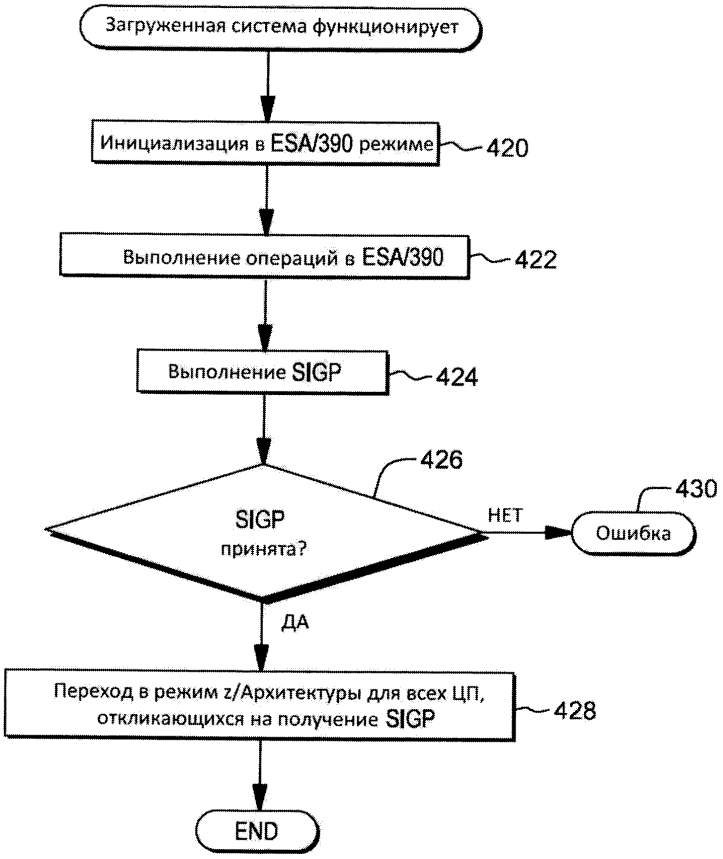

304
312
Память
350
Подпрограмма выборки команд
352
Гостевые команды
356
Подпрограмма трансляции команд
354
Предназначенные для исходной среды команды
Подпрограмма управления эмуляцией
360

Фиг. 3Б

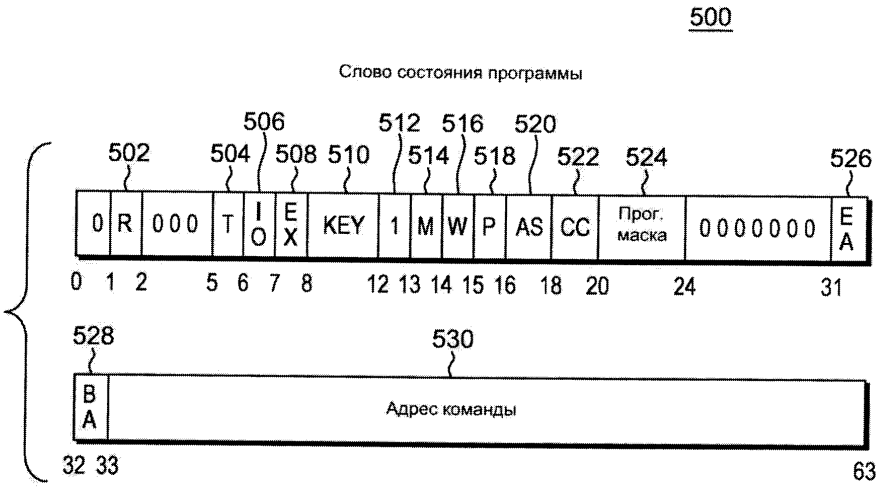

Текущее включение
Задание ESA/390 режима
400
CCW IPL
402
Когда операция ввода-вывода IPL завершается успешно, идентификационное слово подсистемы для устройства IPL сохраняется в абсолютных местоположениях памяти 184-187, нули сохраняются в абсолютных местоположениях памяти 188-191, а новое слово состояния программы PSW загружается из абсолютных местоположений памяти 0-7
404
Вычислительное окружение входит в рабочее состояние, и функционирование вычислительного окружения переходит под управление нового PSW
406
Загруженное вычислительное окружение функционирует
408
END

Фиг. 4A

6


Загруженная система функционирует
Инициализация в ESA/390 режиме
420
Выполнение операций в ESA/390
422
Выполнение SIGP
424
426
SIGP принята?
НЕТ
430
Ошибка
ДА
Переход в режим z/Архитектуры для всех ЦП, откликающихся на получение SIGP
428
END

Фиг. 4Б


500
Слово состояния программы
502
504
506
508
510
512
514
516
518
520
522
524
526
0
R
000
T
I O
E X
KEY
1
M
W
P
AS
CC
Прог. маска
0000000
E A
0 1 2 5 6 7 8 12 13 14 15 16 18 20 24 31
528
530
B A
Адрес команды
32 33
63

Фиг. 5

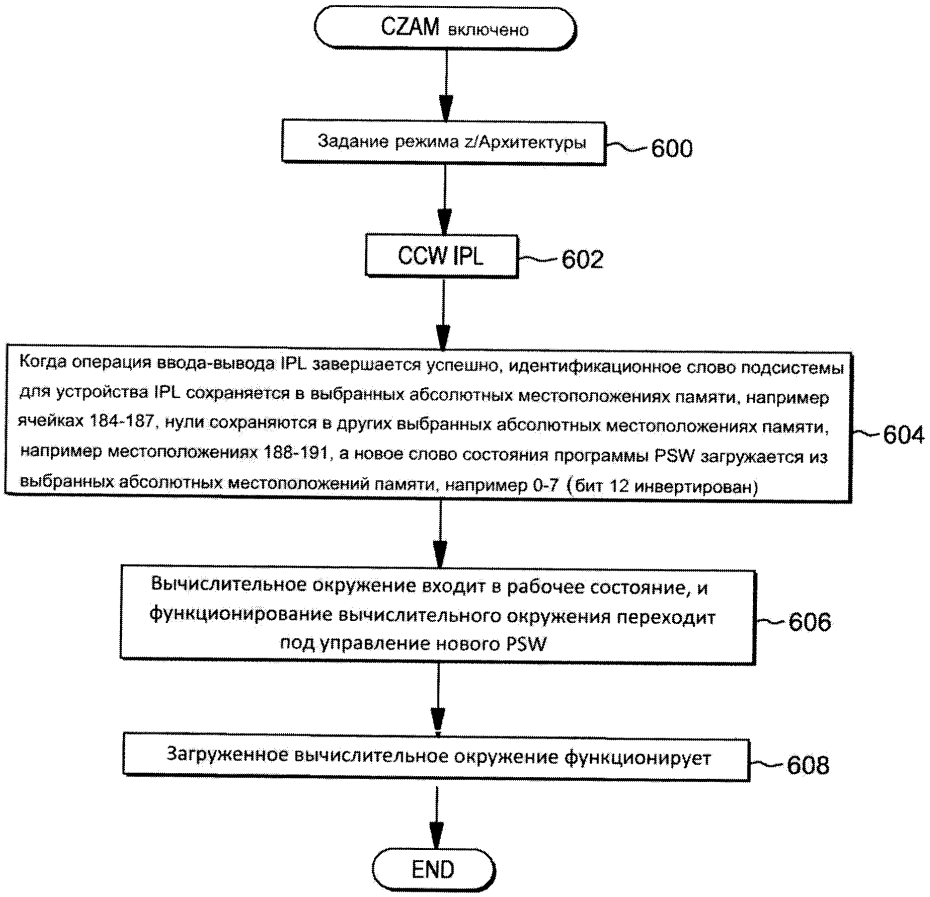
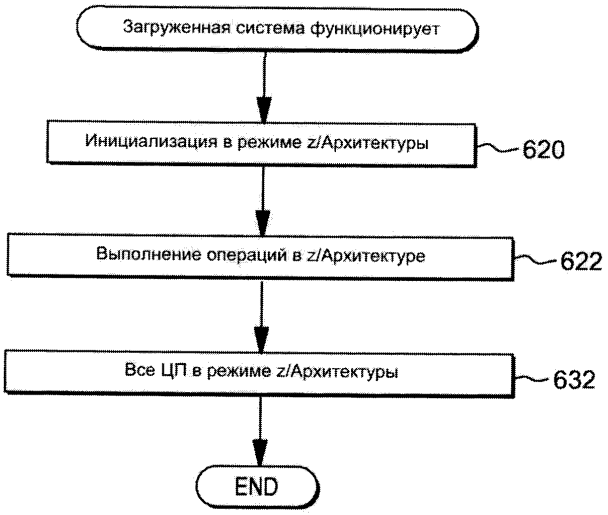

CZAM включено
Задание режима z/Архитектуры
600
CCW IPL
602
Когда операция ввода-вывода IPL завершается успешно, идентификационное слово подсистемы для устройства IPL сохраняется в выбранных абсолютных местоположениях памяти, например ячейках 184-187, нули сохраняются в других выбранных абсолютных местоположениях памяти, например местоположениях 188-191, а новое слово состояния программы PSW загружается из выбранных абсолютных местоположений памяти, например 0-7 (бит 12 инвертирован)
604
Вычислительное окружение входит в рабочее состояние, и функционирование вычислительного окружения переходит под управление нового PSW
606
Загруженное вычислительное окружение функционирует
608
END

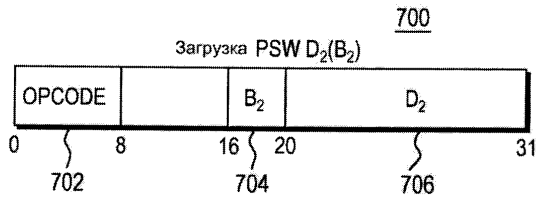
FIG. 6A

9


Загруженная система функционирует
Инициализация в режиме z/Архитектуры
620
Выполнение операций в z/Архитектуре
622
Все ЦП в режиме z/Архитектуры
632
END

Фиг. 6Б

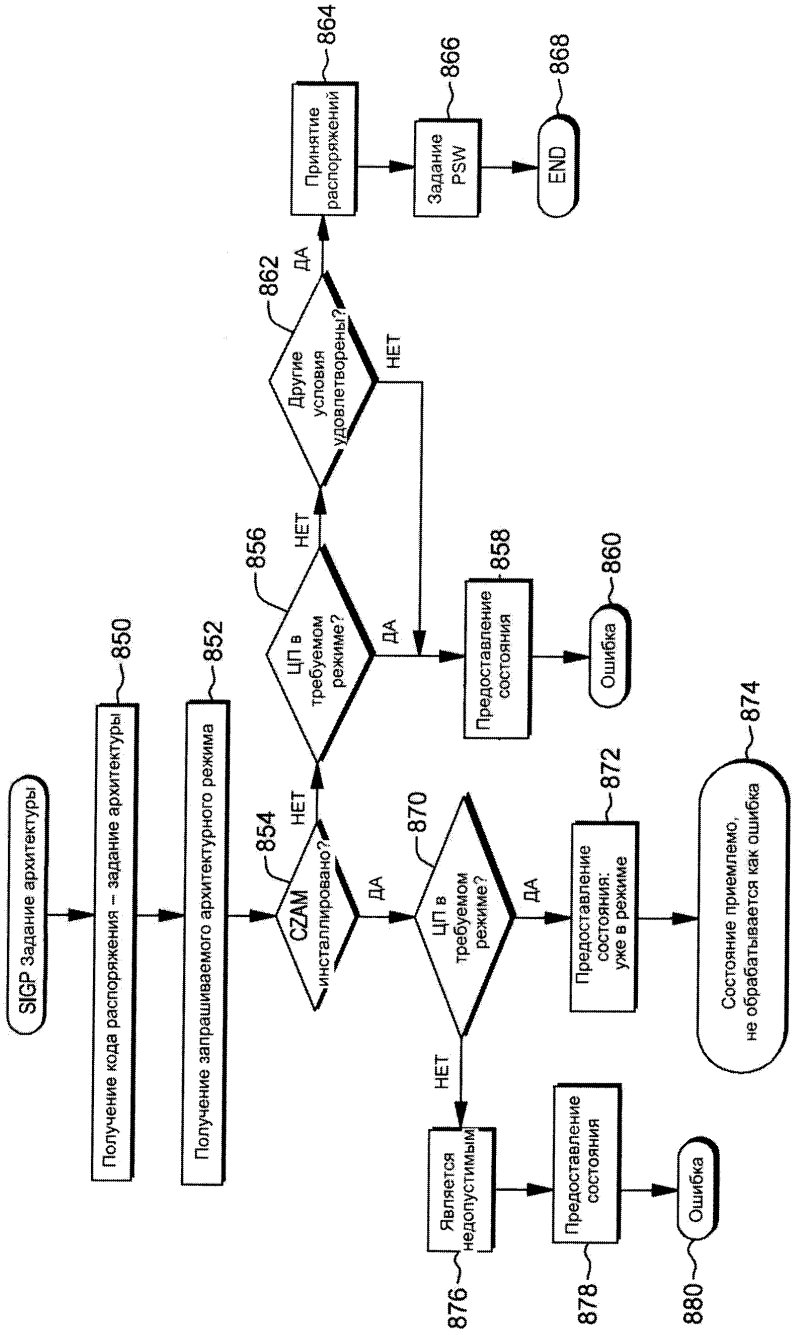
10


700
Загрузка PSW D2(B2)
OPCODE
B2
D2
0
8
16
20
31
702
704
706

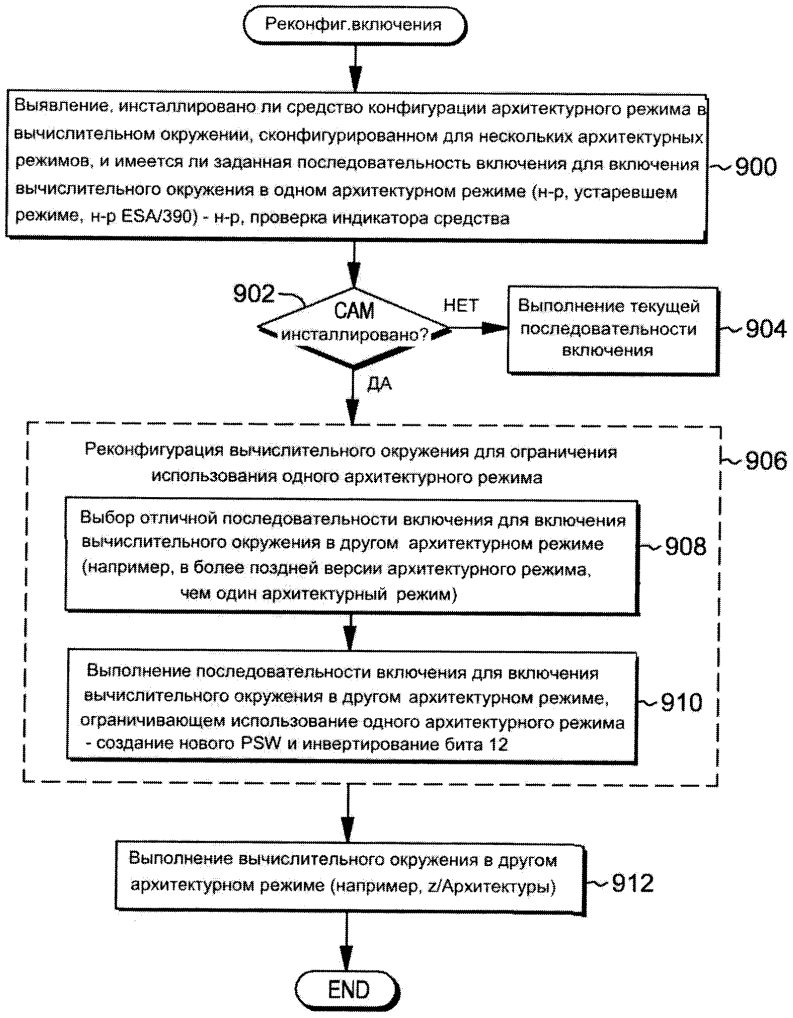
Фиг. 7


800
Процессор обработки сигналов R1, R3, D2(B2)
OPCODE
R1
R3
B2
D2
0
8
20
16
20
31
802
804
806
808
810

Фиг. 8A

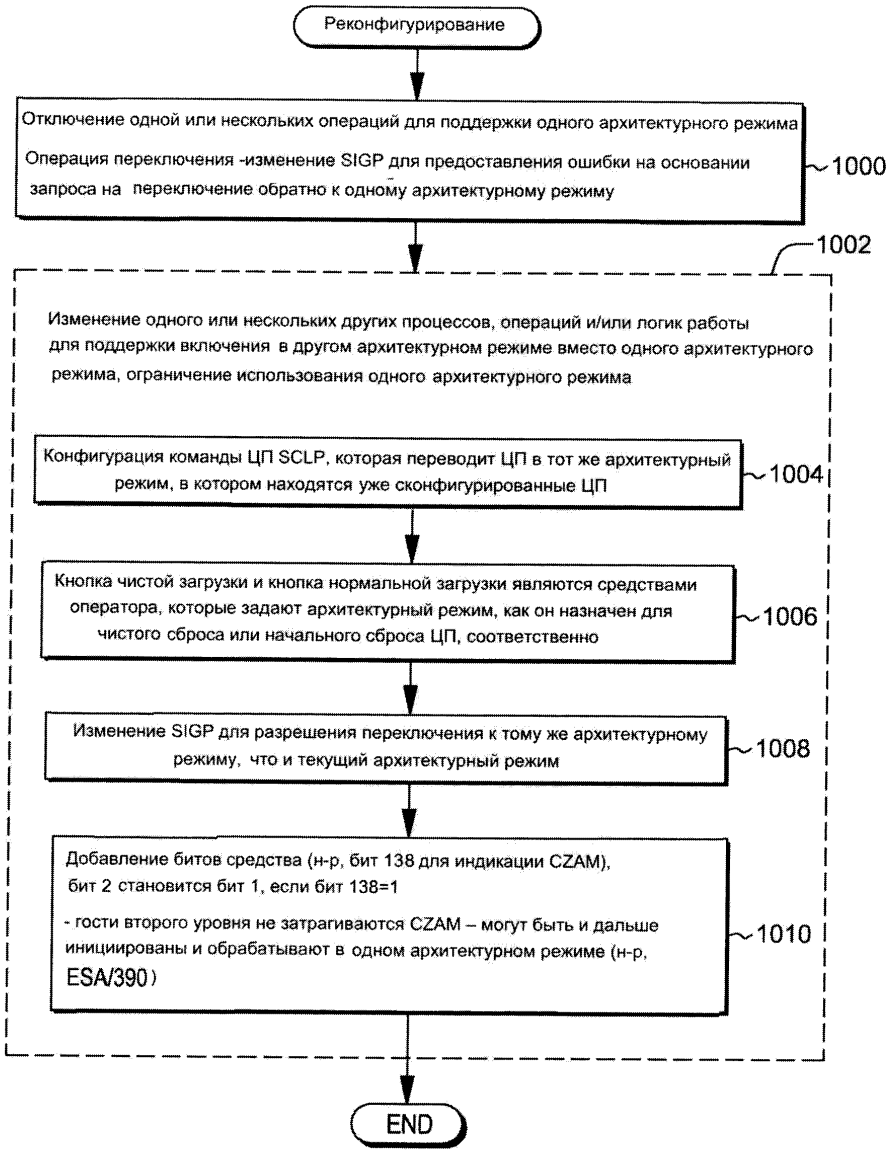

SIGP Задание архитектуры
Получение кода распоряжения – задание архитектуры
850
Получение запрашиваемого архитектурного режима
852
CZAM инсталлировано?
854
НЕТ
ДА
ЦП в требуемом режиме?
856
НЕТ
ДА
Другие условия удовлетворены?
862
ДА
НЕТ
Принятие распоряжений
864
Задание PSW
866
END
868
Предоставление состояния
858
Ошибка
860
ЦП в требуемом режиме?
870
ДА
НЕТ
Предоставление состояния: уже в режиме
872
Состояние приемлемо, не обрабатывается как ошибка
874
Является недопустимым
876
Предоставление состояния
878
Ошибка
880

Фиг. 8Б

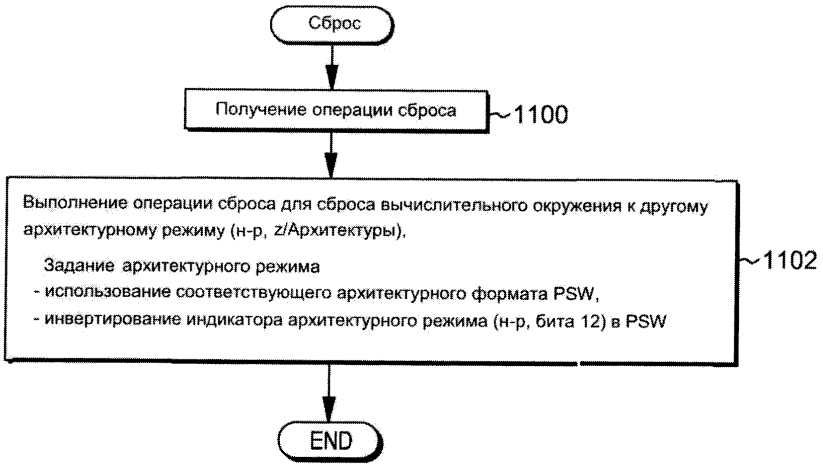

Реконфиг.включения
Выявление, инсталлировано ли средство конфигурации архитектурного режима в вычислительном окружении, сконфигурированном для нескольких архитектурных режимов, и имеется ли заданная последовательность включения для включения вычислительного окружения в одном архитектурном режиме (н-р, устаревшем режиме, н-р ESA/390) - н-р, проверка индикатора средства
900
902
CAM инсталлировано?
НЕТ
Выполнение текущей последовательности включения
904
ДА
Реконфигурация вычислительного окружения для ограничения использования одного архитектурного режима
906
Выбор отличной последовательности включения для включения вычислительного окружения в другом архитектурном режиме (например, в более поздней версии архитектурного режима, чем один архитектурный режим)
908
Выполнение последовательности включения для включения вычислительного окружения в другом архитектурном режиме, ограничивающем использование одного архитектурного режима - создание нового PSW и инвертирование бита 12
910
Выполнение вычислительного окружения в другом архитектурном режиме (например, z/Архитектуры)
912
END

Фиг. 9

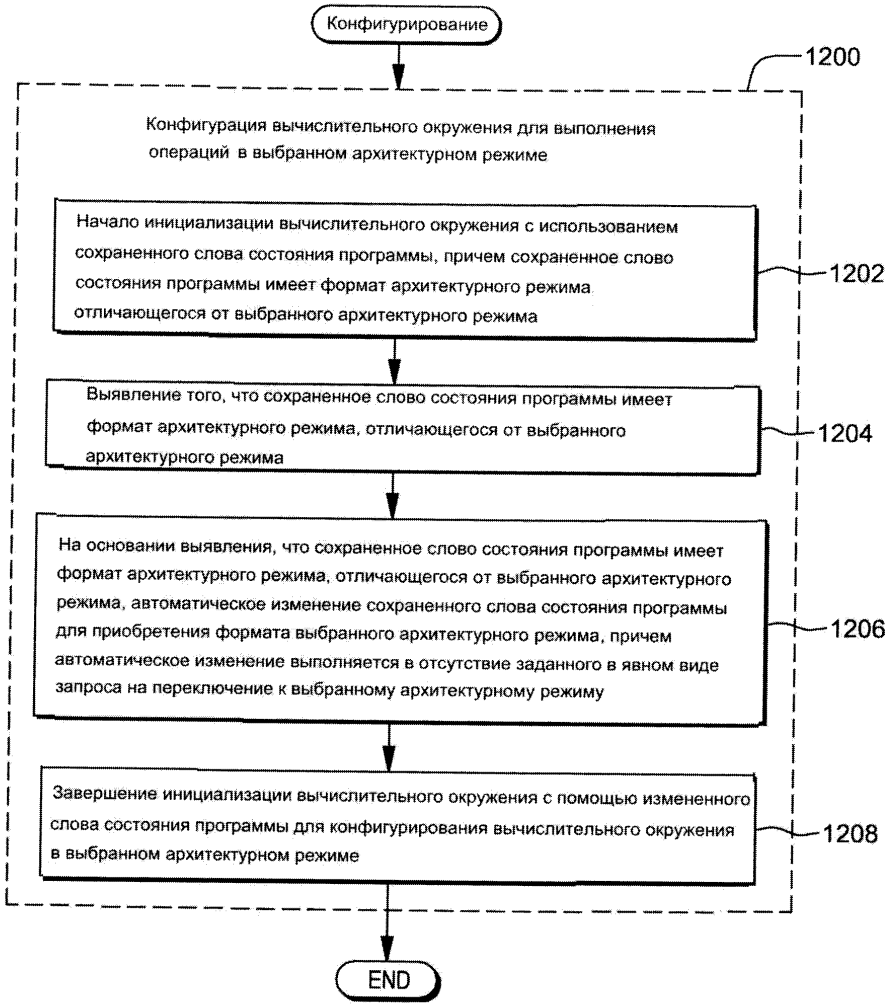
13


Реконфигурирование
Отключение одной или нескольких операций для поддержки одного архитектурного режима
Операция переключения -изменение SIGP для предоставления ошибки на основании запроса на переключение обратно к одному архитектурному режиму
1000
1002
Изменение одного или нескольких других процессов, операций и/или логик работы для поддержки включения в другом архитектурном режиме вместо одного архитектурного режима, ограничение использования одного архитектурного режима
Конфигурация команды ЦП SCLP, которая переводит ЦП в тот же архитектурный режим, в котором находятся уже сконфигурированные ЦП
1004
Кнопка чистой загрузки и кнопка нормальной загрузки являются средствами оператора, которые задают архитектурный режим, как он назначен для чистого сброса или начального сброса ЦП, соответственно
1006
Изменение SIGP для разрешения переключения к тому же архитектурному режиму, что и текущий архитектурный режим
1008
Добавление битов средства (н-р, бит 138 для индикации CZAM), бит 2 становится бит 1, если бит 138=1
- гости второго уровня не затрагиваются CZAM – могут быть и дальше инициированы и обрабатывают в одном архитектурном режиме (н-р, ESA/390)
1010
END

Фиг. 10

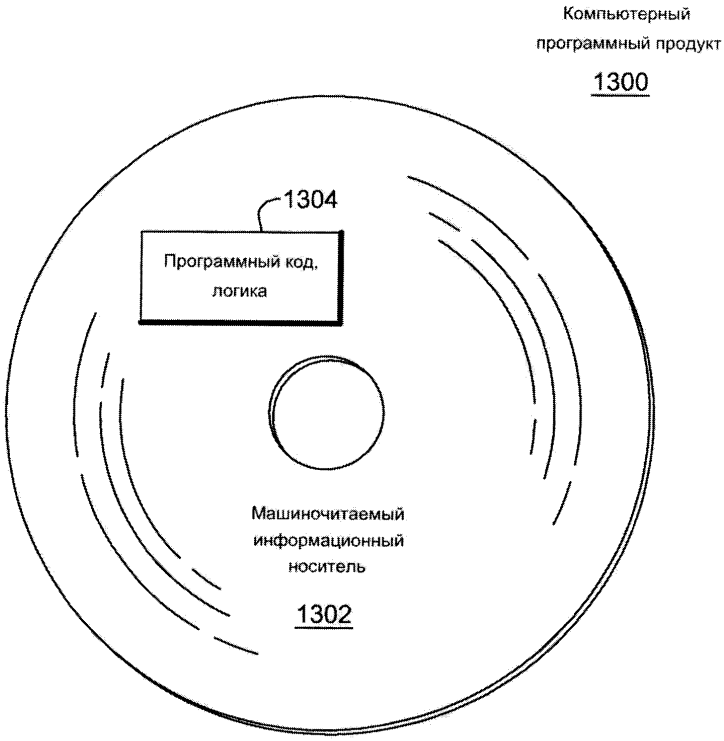

Сброс
Получение операции сброса
1100
Выполнение операции сброса для сброса вычислительного окружения к другому архитектурному режиму (н-р, z/Архитектуры),
Задание архитектурного режима
- использование соответствующего архитектурного формата PSW,
- инвертирование индикатора архитектурного режима (н-р, бита 12) в PSW
1102
END

Фиг. 11

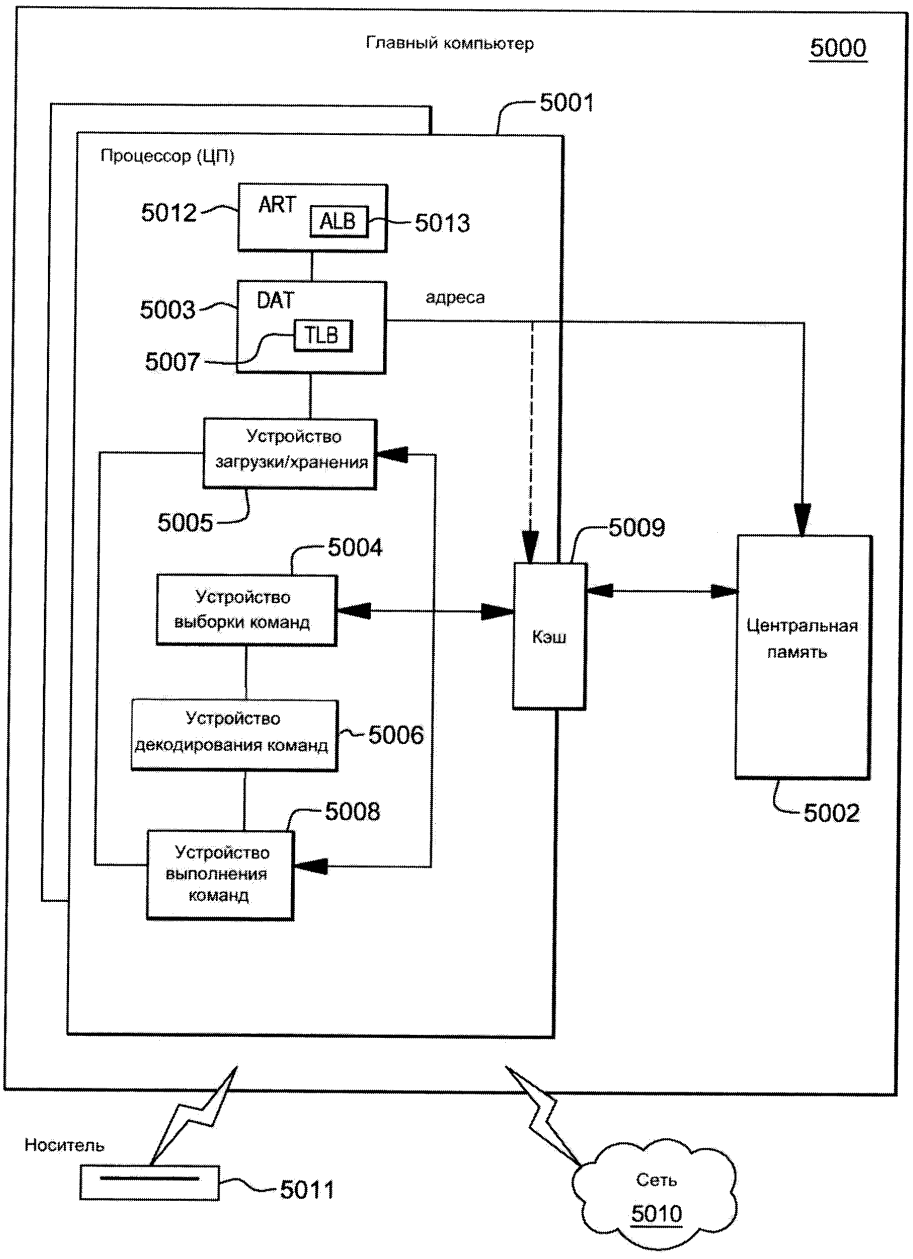

Конфигурирование
1200
Конфигурация вычислительного окружения для выполнения операций в выбранном архитектурном режиме
Начало инициализации вычислительного окружения с использованием сохраненного слова состояния программы, причем сохраненное слово состояния программы имеет формат архитектурного режима отличающегося от выбранного архитектурного режима
1202
Выявление того, что сохраненное слово состояния программы имеет формат архитектурного режима, отличающегося от выбранного архитектурного режима
1204
На основании выявления, что сохраненное слово состояния программы имеет формат архитектурного режима, отличающегося от выбранного архитектурного режима, автоматическое изменение сохраненного слова состояния программы для приобретения формата выбранного архитектурного режима, причем автоматическое изменение выполняется в отсутствие заданного в явном виде запроса на переключение к выбранному архитектурному режиму
1206
Завершение инициализации вычислительного окружения с помощью измененного слова состояния программы для конфигурирования вычислительного окружения в выбранном архитектурном режиме
1208
END

Фиг. 12

16

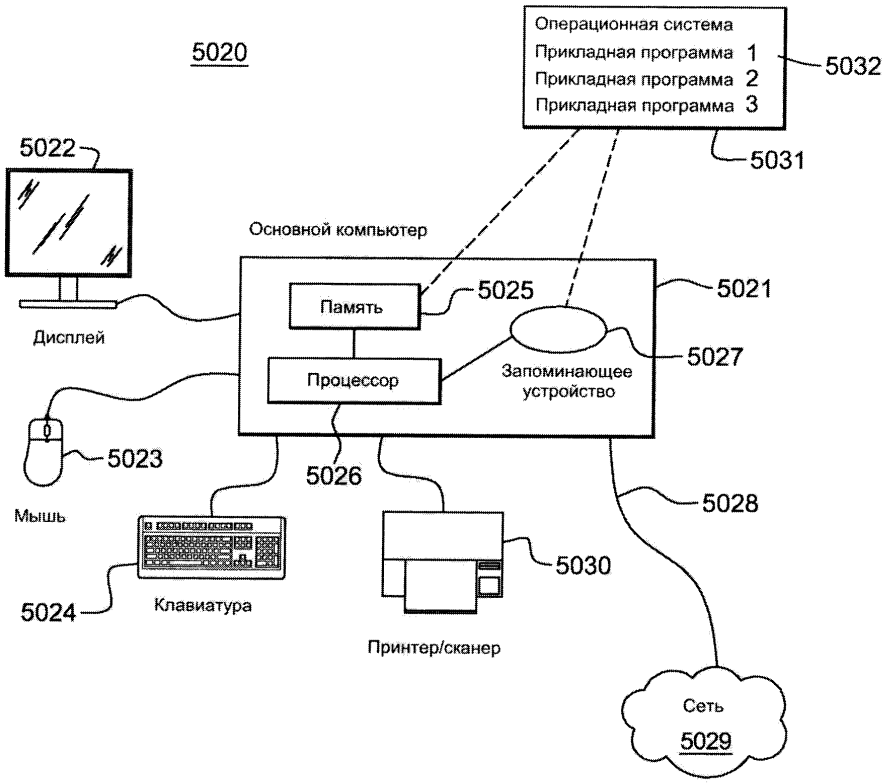

Компьютерный
программный продукт
1300
1304
Программный код,
логика
Машиночитаемый
информационный
носитель
1302

Фиг. 13


Главный компьютер
5000
5001
Процессор (ЦП)
5012
ART
ALB
5013
5003
DAT
TLB
5007
адреса
Устройство загрузки/хранения
5005
5004
Устройство выборки команд
5009
Кэш
Центральная память
5002
Устройство декодирования команд
5006
5008
Устройство выполнения команд
Носитель
5011
Сеть
5010

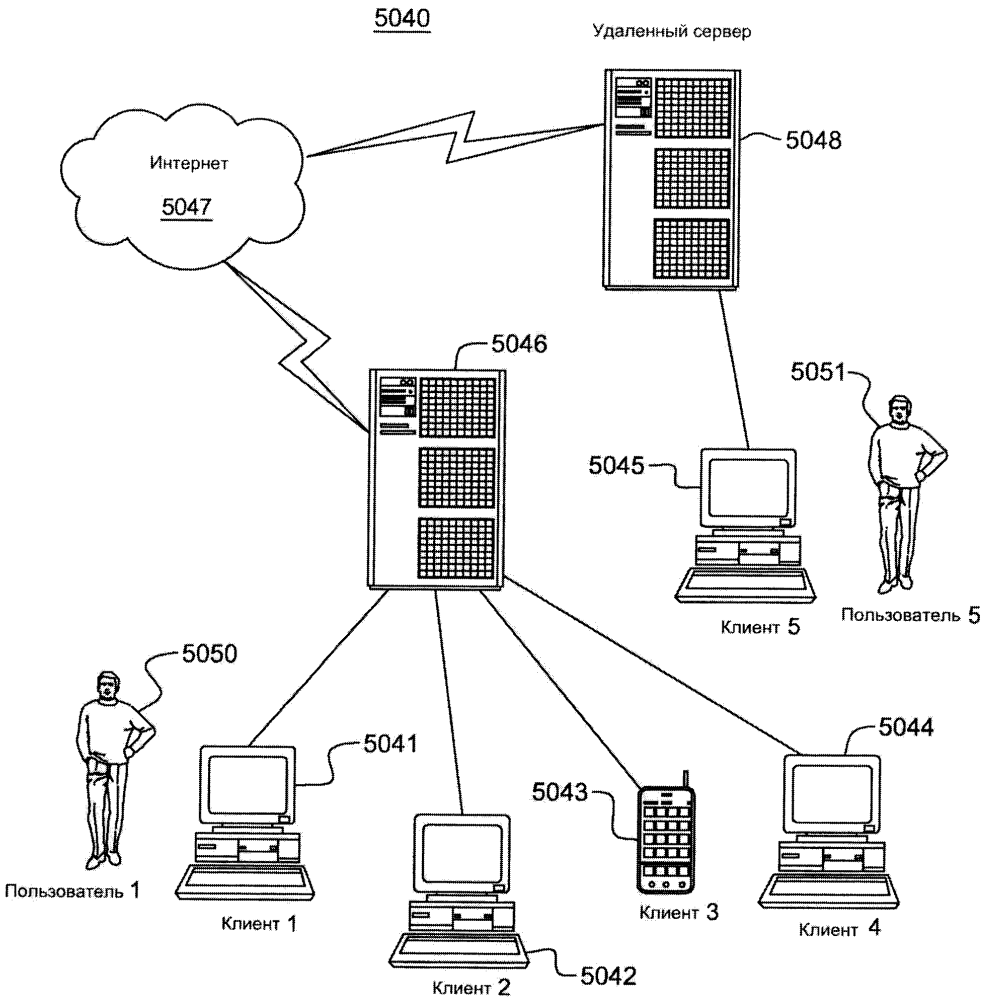
Фиг. 14

18/27

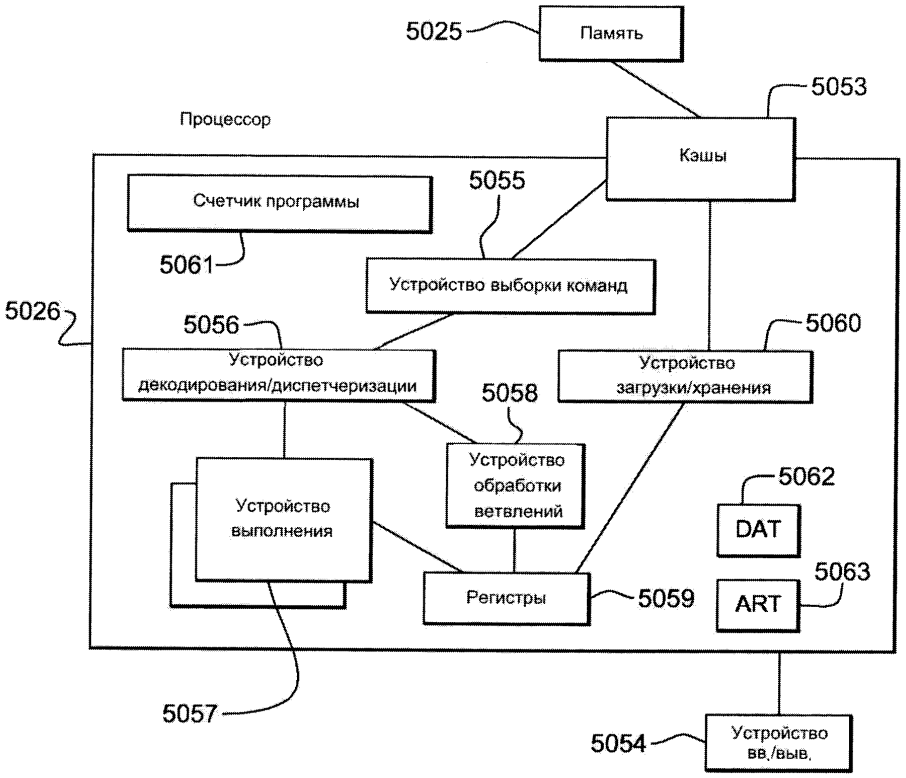


Фиг. 15

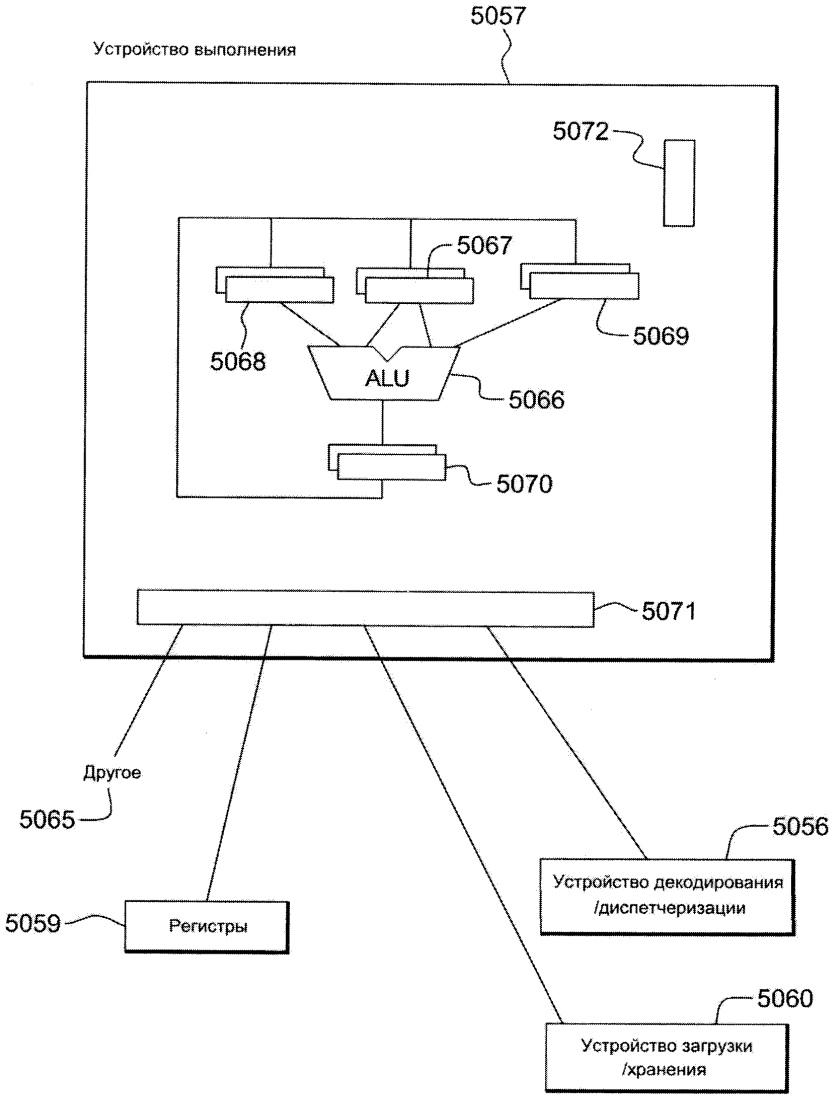
19


5040
Удаленный сервер
Интернет
5047
5048
5046
5051
5045
Клиент 5
Пользователь 5
5050
5041
5044
5043
Пользователь 1
Клиент 1
Клиент 3
Клиент 4
Клиент 2
5042

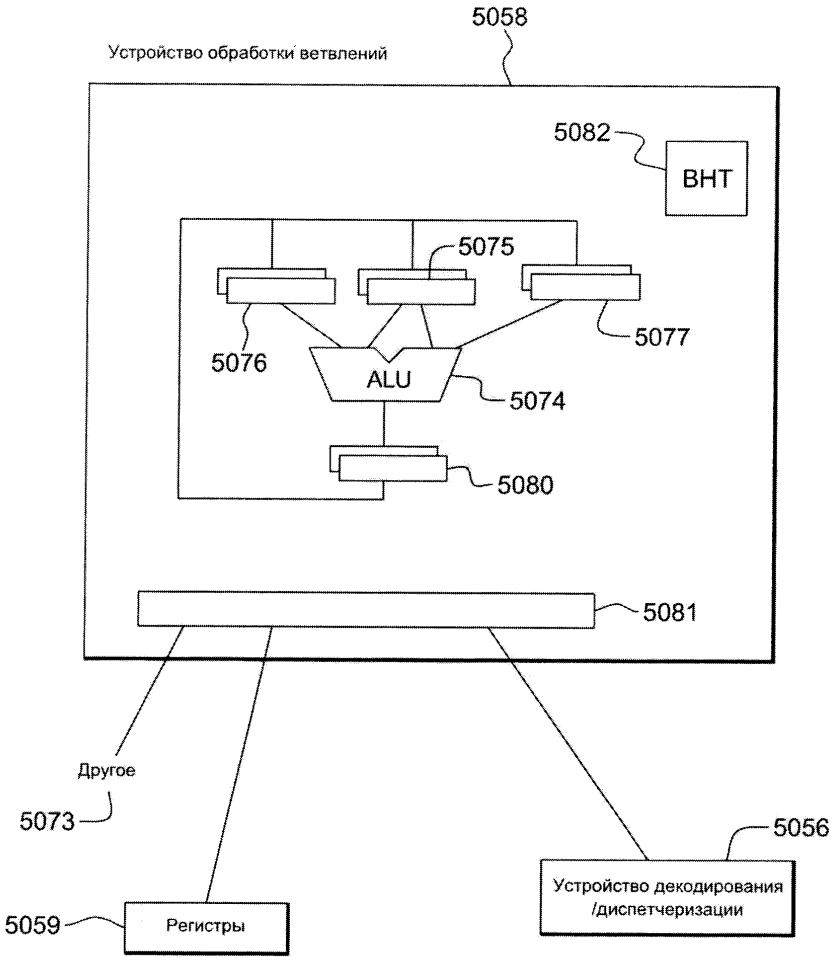
Фиг. 16


5025
Память
5053
Процессор
Кэшы
Счетчик программы
5055
5061
Устройство выборки команд
5026
5056
5060
Устройство декодирования/диспетчеризации
Устройство загрузки/хранения
5058
Устройство обработки ветвлений
5062
Устройство выполнения
DAT
5063
Регистры
5059
ART
5057
5054
Устройство вв./выв.

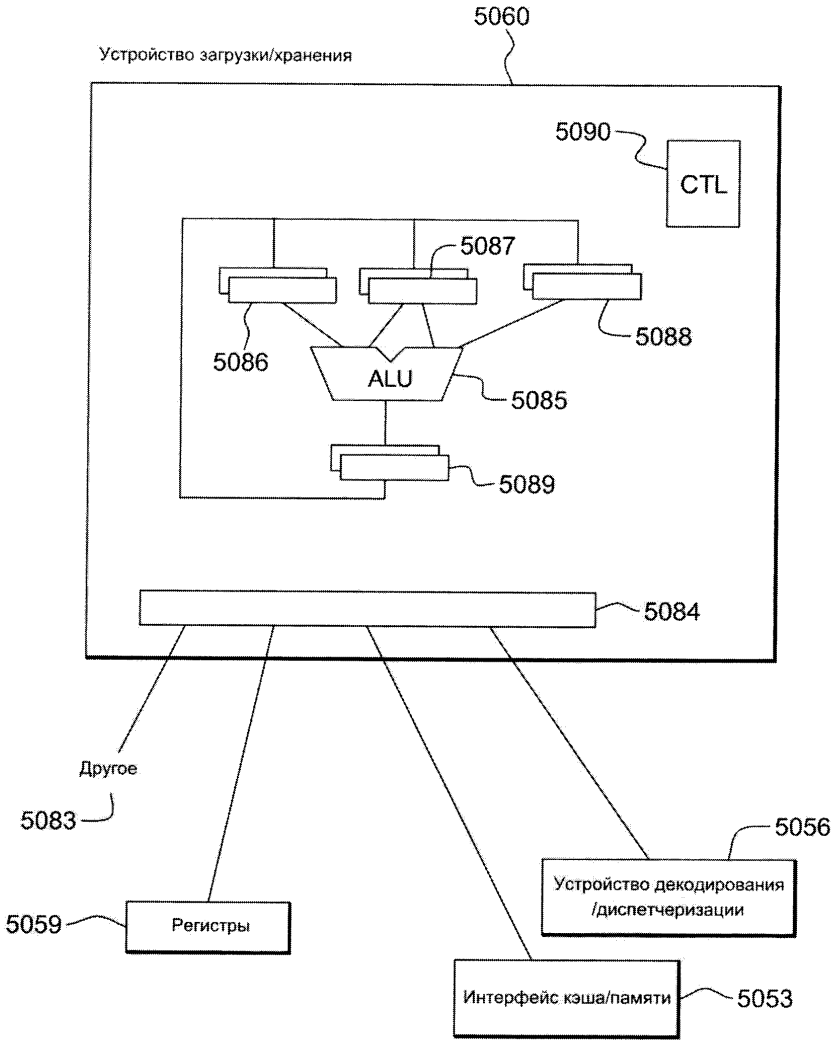
Фиг. 17


5057
Устройство выполнения
5072
5067
5068
5069
ALU
5066
5070
5071
Другое
5065
5056
Устройство декодирования /диспетчеризации
5059
Регистры
5060
Устройство загрузки /хранения

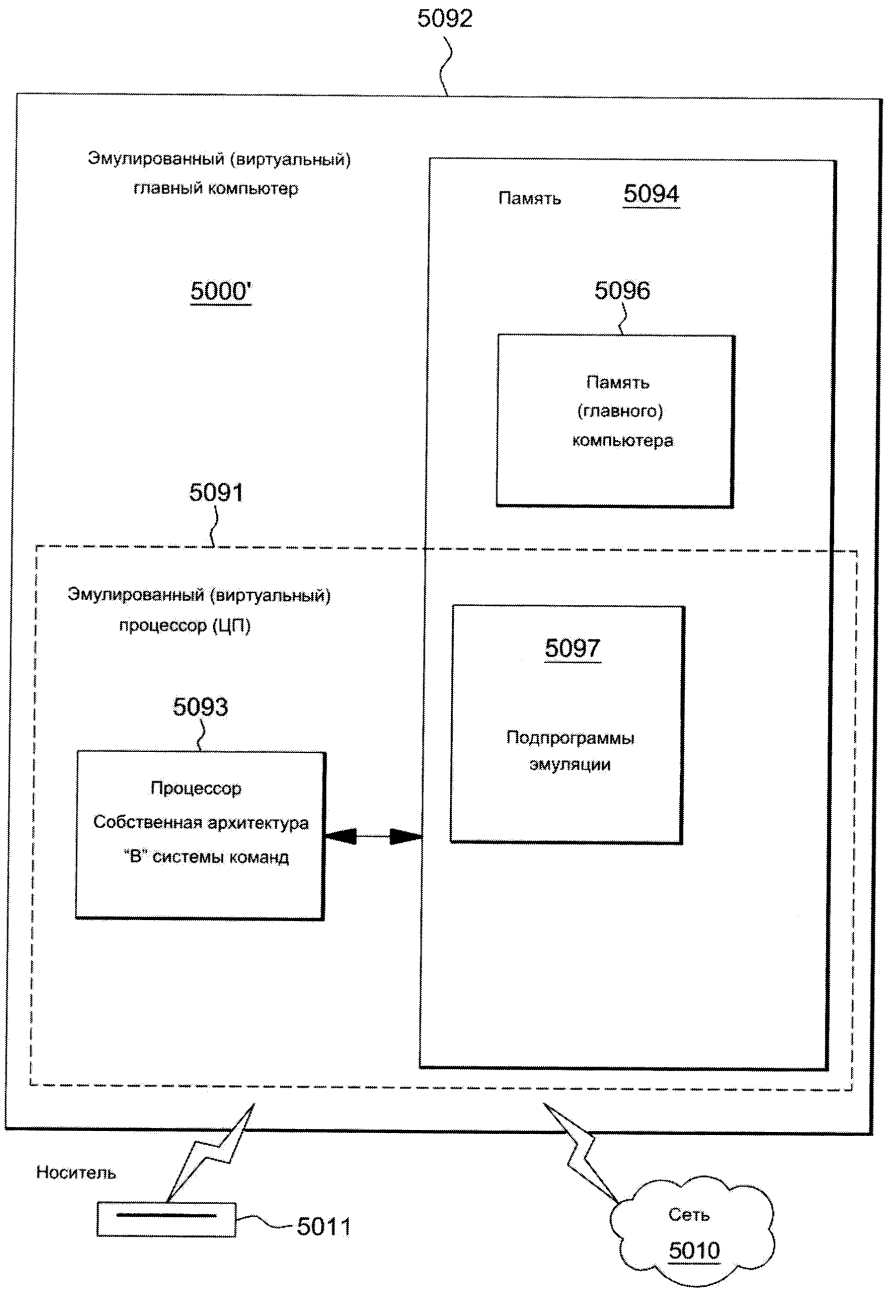
Фиг. 18A


5058
Устройство обработки ветвлений
5082
BHT
5075
5076
5077
ALU
5074
5080
5081
Другое
5073
5056
Устройство декодирования /диспетчеризации
5059
Регистры

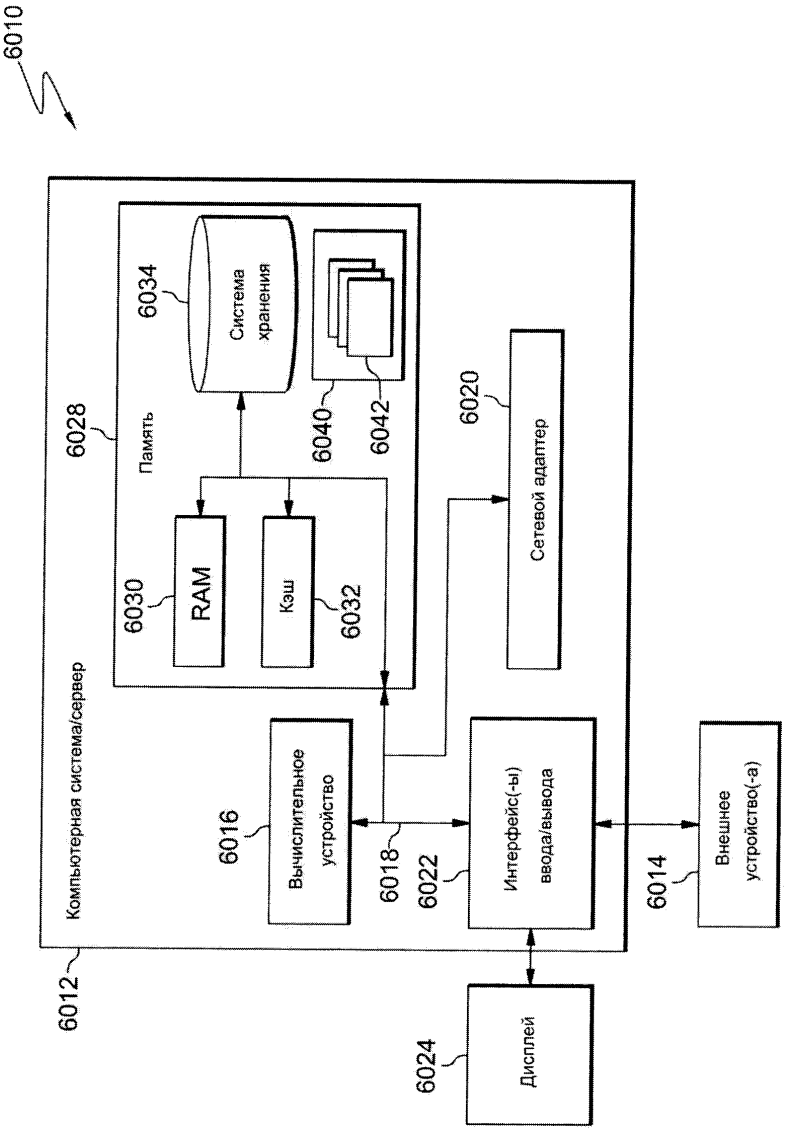
Фиг. 18 Б


5060
Устройство загрузки/хранения
5090
CTL
5087
5086
5088
ALU
5085
5089
5084
Другое
5083
5056
Устройство декодирования /диспетчеризации
5059
Регистры
Интерфейс кэша/памяти
5053

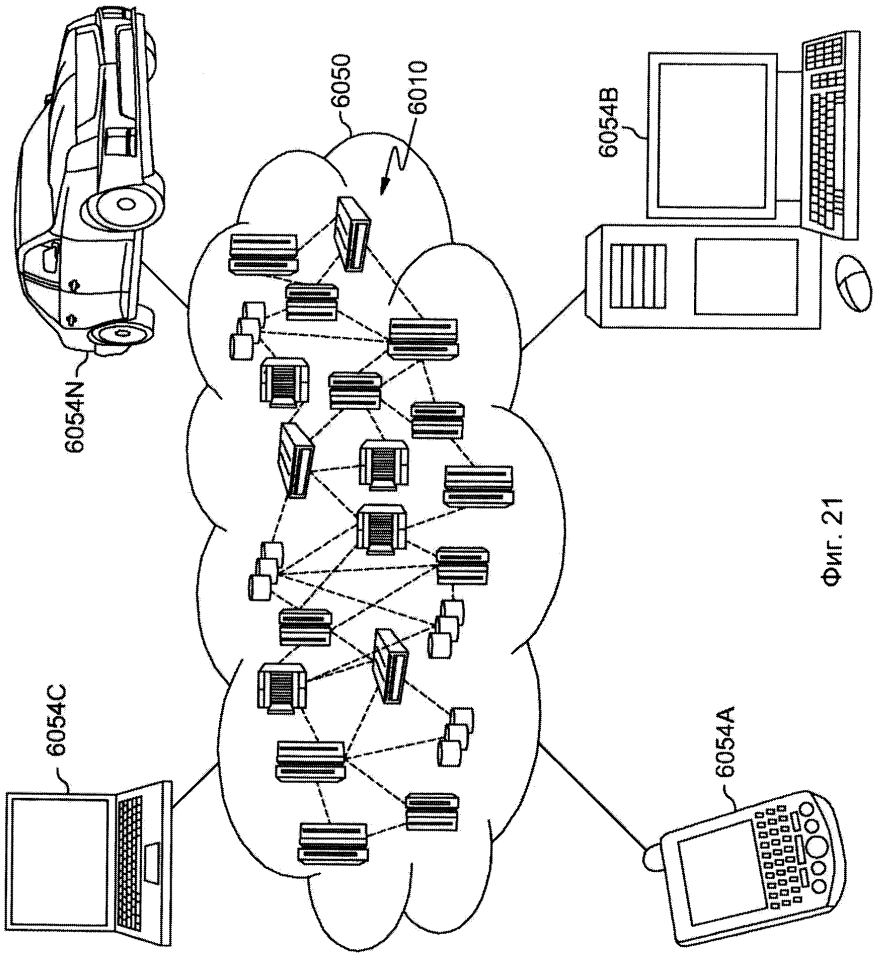
Фиг. 18B


5092
Эмулированный (виртуальный)
главный компьютер
5000'
Память
5094
5096
Память
(главного)
компьютера
5091
Эмулированный (виртуальный)
процессор (ЦП)
5097
Подпрограммы
эмуляции
5093
Процессор
Собственная архитектура
"B" системы команд
Носитель
5011
Сеть
5010

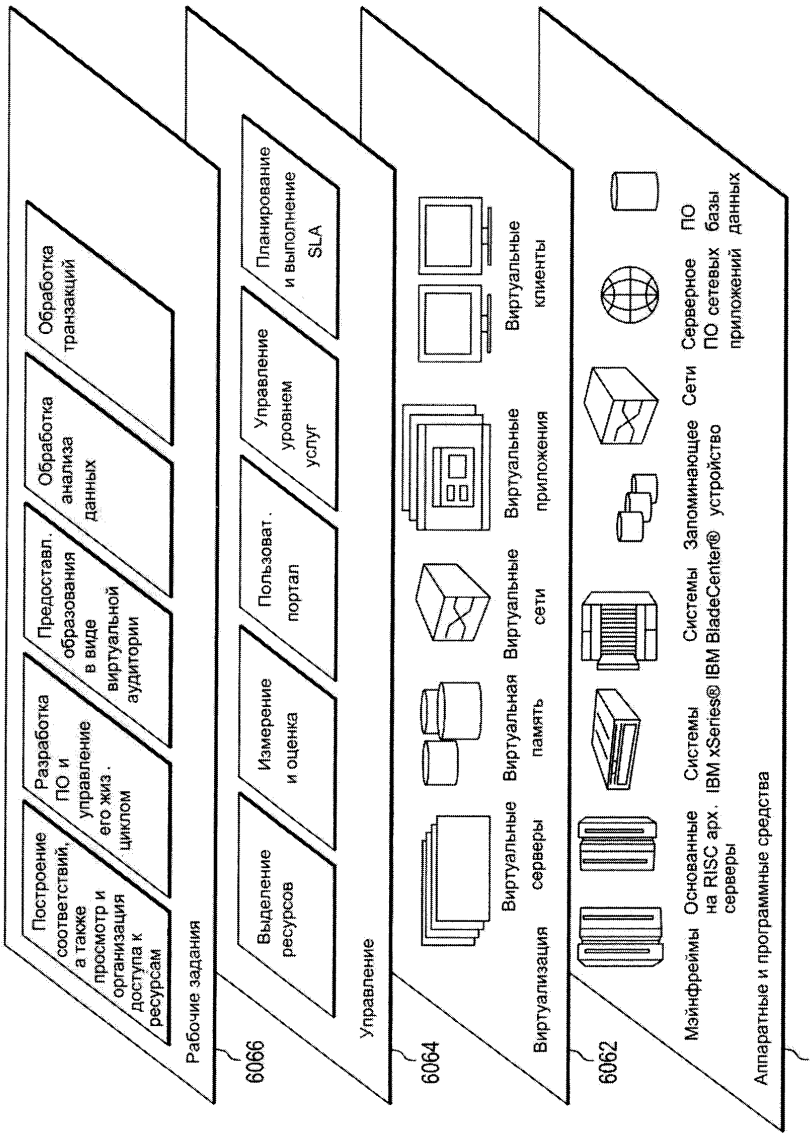
Фиг. 19


6010
Компьютерная система/сервер
6012
Память
6028
Система хранения
6034
6040
6042
RAM
6030
Кэш
6032
Сетевой адаптер
6020
Вычислительное устройство
6016
6018
Интерфейс(-ы) ввода/вывода
6022
Внешнее устройство(-а)
6014
Дисплей
6024

Фиг. 20



Фиг. 21


Рабочие задания
Построение соответствий, а также просмотр и организация доступа к ресурсам
Разработка ПО и управление его жиз. циклом
Предоставл. образования в виде виртуальной аудитории
Обработка анализа данных
Обработка транзакций
6066
Управление
Выделение ресурсов
Измерение и оценка
Пользоват. портал
Управление уровнем услуг
Планирование и выполнение SLA
6064
Виртуализация
Виртуальные серверы
Виртуальная память
Виртуальные сети
Виртуальные приложения
Виртуальные клиенты
6062
Аппаратные и программные средства
Мэйнфреймы
Основанные на RISC арх. серверы
Системы IBM xSeries®
Системы IBM BladeCenter®
Запоминающее устройство
Сети
Серверное ПО
ПО сетевых приложений
ПО базы данных
6060

Фиг. 22