РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 295 756** (13) **C2**

(51) МПК
G06F 15/00 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2004114666/09, 29.10.2002**

(24) Дата начала отсчета срока действия патента:
29.10.2002

(30) Конвенционный приоритет:
16.11.2001 DE 10156394.9

(43) Дата публикации заявки: **27.10.2005**

(45) Опубликовано: **20.03.2007 Бюл. № 8**

(56) Список документов, цитированных в отчете о поиске: **EP 1056012 A2, 29.11.2000. RU 1700562 A1, 23.12.1991. US 5732272 A, 24.03.1998. WO 00/54155 A1, 14.09.2000. RU 2171494 C2, 27.07.2001.**

(85) Дата перевода заявки PCT на национальную фазу:
16.06.2004

(86) Заявка PCT:
EP 02/12074 (29.10.2002)

(87) Публикация PCT:
WO 03/042547 (22.05.2003)

Адрес для переписки:
101990, Москва, Петроверигский пер., 4, "Агентство Ермакова, Столярова и Партнеры", пат.пов. Е.А.Ермаковой

(72) Автор(ы):
ВАЙСС Дитер (DE)

(73) Патентообладатель(и):
ГИЗЕКЕ ЭНД ДЕВРИЕНТ ГмбХ (DE)

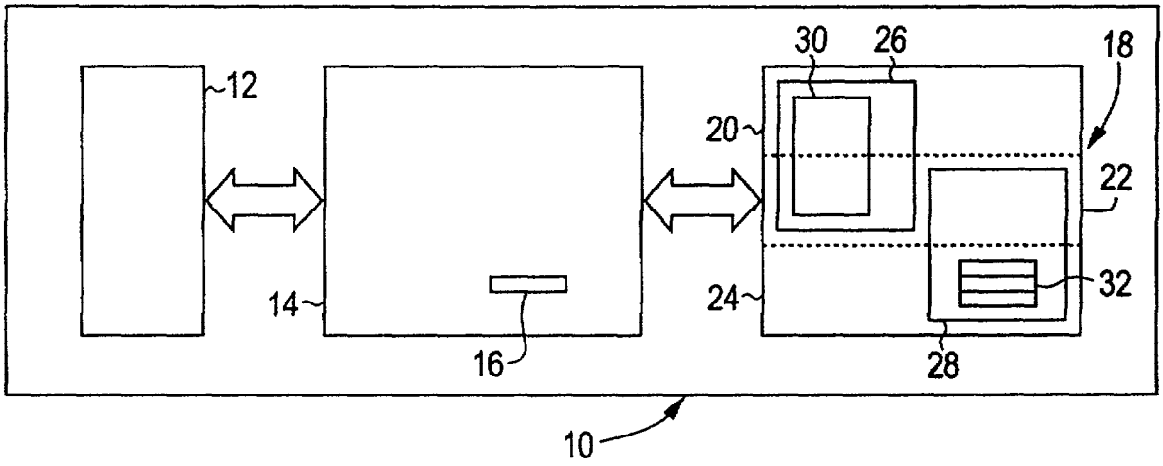
RU 2295756 C2

(54) СПОСОБ ВЫПОЛНЕНИЯ УПРАВЛЯЕМОЙ ПРОГРАММЫ ПОСРЕДСТВОМ ПОРТАТИВНОГО НОСИТЕЛЯ ДАННЫХ

(57) Реферат:

Изобретение относится к области выполнения программ посредством портативного носителя данных. Техническим результатом является повышение защиты выполняемой программы. Способ управляемого выполнения программы посредством портативного носителя данных, содержащего программную память, системное ядро и, по крайней мере, один счетчик состояния, заключается в том, что в ходе функционирования каждого управляемого блока показатель счетчика состояния изменяется с тем, чтобы отразить обработку соответствующих управляемых блоков, во время выполнения команды перехода, в результате которой один или несколько управляемых блоков пропускаются, показатель счетчика состояния изменяется, как если бы пропущенные управляемые блоки были задействованы, показатель счетчика состояния сравнивается, по крайней мере, с одним допустимым значением, по меньшей мере, в одной из точек проверки, в случае соответствия - выполнение программы продолжается, а в случае отличия - происходит обработка ошибок. 2 н. и 18 з.п. ф-лы, 5 ил.

RU 2295756 C2



Фиг. 1

RU 2295756 C2

RU 2295756 C2

RUSSIAN FEDERATION

(19) **RU** (11) **2 295 756** (13) **C2**

(51) Int. Cl.
G06F 15/00 (2006.01)

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(12) ABSTRACT OF INVENTION

(21), (22) Application: **2004114666/09, 29.10.2002**

(24) Effective date for property rights: **29.10.2002**

(30) Priority:
16.11.2001 DE 10156394.9

(43) Application published: **27.10.2005**

(45) Date of publication: **20.03.2007 Bull. 8**

(85) Commencement of national phase: **16.06.2004**

(86) PCT application:
EP 02/12074 (29.10.2002)

(87) PCT publication:
WO 03/042547 (22.05.2003)

Mail address:
**101990, Moskva, Petroverigskij per., 4,
"Agentstvo Ermakova, Stoljarova i Partnery",
pat.pov. E.A.Ermakovoj**

(72) Inventor(s):
VAJSS Diter (DE)

(73) Proprietor(s):
GIZEKE EhND DEVRIENT GmbKh (DE)

(54) METHOD FOR EXECUTION OF CONTROLLED PROGRAM BY MEANS OF PORTABLE DATA CARRIER

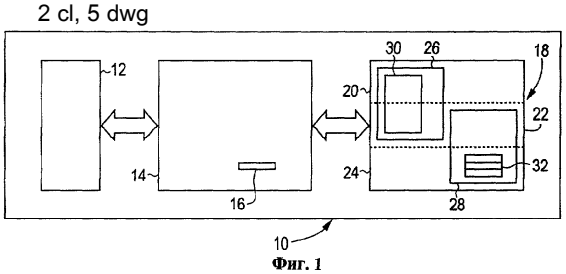
(57) Abstract:

FIELD: technology for executing programs by means of portable data carrier.

SUBSTANCE: in accordance to method for controlled execution of program by means of portable data carrier, containing program memory, system core and at least one condition counter, during functioning of each controlled block coefficient of condition counter changes to indicate processing of appropriate controlled blocks, during realization of transition command, as a result of which one or several controlled blocks are skipped, coefficient of condition counter is changed, as if skipped control blocks have been used, coefficient of condition counter is compared to at least one acceptable value, of at least one check point, in case of match - program execution continues, and in case of mismatch - processing of errors occurs.

EFFECT: increased protection of program being executed.

2 cl, 5 dwg



Фиг. 1

RU 2295756 C2

RU 2295756 C2

Настоящее изобретение в целом относится к технической области выполнения программ посредством портативного носителя данных, содержащего запоминающее устройство для хранения программ, в которой хранится как минимум одна программа, а также содержащего системное ядро, предназначенное для выполнения такой программы. Портативный носитель данных указанного типа может, в частности, быть платой (чип-картой) для установки ИС различных типов или микросхемным модулем (чип-модулем). Данное изобретение в частности относится к управляемому выполнению программы с целью выявления помех или попыток проникновения в защищенную систему и с целью запуска соответствующей обработки ошибок.

Во многих способах проникновения в портативные носители данных обычное выполнение программы затрудняется из-за внешних воздействий. Воздействия такого типа могут быть вызваны, в частности, импульсами напряжения, воздействием теплоты или холода, электрическим или магнитным полями, электромагнитными волнами или корпускулярным излучением. Например, содержимым реестра в системном ядре можно манипулировать посредством вспышек света на незащищенной полупроводниковой микросхеме (чипе) портативного носителя данных. Программный счетчик может быть потенциально изменен при этом способе проникновения с помощью света таким образом, что портативный носитель информации в результате начинает упускать конфиденциальную информацию из входного/выходного буферного запоминающего устройства, несмотря на запрограммированные данные, хранящиеся на нем.

Из европейской патентной заявки EP 1056012 A2 известно устройство, контролирующее механизм дверного замка в автомобиле, в котором управляющий процессор выполняет множество дополнительных программ поочередно, а число на счетчике увеличивается после выполнения каждой такой программы. После выполнения всех дополнительных программ производится проверка на предмет того, соответствуют ли показания счетчика числу дополнительных программ.

Тем не менее, это известное устройство имеет отношение исключительно к системе, постоянно установленной в автомобиле. Более широкое использование изобретения, в частности, в портативном носителе данных, в заявке не предусматривается. Кроме того, в таком устройстве используется строго поочередная последовательность запуска всех дополнительных программ. Таким образом, изобретение, известное из европейской патентной заявки EP 1056012 A2, не подходит для более сложных приложений, в которых отдельные дополнительные программы могут по выбору пропускаться.

Таким образом, целью данного изобретения является, по меньшей мере, частичное устранение проблем, свойственных предшествующему уровню техники, и создание способа управляемого выполнения программ посредством портативного носителя данных, а также портативного носителя данных, обеспечивающих надежную защиту от манипуляций важной информацией на программном счетчике и способных к применению для выполнения сложных программ. В предпочтительных вариантах изобретения, в частности, необходимо достигнуть высокой степени защиты при минимальных затратах.

В соответствии с изобретением данная цель полностью или частично достигается при помощи способа с признаками, отраженными в пункте 1 формулы изобретения, а также посредством описания портативного носителя данных, имеющего признаки, указанные в пункте 10 формулы изобретения. Зависимые пункты формулы относятся к предпочтительным вариантам воплощения изобретения.

Изобретение берет свое начало в базовой концепции управления правильным выполнением программы посредством счетчика состояния, который загружается во время выполнения программы. Программа имеет множество управляемых блоков, обработка которых ведет в каждом случае к изменениям в счетчике состояния. Управляемые блоки не обязательно совпадают с дополнительными программами или программными модулями. Показания счетчика состояния сравниваются с одним или несколькими допустимыми значениями, как минимум, в одной точке проверки программы, и в случае, если выявляется разница, происходит обработка соответствующей ошибки.

Согласно изобретению обеспечивается, по меньшей мере, возможность поддержки программы, содержащей, по крайней мере, одну команду перехода, в результате которой один или несколько управляемых блоков пропускаются и, таким образом, не задействуются. В этом случае показатель счетчика состояния изменяется в соответствии с выполнением команды перехода, как если бы пропущенные управляемые блоки были бы задействованы. Значение счетчика состояния затем может быть проверено в последующей временной точке, независимо от выполнения команды перехода. Гибкость в конфигурации выполняемой программы может, таким образом, существенно увеличиться без ограничений применительно к уровню безопасности, который необходимо обеспечить.

Счетчик состояния можно настроить как реестр в системном ядре или как раздел в рабочей памяти (ОЗУ) портативного носителя данных. В предпочтительных вариантах изобретения счетчик состояния настраивается как дифференциальный счетчик состояния, значение в котором выводится из разницы между двумя счетчиками, внедренными на уровне аппаратного оборудования, известными как "базовый счетчик" и "активный счетчик". Счетчик состояния, таким образом, защищается от постоянного допуска одной и той же последовательности значений по ходу выполнения программы. Использование дифференциального счетчика состояния, таким образом, усложняет возможность стороннего выслеживания используемого способа и увеличивает его безопасность. Базовый счетчик предпочтительно регулярно устанавливать в исходное положение, в котором он настроен на соответствующий показатель активного счетчика.

В принципе, структура выполняемой программы ничем не ограничена. В предпочтительных конфигурациях, которые предусмотрены, в частности, для плат (чип-карт) и микросхемных модулей (чип-модулей), основной структурной единицей программы является цикл, который, однако, включает в себя блок запуска, множество управляемых блоков и блок обработки событий.

Желательно, чтобы блок запуска инициализировал считывание внешней команды, управляемые блоки соотносились с обработкой этой команды, а ответ выдавался в блоке обработки событий. В этом случае предпочтительно, чтобы точка проверки или, если их множество, - одна из нескольких точек проверки находилась бы между последним управляемым блоком и блоком обработки событий, чтобы ответ выдавался только тогда, когда проверка счетчика состояния не привела бы к обнаружению вмешательств в выполнение программы. Общая характеристика предпочтительных вариантов воплощения изобретения, независимо от вышеописанной структуры программы, заключается в обеспечении точки проверки перед каждым событием в портативном носителе данных, ориентированным извне.

Программа преимущественно состоит из одного или нескольких модулей, выполняемых один за другим, которые, в свою очередь, содержат, по крайней мере, один управляемый блок. В некоторых вариантах каждый модуль состоит как раз из одного управляемого блока и в качестве дополнения, содержит тестирующий блок, обеспечивающий проверку на предмет того, следует выполнить или пропустить последующие модули. В других вариантах, по меньшей мере, некоторые из модулей содержат множество управляемых блоков.

Дальнейшее увеличение степени отслеживания достигается при условии, если, по меньшей мере, один из модулей настроен как управляемый модуль, по крайней мере, с одной встроенной в него точкой проверки. Значение счетчика состояния с начала выполнения программы, либо с начала выполнения текущего цикла может проверяться на соответствие с одной или несколькими предопределенными значениями в этой точке проверки, встроенной в модуль, либо может выполняться проверка изменения в счетчике состояния с начала выполнения модуля. Во втором случае, показатель счетчика состояния или значение, связанное с ним, например, состояние активного счетчика, желательно сохранить в памяти локального базового счетчика перед первым изменением в счетчике состояния в этом модуле.

Локальный счетчик может быть аналогично введен в дальнейших конфигурациях внутри

модуля или внутри блока и отслеживает правильное выполнение программы внутри модуля или внутри блока независимо от глобального счетчика состояния. Локальный счетчик может быть настроен в соответствии со способами, описанными выше, совместно с глобальным счетчиком состояния, возможна, в частности, настройка в качестве дифференциального локального счетчика.

Каждая команда перехода, посредством которой управляемые блоки могут быть пропущены, в предпочтительном варианте объединяется с соответствующими командами для адаптации счетчика состояния. Однако согласно настоящему изобретению могут также предусматриваться конфигурации, в которых команды индивидуального перехода не сопровождаются адаптацией счетчика состояния. В этом случае множество допустимых состояний счетчика обычно определяются в точке или в точках проверки. Поскольку в такой ситуации точность отслеживания уменьшается, желательно, подобную структуру желательно использовать только применительно к упомянутому локальному счетчику состояния.

В соответствии с настоящим изобретением портативный носитель данных предпочтительно представляет собой плату (чип-карту) или микросхемный модуль (чип-модуль). В предпочтительных конфигурациях это устройство характеризуется признаками, соответствующими тем, которые были описаны ранее и/или раскрытыми в зависимых пунктах формулы изобретения, которые относятся к способу.

Иные признаки, преимущества и цели изобретения станут очевидными из последующего подробного описания множества вариантов и альтернативных вариантов воплощения изобретения

Далее приводится ссылка на чертежи-схемы, где

Фиг.1: Блок-схема, отражающая функциональные единицы портативного носителя данных в соответствии с одним из вариантов воплощения изобретения

Фиг 2: График последовательности операций выполнения программы в варианте, представленном на Фиг.1

Фиг.3: График последовательности операций в разновидности конфигурации, представленной на Фиг.2

Фиг 4: График последовательности операций выполнения программы в управляемом модуле в соответствии с иной разновидностью конфигурации согласно Фиг.2

Фиг.5: График последовательности операций выполнения программы в блоке, состоящем из множества управляемых подблоков.

Портативный носитель данных 10, изображенный на Фиг.1, сконструирован в данном варианте как плата (чип-карта). Он содержит интерфейс 12 для бесконтактного или контактного обмена данными со средой, системное ядро 14 с программным счетчиком 16 и запоминающим устройством 18. Системное ядро 14, запоминающее устройство 18 и элементы интерфейса 12 обычно интегрируются в одной полупроводниковой микросхеме (чипе)

Запоминающее устройство 18 имеет множество зон, настроенных по различным методам использования памяти. В варианте, показанном на Фиг.1, это запоминающее устройство только для чтения 20, настроенное, например, как запрограммированное фотошаблонами ПЗУ, долговременное двусторонное запоминающее устройство 22, настроенное по технологии FLASH, а также долговременное двустороннее запоминающее устройство 24, настроенное как ОЗУ. В концептуальном плане запоминающее устройство 18 содержит программную память 26, расположенную в зонах 20 и 22, и рабочую память 28 в зонах 22 и 24.

Программная память 26 содержит множество программ, одна из которых проиллюстрирована на Фиг.1 и снабжена номером для ссылок 30. Места в памяти для множества счетчиков 32 зарезервированы в рабочей памяти 28, а точнее, как показаны в варианте на Фиг.2 для счетчика состояния ZZ, в варианте на Фиг.3 - для активного счетчика AZ и базового счетчика BZ, в модификации на Фиг.4 - для счетчика состояния ZZ и локального базового счетчика LBZ и в модификации на Фиг.5 - для счетчика

состояния ZZ и локального счетчика LZ. В альтернативных конфигурациях эти счетчики, все или частично, могут быть также настроены как реестры системного ядра 14 вместо их расположения в рабочей памяти 28, либо могут храниться в прочих узлах портативного носителя данных 10, не показанных на Фиг.1.

В настоящем варианте конфигурации на программный счетчик 16 системного ядра 14 можно повлиять вспышками света, направленными на полупроводниковую микросхему (чип). Для того чтобы все-таки воспрепятствовать потенциальному вторжению в данные как следствию подобной атаки, управление которыми ведется в системном ядре, счетчик состояния ZZ при выполнении программы в соответствии с Фиг.2 загружается во время выполнения программы как механизм управления, не зависящий от программного счетчика 16, и в каждом случае выполняет проверку на связанность данных с предопределенными показаниями счетчика состояния, прежде чем выполнить действие, направленное извне.

В варианте на Фиг.2 программа 30 обладает структурой бесконечного цикла, включающего в себя блок запуска 34, множество управляемых блоков 36.1, 362,...36.N, далее совместно обозначаемые как 36.x, точки проверки 38 и блок обработки событий 40. Команда, подлежащая обработке портативным носителем данных 10, может представлять собой, например, считывание во внутреннюю память через интерфейс 12 в блоке запуска 34. Управляемые блоки 36.x соотносятся с этапами обработки выполняемой команды поочередно, при этом, например, команда сначала проверяется на истинность, затем происходит проверка на авторизацию, затем - проверка выполнимости команды, например, посредством проверки на наличие запрошенных данных, и, наконец, команда выполняется таким образом, например, что запрошенные данные записываются в выходной буфер. В настоящем варианте ответ передается через интерфейс 12 в блоке обработки событий 40 и, в результате, выдается сообщение об успешном выполнении команды либо невозможности ее выполнения.

В функциях отдельно управляемых блоков 36.x, перечисленных в качестве примера выше, могут появляться ошибки, что делает выполнение последующих управляемых блоков 36.x при выполнении программы излишними. Если, например, команда не должна выполняться по причине отсутствия авторизации, последующие шаги обработки являются избыточными. Для этого существуют тесты на выход 42.1, 42.2 и т.д., вместе обозначаемые далее как 42.x, в рамках которых осуществляется проверка на предмет того, следует ли задействовать или пропустить управляемые блоки 36.x, следующие в предписанной последовательности. Каждый тест на выход 42.x является командой об обусловленном переходе, которая проверяет значение признака выхода. Признак выхода, в свою очередь, задается соответствующим предыдущим управляемым блоком 36.x, указывая на то, что необходимо произвести выход. Несомненно, проверку на выход 42.x необходимо ассоциировать только с управляемыми блоками 36.x, когда значение признака выхода может измениться в этих управляемых блоках 36.x.

Чтобы проверить выполнение программы независимо от состояния программного счетчика 16 и, таким образом, получить возможность распознавать вмешательство посредством светового воздействия, используется счетчик состояния ZZ. Счетчик состояния ZZ установлен на исходное значение посредством команды об инициализации 44 в сочетании с выполнением блока запуска 34. В варианте на Фиг.2 это значение равно нулю. Каждый из управляемых блоков 36.x также содержит команду 46.x, по которой показатель счетчика состояния ZZ изменяется, а именно, его значение увеличивается на одну единицу в настоящем варианте. Затем команда 46.x исполняется в рамках каждого управляемого блока 36.x только тогда, когда операции, подлежащие выполнению соответствующим блоком 36.x, закончили выполняться. Команда 46.x в предпочтительном варианте является последней командой в каждом управляемом блоке 36.x в каждом случае.

В целом, таким образом, счетчик состояния ZZ, если предположить, что не выполняются никакие переходы, содержит значение, которое в процессе выполнения программы соответствует числу выполняемых управляемых блоков 36.x. В последовательности на

Фиг.2, приведенной в качестве примера, это число обозначается значением N. Проверка 48, таким образом, происходит в точке проверки 38, в которой определяется, имеет ли показатель счетчика состояния ZZ единственное допустимое значение N. В этом случае задействуется блок обработки событий 40. Если счетчик состояния ZZ имеет другое значение, выполнение программы переходит к процедуре обработки ошибок 50.

Полная перезагрузка портативного носителя данных 10 или даже, к примеру, блокирование портативного носителя данных 10 при повторяющейся ошибке, может произойти в результате обработки ошибок 50. Соответствующее сообщение об ошибке может быть также отправлено посредством интерфейса 12. В любом случае, обработка ошибок 50 должна гарантировать, что никакая информация на портативном носителе данных 10 не подвергается потенциальной угрозе выхода наружу через интерфейс 12.

В соответствии с ранее описанной последовательностью, счетчик состояния ZZ не будет иметь допустимое значение N, если переход через один или несколько управляемых блоков 36.6 был вызван одной из проверок на выход 42.x. Чтобы обеспечить правильную работу программы и для таких случаев, соответствующие команды о корректировке 52.1, 52.2, далее вместе обозначаемые как 52.x, выполняются каждый раз, когда управляемые блоки 36.x пропускаются в результате проверки на выход 42.x. Показатель счетчика состояния ZZ изменяется при любой команде о корректировке 52.x, как если бы пропущенные управляемые блоки 36.x были бы задействованы. В настоящем случае показатель счетчика состояния ZZ увеличивается после каждой команды о корректировке 52.x на число пропущенных блоков 36.x. Счетчик состояния ZZ, таким образом, имеет единое допустимое значение N при достижении точки проверки 38, даже если выход произошел во время выполнения цикла.

В описанном варианте показатель счетчика состояния ZZ устанавливают на нулевое значение и увеличивают на одну единицу во время обработки каждого управляемого блока. В других конфигурациях используют другие начальные значения и/или другие способы изменения счетчика состояния. Например, счетчик состояния ZZ может быть изначально установлен командой 44 на значение 1, или N, или другое значение. Также возможно, что в случае, если после проверки на выход 42.x управляемые блоки 36.x пропускаются, счетчик состояния ZZ продолжает счет без корректировок, и вместо этого уменьшает значение N на число пропущенных блоков 36.x. В таком случае, команды о корректировке 52.1, 52.2 будут иметь следующий вид N=1 и N=2, и так далее. В командах 46.x могут выполняться любые математические операции, например, вычитание единицы или поразрядный сдвиг или циклический сдвиг. Допустимое значение при проверке 48, несомненно, должно настраиваться в соответствии с этими способами.

На Фиг.3 показана следующая разновидность последовательности, изображенной на Фиг.2. Счетчик состояния ZZ не встроен здесь прямо в качестве счетчика 32 в рабочей памяти 28, а имеются два счетчика 32 в рабочей памяти 28, а именно активный счетчик AZ и базовый счетчик BZ. Такая конфигурация счетчика состояния ZZ в виде дифференциального счетчика позволяет избегать ситуаций, когда каждое состояние программного счетчика 16 соответствует фиксированным и всегда идентичным значениям счетчиков 32 в рабочей памяти 28.

Последовательность, изображенная на Фиг.3, начинается с этапа инициализации 54, на котором активный счетчик AZ установлен на начальное значение, в настоящем варианте значение равно нулю. После этапа инициализации 54 структура программы, изображенная на Фиг.3, аналогична структуре на Фиг.2, по этой причине одни и те же номера позиций используются на Фиг.3 и 2. Различия состоят в том, что базовый счетчик BZ установлен на соответствующее состояние активного счетчика AZ командой 44. Состояние базового счетчика BZ, таким образом, является в любом цикле "нулевой отметкой", от которой активный счетчик AZ начинает отсчет.

Согласно Фиг.3 активный счетчик AZ изменяется командами 46.x и 52.x таким же образом, как счетчик состояния ZZ на Фиг.2. И, наконец, разница между активным счетчиком AZ и базовым счетчиком BZ сравнивается при проверке 48 с допустимым

значением N. Сравнения с постоянным значением N предпочтительно избегаются при фактическом программировании проверки 48, когда, например, изначально рассчитывается промежуточный результат BZ+N и затем этот промежуточный результат сравнивается со значением активного счетчика AZ.

Показатель активного счетчика AZ не сбрасывается после каждого выполнения цикла, вместо этого "нулевая отметка", определяемая базовым счетчиком BZ, просто снова задается командой 44. Значения счетчиков AZ и BZ, таким образом, изменяются после каждого выполнения цикла, так что очень сложным становится выследить метод, представленный на Фиг.3. В разновидностях способа, представленного на Фиг.3, значение активного счетчика AZ может произвольно сбрасываться после проверки 48. Функции счетчиков 32 в рабочей памяти 28, которые соответственно являются активным счетчиком AZ и базовым счетчиком BZ, могут также время от времени изменяться.

На Фиг.2 и 3 в качестве примера показан модуль 56, который включает в себя два управляемых блока 36.1 и 36.2 и два локализованных теста на переход 42.1 и 42.2. На Фиг.4 показан этот модуль 56, изображенный на Фиг.2, в последующей разновидности, содержащей дополнительную локальную проверку на изменения в счетчике состояния ZZ во время функционирования модуля 56. Согласно Фиг.4 показатель счетчика состояния ZZ сохраняется для этой цели в начале выполнения модуля в локальном базовом счетчике LBZ.

После обработки каждого управляемого блока 36.1 и 36.2 и результатов соответствующих тестов на переход 42.1 и 42.2, выполнение программы доходит до внутримодульной точки проверки 60. Здесь посредством теста 62 выполняется проверка на предмет того, соответствует ли изменение в счетчике состояния ZZ по отношению к значению локального базового счетчика LBZ, сохраненному в начале выполнения модуля, - числу M управляемых блоков 36.x, задействованных за это время. В настоящем примере число M имеет значение 2. Если внутримодульное изменение в счетчике состояния ZZ имеет допустимое значение, выполнение программы продолжается, в противном случае выполняется внутримодульная процедура по исправлению ошибок 64.

Разновидность конфигурации на Фиг.4 может также использоваться в способе в соответствии с Фиг.3 применительно к дифференциальному счетчику состояния. В этом случае либо значение активного счетчика AZ, либо разница AZ-BZ сохраняется на этапе 58 в локальном базовом счетчике LBZ, а при тесте 62 рассчитывается разница между этим значением или этой разницей и значением локального базового счетчика LBZ.

В то время как в ходе проверки внутримодульного изменения согласно Фиг.4 глобальный счетчик состояния ZZ увеличивает значение, также возможно ввести следующий локальный счетчик состояния в модуле 56 или внутри управляемых блоков 36.x. На Фиг.5 в качестве примера показана последовательность такой разновидности конфигурации, в которой управляемые блоки 36.x, как показаны на Фиг.2, еще один раз защищаются локальным счетчиком состояния LZ.

Согласно Фиг.5 локальный счетчик состояния LZ настраивается на начальное нулевое значение на этапе 66. Начинается функционирование нескольких, в данном примере - трех управляемых подблоков 68.1, 68.2, 68.3, совместно обозначаемых как 68.x. Локальный счетчик LZ приращивается в каждом из управляемых подблоков 68.x за счет соответствующей команды 70.x. Второй управляемый подблок 68.2 содержит обусловленный выход, изображенный на Фиг.5 пунктирной стрелочкой, посредством которого этап приращения 70.2 и третий подблок 68.3 пропускаются.

После обработки последнего управляемого подблока 68.3 выполнение программы достигает точки локальной проверки 72. В целях проверки выполняется тест 74, при котором статус локального счетчика LZ сравнивается с допустимыми значениями 1 и 3. Если существует соответствие с одним из этих значений, выполнение программы продолжается командой 46.x об изменении глобального счетчика состояния ZZ, в противном случае происходит обработка ошибок 76.

Разновидности, уже описанные выше в связи со счетчиком ZZ, также могут

использоваться применительно к локальному счетчику LZ, показанному на Фиг.5. В частности, локальный счетчик LZ может также быть настроен как дифференциальный счетчик. Если способ, отображенный на Фиг.5, используется в совокупности с конфигурацией, показанной на Фиг.3, то необходимо настроить только локализацию 46.x.

На Фиг.5 показано сравнение счетчика с множеством допустимых значений, что необходимо, когда имеются различные цепи выполнения программ без соответствующих команд о корректировке 52.x. В некоторых вариантах изобретения может также использоваться сравнение такого типа с множеством допустимых значений для глобального счетчика состояния ZZ. Однако такая конфигурация снижает надежность управления и, таким образом, предпочтительна только для локальных счетчиков состояния LZ. И наоборот, выход во втором управляемом подблоке 68.2 может также выполняться в других вариантах через соответствующую команду о корректировке. В этом случае только одно состояние локального счетчика LZ, а именно состояние счетчика 3, допустимо в проверке 74.

Формула изобретения

1. Способ управляемого выполнения программы посредством портативного носителя данных (10), содержащего программную память (26), системное ядро (14) и, по крайней мере, один счетчик состояния (ZZ), отличающийся тем, что программная память (26) содержит, по крайней мере, одну программу (30), выполняемую посредством системного ядра (14) и множества управляемых блоков (36.x), а также тем, что в ходе функционирования каждого управляемого блока (36.x) показатель счетчика состояния (ZZ) изменяется с тем, чтобы отразить обработку соответствующих управляемых блоков (36.x), во время выполнения команды перехода (42.x), в результате которой один или несколько управляемых блоков (36.x) пропускаются, показатель счетчика состояния (ZZ) изменяется, как если бы пропущенные управляемые блоки (36.x) были задействованы, показатель счетчика состояния (ZZ) сравнивается, по крайней мере, с одним допустимым значением, по меньшей мере, в одной из точек проверки (38, 60, 72), в результате чего в случае соответствия выполнение программы продолжается, а в случае отличия происходит обработка ошибок (50, 64, 76).

2. Способ по п.1, отличающийся тем, что счетчик состояния (ZZ) настраивается как дифференциальный счетчик состояния, значение которого соответствует разнице между значениями активного счетчика (AZ) и базового счетчика (BZ).

3. Способ по п.1 или 2, отличающийся тем, что программа (30) имеет конфигурацию цикла, включающего блок запуска (34), множество управляемых блоков (36.x) и блок обработки событий (40), точки проверки (38) или одной из точек проверки, локализованных между последним управляемым блоком (36 N) и блоком обработки событий (40).

4. Способ по п.2, отличающийся тем, что базовый счетчик (BZ) настраивается на соответствующее значение активного счетчика (AZ) перед задействованием управляемых блоков (36.x).

5. Способ по п.3, отличающийся тем, что базовый счетчик (BZ) настраивается на соответствующее значение активного счетчика (AZ) перед задействованием управляемых блоков (36.x).

6. Способ по любому из пп.1, 2, 4, 5, отличающийся тем, что программа (30) содержит, по крайней мере, один модуль (56), включающий один или несколько управляемых блоков (36.x).

7. Способ по п.3, отличающийся тем, что программа (30) содержит, по крайней мере, один модуль (56), включающий один или несколько управляемых блоков (36.x).

8. Способ по п.6, отличающийся тем, что модуль (56) является управляемым модулем, в котором изменение счетчика состояния (ZZ) в модуле (56) сравнивается в точке проверки (60), по крайней мере, с одним допустимым значением.

9. Способ по п.7, отличающийся тем, что модуль (56) является управляемым модулем, в

котором изменение счетчика состояния (ZZ) в модуле (56) сравнивается в точке проверки (60), по крайней мере, с одним допустимым значением.

10. Способ по п.8 или 9, отличающийся тем, что до первого соответствующего изменения счетчика состояния (ZZ) в управляемом модуле (56) показатель счетчика состояния (ZZ) или связанное с ним значение сохраняется в буфере локального базового счетчика (LBZ).

11. Способ по любому из пп.1, 2, 4, 5, 7, 8, 9, отличающийся тем, что, по меньшей мере, один из управляемых блоков (36.x) содержит множество управляемых подблоков (68.x), причем значение локального счетчика (LZ) изменяется во время функционирования каждого управляемого подблока (68.x) с тем, чтобы отразить обработку соответствующего управляемого подблока (68.x), а значение локального счетчика (LZ) сравнивается, по крайней мере, в одной из точек проверки (72) как минимум с одним допустимым значением.

12. Способ по п.3, отличающийся тем, что, по меньшей мере, один из управляемых блоков (36.x) содержит множество управляемых подблоков (68.x), причем значение локального счетчика (LZ) изменяется во время функционирования каждого управляемого подблока (68.x) с тем, чтобы отразить обработку соответствующего управляемого подблока (68.x), а значение локального счетчика (LZ) сравнивается, по крайней мере, в одной из точек проверки (72), как минимум, с одним допустимым значением.

13. Способ по п.6, отличающийся тем, что, по меньшей мере, один из управляемых блоков (36.x) содержит множество управляемых подблоков (68.x), причем значение локального счетчика (LZ) изменяется во время функционирования каждого управляемого подблока (68.x) с тем, чтобы отразить обработку соответствующего управляемого подблока (68.x), а значение локального счетчика (LZ) сравнивается, по крайней мере, в одной из точек проверки (72), как минимум, с одним допустимым значением.

14. Способ по п.10, отличающийся тем, что, по меньшей мере, один из управляемых блоков (36.x) содержит множество управляемых подблоков (68.x), причем значение локального счетчика (LZ) изменяется во время функционирования каждого управляемого подблока (68.x) с тем, чтобы отразить обработку соответствующего управляемого подблока (68.x), а значение локального счетчика (LZ) сравнивается, по крайней мере, в одной из точек проверки (72), как минимум, с одним допустимым значением.

15. Способ по любому из пп.1, 2, 4, 5, 7, 8, 9, 12, 13, 14, отличающийся тем, что способ применяется в целях противодействия попыткам вторжения с помощью светового воздействия, при которых происходит манипулирование значением программного счетчика (16) системного ядра (14) посредством светового воздействия.

16. Способ по п.3, отличающийся тем, что способ применяется в целях противодействия попыткам вторжения с помощью светового воздействия, при которых происходит манипулирование значением программного счетчика (16) системного ядра (14) посредством светового воздействия.

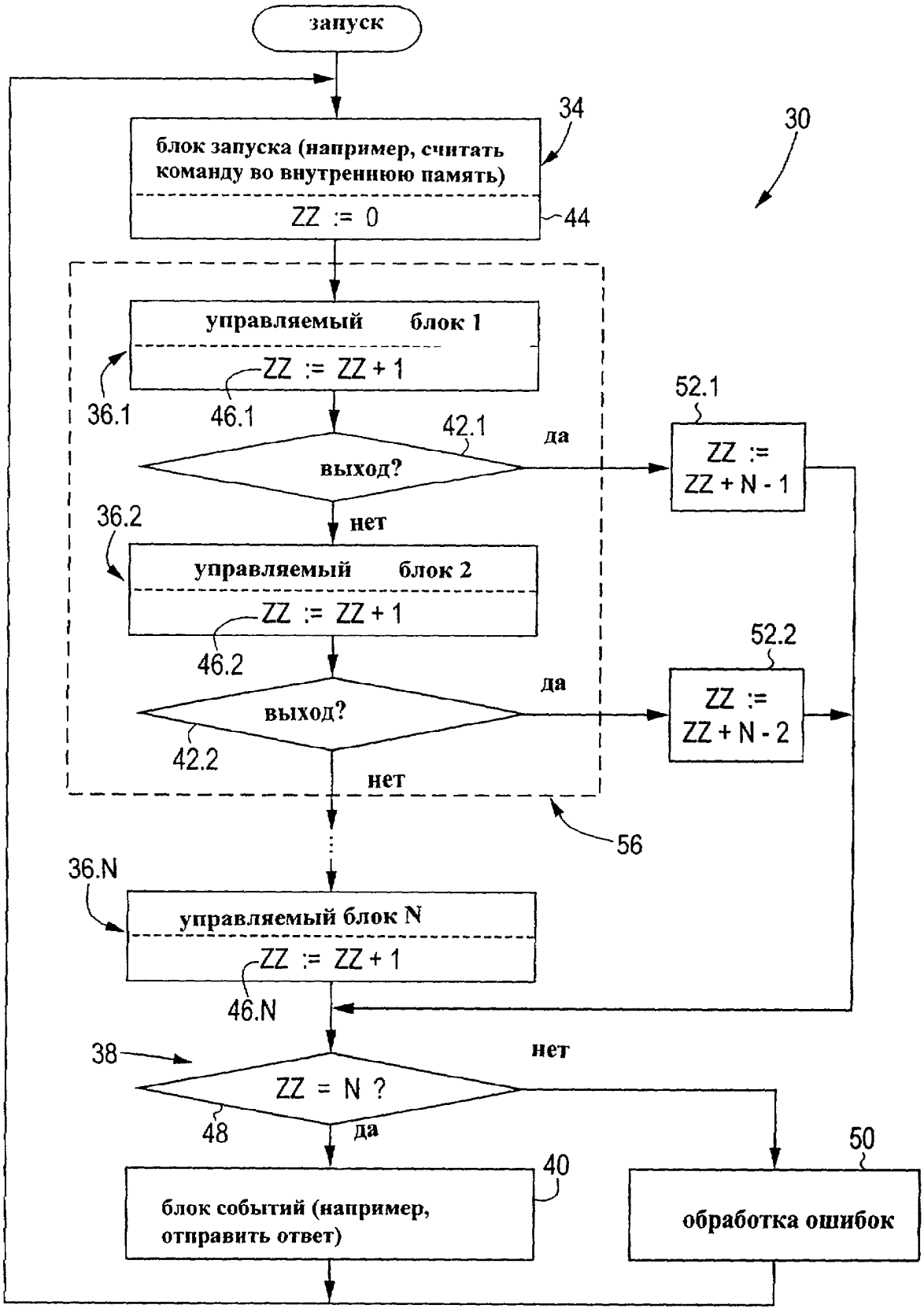
17. Способ по п.6, отличающийся тем, что способ применяется в целях противодействия попыткам вторжения с помощью светового воздействия, при которых происходит манипулирование значением программного счетчика (16) системного ядра (14) посредством светового воздействия.

18. Способ по п.10, отличающийся тем, что способ применяется в целях противодействия попыткам вторжения с помощью светового воздействия, при которых происходит манипулирование значением программного счетчика (16) системного ядра (14) посредством светового воздействия.

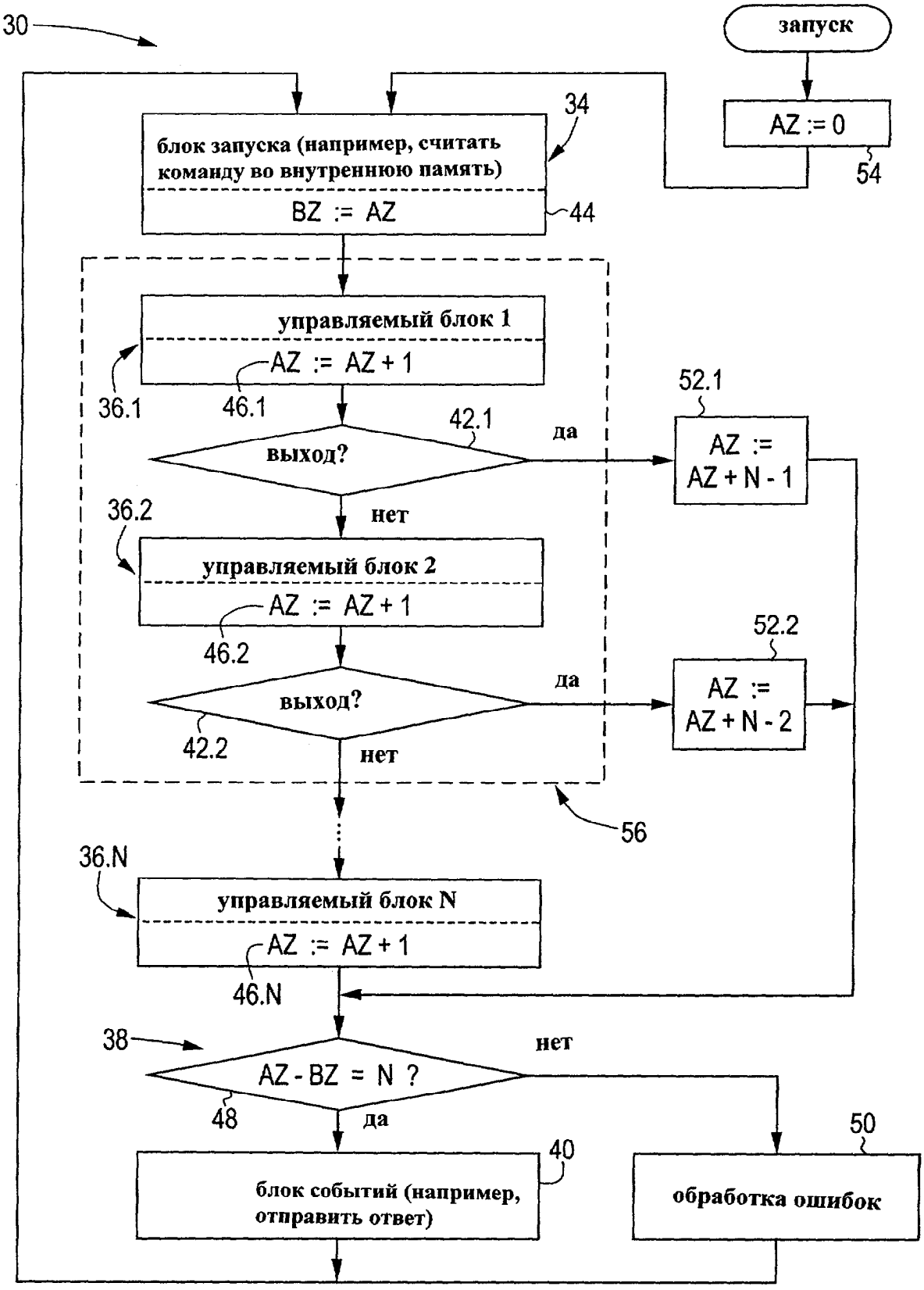
19. Способ, соответствующий формуле 11, отличающийся тем, что способ применяется в целях противодействия попыткам вторжения с помощью светового воздействия, при которых происходит манипулирование значением программного счетчика (16) системного ядра (14) посредством светового воздействия.

20. Портативный носитель данных (10), в частности, плата (чип-карта) или микросхемный модуль (чип-модуль), содержащий программную память (26), системное ядро (14) и, по крайней мере, один счетчик состояния (ZZ), отличающийся тем, что

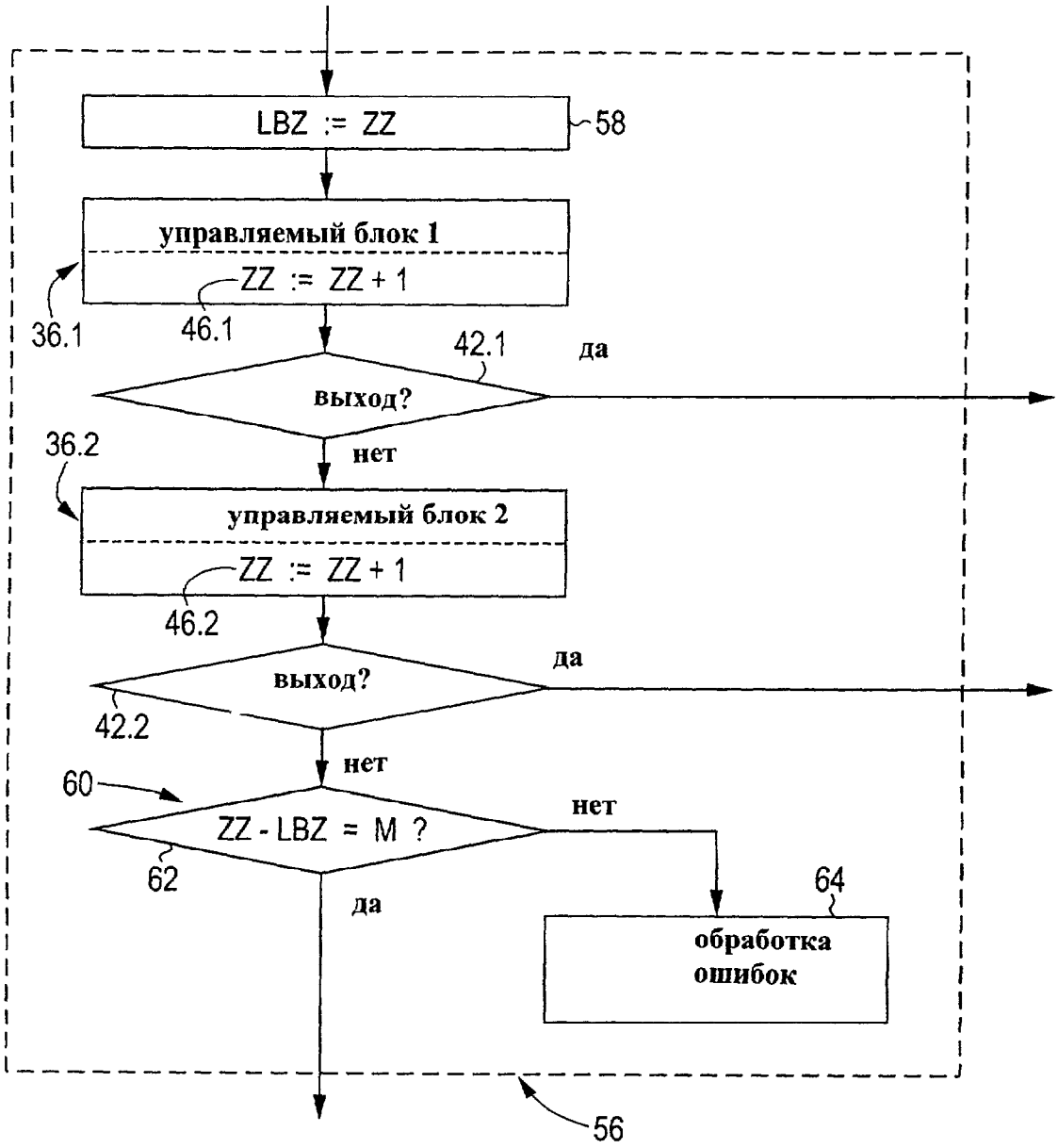
программная память (26) содержит, как минимум, одну программу (30), выполняемую системным ядром (14) и множеством управляемых блоков (36.x), при этом программная память (26) содержит команды, заставляющие системное ядро (14) выполнять способ по любому из пп.1-19.


запуск
34
30
блок запуска (например, считать команду во внутреннюю память)
ZZ := 0
44
управляемый блок 1
ZZ := ZZ + 1
36.1
46.1
42.1
да
выход?
52.1
ZZ := ZZ + N - 1
нет
36.2
управляемый блок 2
ZZ := ZZ + 1
46.2
52.2
да
выход?
ZZ := ZZ + N - 2
42.2
нет
56
36.N
управляемый блок N
ZZ := ZZ + 1
46.N
38
нет
ZZ = N ?
48
да
50
40
блок событий (например, отправить ответ)
обработка ошибок

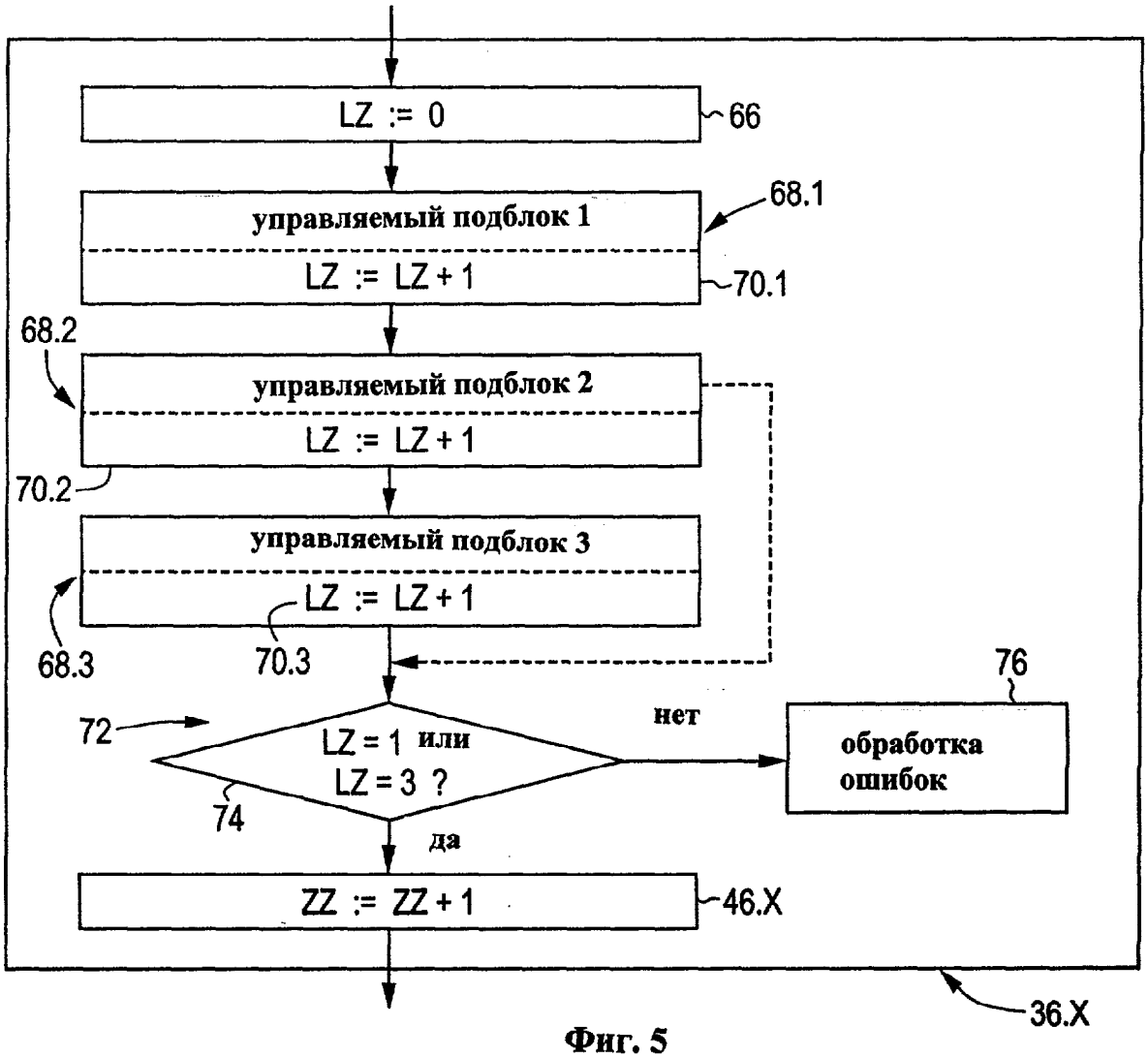
Фиг. 2


30
запуск
AZ := 0
54
34
блок запуска (например, считать команду во внутреннюю память)
BZ := AZ
44
управляемый блок 1
AZ := AZ + 1
36.1
46.1
42.1
выход?
да
52.1
AZ := AZ + N - 1
нет
36.2
управляемый блок 2
AZ := AZ + 1
46.2
52.2
выход?
да
AZ := AZ + N - 2
42.2
нет
56
36.N
управляемый блок N
AZ := AZ + 1
46.N
38
нет
AZ - BZ = N ?
48
да
50
40
блок событий (например, отправить ответ)
обработка ошибок

Фиг. 3


LBZ := ZZ
58
управляемый блок 1
ZZ := ZZ + 1
36.1
46.1
42.1
да
выход?
нет
36.2
управляемый блок 2
ZZ := ZZ + 1
46.2
да
выход?
42.2
нет
60
нет
ZZ - LBZ = M ?
62
64
да
обработка ошибок
56

Фиг. 4


LZ := 0
66
управляемый подблок 1
LZ := LZ + 1
68.1
70.1
68.2
управляемый подблок 2
LZ := LZ + 1
70.2
управляемый подблок 3
LZ := LZ + 1
68.3
70.3
76
72
LZ = 1 или
LZ = 3 ?
74
нет
обработка
ошибок
да
ZZ := ZZ + 1
46.X
36.X

Фиг. 5