РОССИЙСКАЯ ФЕДЕРАЦИЯ

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(19) **RU** (11) **2 643 484**(13) **C2**

(51) МПК
G06F 9/45 (2006.01)

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
G06F 9/445 (2006.01); *G06F 8/40* (2006.01); *G06F 8/41* (2006.01); *G06F 8/65* (2006.01)

(21)(22) Заявка: **2015107557, 03.09.2013**

(24) Дата начала отсчета срока действия патента:
03.09.2013

Дата регистрации:
01.02.2018

Приоритет(ы):
(30) Конвенционный приоритет:
05.09.2012 US 13/604,618

(43) Дата публикации заявки: **27.09.2016** Бюл. № 27

(45) Опубликовано: **01.02.2018** Бюл. № 4

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **04.03.2015**

(86) Заявка PCT:
US 2013/057892 (03.09.2013)

(87) Публикация заявки PCT:
WO 2014/039458 (13.03.2014)

Адрес для переписки:
129090, Москва, ул. Б. Спасская, 25, стр. 3, ООО "Юридическая фирма Городисский и Партнеры"

(72) Автор(ы):
ТЕДЖАНИ Самир (US),
ТРУФИНЕСКУ Адина М. (US),
ШААБАН Яссер (US),
ГБАДЕГЕСИН Аболаде (US),
БАББАР Ашиш (US),
ЦАЙ Мэй-Чинь (US),
РАМАСВАМИ Субраманиан (US),
ФЕРНАНДО Казимир Лакшан (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ ТЕКНОЛОДЖИ ЛАЙСЕНСИНГ, ЭлЭлСи (US)

(56) Список документов, цитированных в отчете о поиске: **US 2011/0258615 A1, 20.10.2011. GB 2467495 A, 04.08.2010. US 2010/0280952 A1, 04.11.2010. RU 2443012 C2, 20.02.2012.**

(54) ГЕНЕРАЦИЯ СОБСТВЕННОГО КОДА ИЗ КОДА НА ПРОМЕЖУТОЧНОМ ЯЗЫКЕ ДЛЯ ПРИЛОЖЕНИЯ

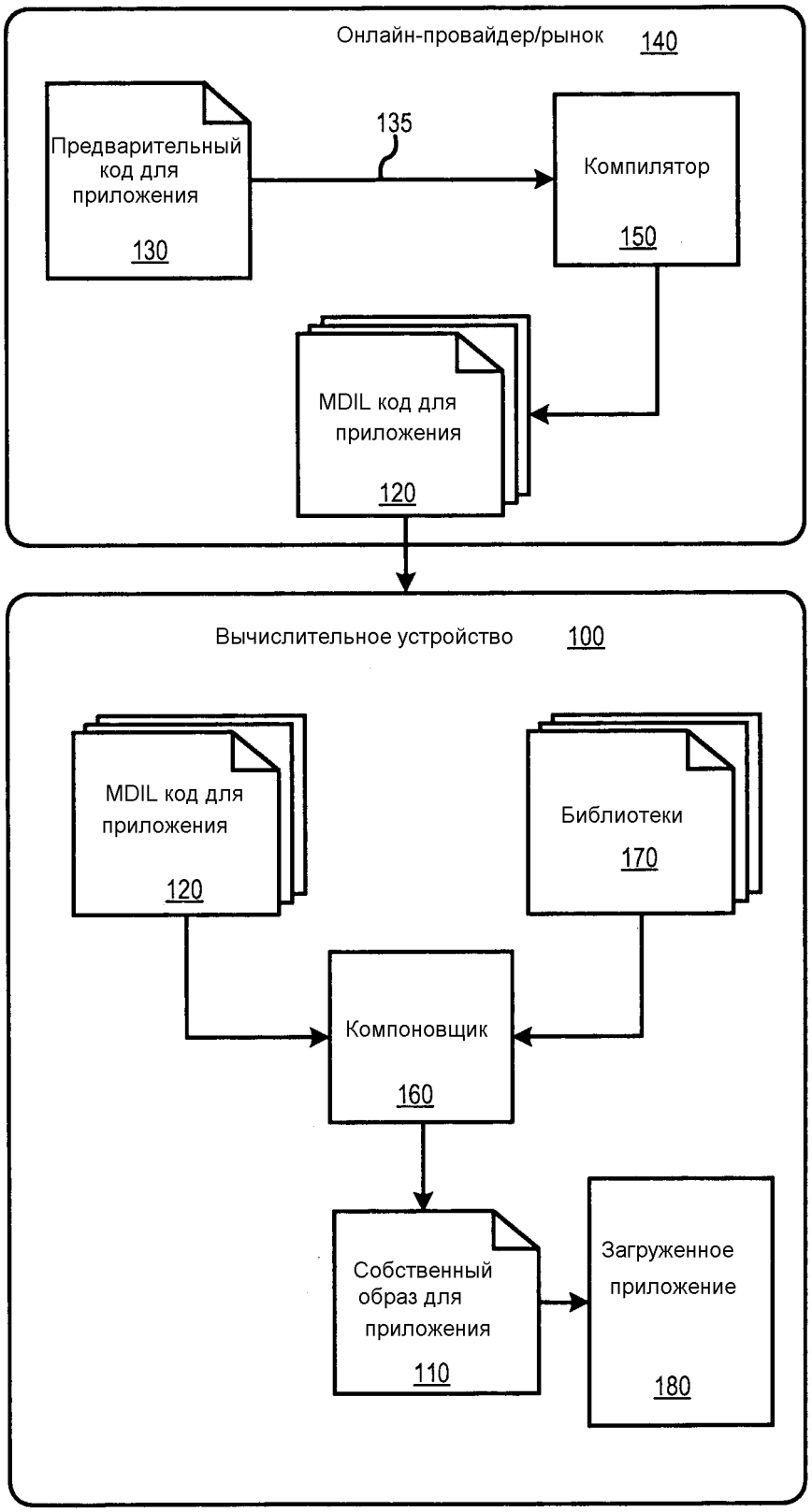
(57) Реферат:

Изобретение относится к средствам установки, исполнения и/или обновления управляемых приложений путем генерации собственного кода из кода на промежуточном языке. Технический результат заключается в расширении арсенала средств установки, исполнения и/или обновления управляемых приложений. В способе генерации собственного кода из кода на промежуточном языке для приложения: принимают в вычислительном устройстве характерный для типа устройства установочный пакет для приложения от онлайн-провайдер, пакет включает в себя файлы на машинно-зависимом промежуточном языке MDIL-файлы, генерируемые онлайн-провайдером для приложения, список связывания, который включает в себя перечень MDIL-файлов приложения, которые подлежат связыванию для генерации собственного образа для приложения, и перечень набора из одной или более библиотек, которые подлежат связыванию с соответствующими MDIL-файлами, сохраняют собственный образ на вычислительном устройстве для использования при загрузке

RU 2643484 C2

RU 2643484 C2

приложения для исполнения. 2 н. и 6 з.п. ф-лы, 10 ил.



ФИГ.1

RU 2643484 C2

RU 2643484 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 643 484**(13) **C2**

(51) Int. Cl.
G06F 9/45 (2006.01)

(12) ABSTRACT OF INVENTION

(52) CPC
G06F 9/445 (2006.01); *G06F 8/40* (2006.01); *G06F 8/41* (2006.01); *G06F 8/65* (2006.01)

(21)(22) Application: **2015107557, 03.09.2013**

(24) Effective date for property rights:
03.09.2013

Registration date:
01.02.2018

Priority:
(30) Convention priority:
05.09.2012 US 13/604,618

(43) Application published: **27.09.2016** Bull. № **27**

(45) Date of publication: **01.02.2018** Bull. № **4**

(85) Commencement of national phase: **04.03.2015**

(86) PCT application:
US 2013/057892 (03.09.2013)

(87) PCT publication:
WO 2014/039458 (13.03.2014)

Mail address:
129090, Moskva, ul. B. Spasskaya, 25, str. 3, OOO "Yuridicheskaya firma Gorodisskij i Partnery"

(72) Inventor(s):
TEDZHANI Samir (US),
TRUFINESKU Adina M. (US),
SHAABAN Yasser (US),
GBADEGESIN Abolade (US),
BABBAR Ashish (US),
TSAJ Mej-Chin (US),
RAMASVAMI Subramanian (US),
FERNANDO Kazimir Lakshan (US)

(73) Proprietor(s):
MAJKROSOFT TEKNOLODZHI LAJSENSING, ElElSi (US)

RU 2643484 C2

RU 2643484 C2

(54) GENERATION OF OWN CODE FROM CODE IN INTERMEDIATE LANGUAGE FOR APPLICATION

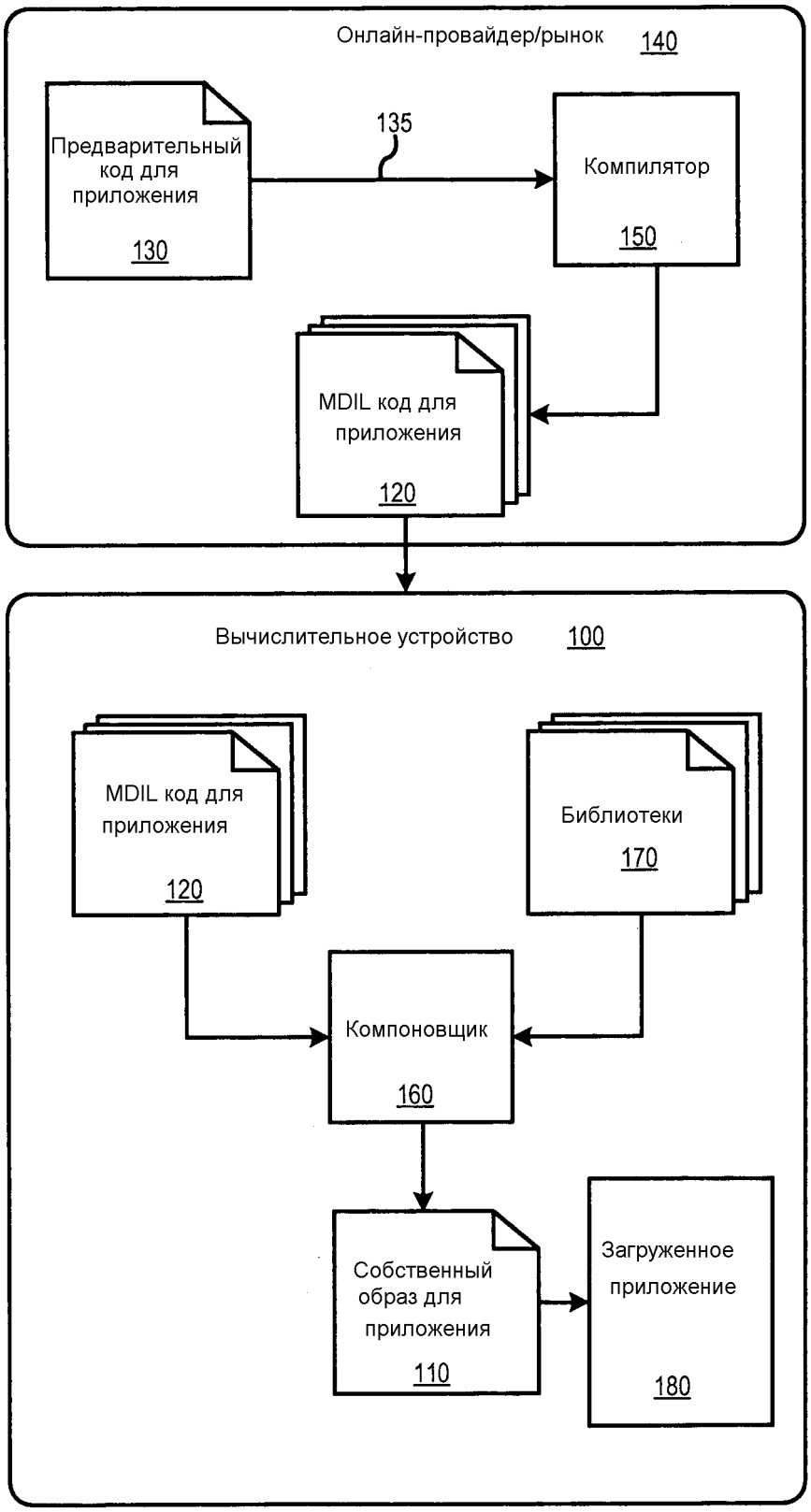
(57) Abstract:

FIELD: information technology.

SUBSTANCE: in the method for generating own code from a code in intermediate language for an application: in compute device an installation package for an application typical for the type of device is received from the online provider, the package includes files on machine-dependent intermediate language MDIL-files generated by online provider for an application, a binding list which includes a list of application MDIL-files subject to binding to generate own image for an application and a list of a set from one or more libraries subject to binding with corresponding MDIL-files, own image is saved on the computing device for use when the application is loaded for execution.

EFFECT: expanding the arsenal of means of installing, executing and updating managed applications.

8 cl, 10 dwg



ФИГ.1

RU 2643484 C2

ПРЕДПОСЫЛКИ

[001] Различные мобильные устройства также поддерживают приложения, которые могут быть загружены в представлении, которое не может выполняться впрямую, но может выполняться при компиляции на мобильном устройстве с использованием динамической (“точно к нужному моменту”) компиляции (JIT компиляции). Хотя JIT компиляция кода используется для исполнения приложений на компьютере, исполнение приложений с использованием JIT компиляции имеет ограничения, в том числе дополнительное время, необходимое для компиляции, когда приложение исполняется, и, возможно, неоптимальные решения, принятые во время компиляции.

СУЩНОСТЬ ИЗОБРЕТЕНИЯ

[002] Среди других нововведений, описанных здесь, настоящее раскрытие представляет различные характерные варианты инструментальных средств и способов для установки, выполнения и/или обновления управляемых приложений путем генерации собственного кода из кода на промежуточном языке. Согласно одному примерному методу, вычислительное устройство получает машинно-зависимый код на промежуточном языке (MDIL код), сгенерированный онлайн-провайдером для приложения. Вычислительное устройство инсталлирует (устанавливает) приложение на вычислительном устройстве путем генерации собственного (машинного) образа для приложения, в том числе путем связывания части MDIL кода с одной или более библиотеками на вычислительном устройстве. Кроме того, собственный образ сохраняется на вычислительном устройстве для использования при загрузке приложения для исполнения.

[003] В соответствии с примерным инструментальным средством, вычислительное устройство получает от онлайн-провайдера установочный пакет для приложения, где установочный пакет включает в себя набор машинно-зависимых файлов на промежуточном языке (MDIL файлов). Кроме того, на компоновщик (модуль связывания) вычислительного устройства предоставляется по меньшей мере один файл из набора MDIL файлов и одна или более библиотек, подлежащих связыванию с по меньшей мере одним файлом. Компоновщик генерирует собственный образ для приложения путем связывания MDIL кода по меньшей мере одного MDIL файла с использованием одной или более библиотек.

[004] В другом примерном методе, онлайн-провайдер генерирует MDIL код из предварительного кода для приложения. Компьютерная система генерирует набор MDIL файлов для приложения путем компиляции предварительного кода приложения в MDIL код, затем подписывает соответствующие файлы из набора MDIL файлов, чтобы указывать, что соответствующие файлы являются доверенными как происходящие из онлайн-рынка. Компьютерная система генерирует список связывания, который идентифицирует соответствующие файлы из набора MDIL файлов, а также генерирует установочный пакет для приложения, содержащего набор MDIL файлов для приложения и список связывания. Затем компьютерная система предоставляет, на онлайн-рынке, установочный пакет для скачивания.

[005] При приеме установочного пакета, вычислительное устройство получает MDIL код и генерирует собственный образ для приложения, чтобы устанавливать приложение на вычислительном устройстве. Кроме того, процессор (механизм) среды выполнения вычислительного устройства и/или одна или более библиотек, используемых установленным приложением, могут быть обновлены на вычислительном устройстве во время обновления вычислительного устройства, и приложение автоматически обновляется в ответ. Приложение обновляется путем генерации обновленного

собственного образа для приложения с использованием одной или более библиотек, которые были обновлены. Обновленный собственный образ генерируется так, что он является исполняемым с использованием обновленного механизма среды выполнения на вычислительном устройстве. После того как обновленный собственный образ сгенерирован, приложение запускается путем загрузки обновленного собственного образа вместо сгенерированного ранее собственного образа.

[006] Это краткое изложение приведено, чтобы ввести выбор концепций в упрощенной форме, которые дополнительно описаны ниже. Это краткое изложение не предназначено для определения ключевых признаков или существенных признаков заявленного изобретения, а также не предназначено для использования, чтобы ограничивать объем заявленного изобретения. Указанные выше и другие цели, признаки и преимущества упомянутых технологий станут более очевидными из последующего подробного описания, которое выполнено со ссылкой на прилагаемые чертежи.

КРАТКОЕ ОПИСАНИЕ ЧЕРТЕЖЕЙ

[007] Фиг. 1 - схема, иллюстрирующая пример онлайн-провайдера/рынка и примерное вычислительное устройство, которое загружает собственный образ приложения, установленного путем связывания машинно-зависимого кода на промежуточном языке (MDIL кода).

[008] Фиг. 2 - блок-схема последовательности операций примерного способа связывания MDIL кода приложения, чтобы установить приложение на вычислительном устройстве.

[009] Фиг. 3 - схема, иллюстрирующая пример онлайн-провайдера/рынка и примерное вычислительное устройство, которое может устанавливать одно или более управляемых приложений путем генерации одного или более собственных образов из кода на промежуточном языке, который предоставляется и генерируется онлайн-провайдером/ рынком (или другими онлайн-провайдерами/ рынками).

[010] На Фиг. 4 представлена блок-схема последовательности операций примерного способа для установки и обновления приложения через генерацию собственного кода с использованием кода в MDIL.

[011] На Фиг. 5 представлена блок-схема последовательности операций примерного способа генерации установочного пакета для приложения, где установочный пакет может быть предоставлен для загрузки.

[012] На Фиг. 6 показана схема вычислительного устройства, которое может генерировать один или более обновленных собственных образов для одного или более установленных приложений.

[013] На Фиг. 7 представлена блок-схема последовательности операций примерного способа обновления приложения путем генерации собственного образа для приложения из кода в MDIL.

[014] На Фиг. 8 показана схема, изображающая примерное мобильное устройство, в котором могут быть реализованы по меньшей мере некоторые из раскрытых вариантов осуществления.

[015] На Фиг. 9 показана схема, иллюстрирующая обобщенный пример подходящей среды реализации для по меньшей мере некоторых из раскрытых вариантов осуществления.

[016] На Фиг. 10 показана схема, иллюстрирующая обобщенный пример подходящей вычислительной среды для по меньшей мере некоторых из раскрытых вариантов осуществления.

ПОДРОБНОЕ ОПИСАНИЕ

[017] Настоящее раскрытие представляет различные характерные варианты инструментальных средств и способов для установки, исполнения и/или обновления управляемых приложений посредством генерации собственного кода из кода на промежуточном языке. Например, вычислительное устройство принимает MDIL код, сгенерированный онлайн-провайдером для приложения. Вычислительное устройство устанавливает приложение, генерируя собственный образ приложения. В рамках генерации собственного образа, вычислительное устройство связывает часть MDIL кода с одной или более библиотек на вычислительном устройстве. Вычислительное устройство сохраняет собственный образ для использования при загрузке приложения для исполнения. Таким образом, выполнение приложения может извлечь выгоду из предыдущей офлайн компиляции по меньшей мере части приложения (для получения MDIL кода, подходящего для устройства). В то же время, этот подход может облегчить обновление приложения, когда происходит изменение в библиотеке, используемой приложением, и/или MDIL коде для приложения.

[018] Различные особенности способов и инструментальных средств, описанных здесь, могут быть использованы в комбинации или отдельно, в зависимости от реализации. Различные признаки способов и инструментальных средств, описанных в данном документе, влияют на различные этапы обработки, в том числе прием, установку, исполнение, обновление приложения и обновление устройства. Термин «прием» обычно относится к процессу, в котором разработчик выгружает приложение к онлайн-провайдеру, который обрабатывает приложение для эффективного исполнения на одном или более типах вычислительных устройств, проверяет приложение и делает приложение доступным для загрузки. Термин «установка» обычно относится к процессу адаптации приложения для работы на конкретном вычислительном устройстве, который преобразует приложение в форму, более подходящую для быстрого и безопасного выполнения на устройстве (например, преобразование MDIL кода в собственные инструкции во время связывания путем разрешения ссылок на библиотеки и другие ресурсы, хранение собственного образа, который может быть загружен для исполнения, маркировку приложения как доверительного и т.д.). Во время исполнения, доверительный собственный образ для приложения может быть быстро загружен и запущен. Термин «обновление приложения» обычно относится к процессу, в котором обновляется приложение (например, MDIL код для приложения), который может включать в себя переустановку приложения на устройстве. Термин «обновление вычислительного устройства» обычно относится к процессу, который следует после того, как механизм среды выполнения, библиотека или другой ресурс, на который ссылается приложение, был обновлен, что обычно включает в себя повторное связывание приложений, которые зависят от библиотеки или другого ресурса, который изменился.

Примерная система для установки и загрузки приложения путем генерации собственного кода из кода в MDIL

[019] Фиг. 1 является схемой, иллюстрирующей примерного онлайн-провайдера/рынок и примерное вычислительное устройство 100, которое загружает собственный образ 110 приложения, установленного путем связывания MDIL кода 120. На Фиг. 1 предварительный код 130 для приложения принимается онлайн-провайдером 140. Например, предварительный код 130 может быть кодом для приложения, который является исходным кодом или кодом на промежуточном языке, таком как Microsoft Intermediate Language (MSIL). В блоке 135, предварительный код поступает на компилятор 150, который компилирует предварительный код 130 в MDIL код 120 для приложения.

Предварительный код 130 может быть представлением более высокого уровня кода для приложения, чем MDIL код 120. Например, предварительный код 130 может быть на промежуточном языке (IL), который находится на более высоком уровне, чем MDIL код 120, который находится на уровне близком к машинному коду. MDIL код 120 может быть машинно-зависимым, так что он включает в себя собственный код, который основан на наборе инструкций процессора. Набор инструкций процессора может быть набором инструкций для процессора ARM, процессора x86 или другого процессора. MDIL код 120 также может включать в себя неразрешенные псевдо инструкции с символическими ссылками, которые могут быть разрешены, чтобы генерировать собственный код, посредством связывания.

[020] MDIL код 120 для приложения принимается в вычислительном устройстве 100. Как часть установки приложения, MDIL код 120 подается на компоновщик 160, который связывает MDIL код 120 с одной или более библиотеками 170 на устройстве. Одна или более библиотек 170 могут быть библиотеками кодов и/или библиотеками архитектуры для классов, типов или т.п. Путем связывания MDIL кода 120 на вычислительном устройстве 100, компоновщик 160 генерирует собственный образ 110 приложения. Собственный образ 110 включает в себя собственный код для приложения и сохраняется на устройстве. В блоке 180 собственный образ загружается для исполнения приложения.

Примерный способ связывания MDIL кода для установки приложения на вычислительное устройство

[021] На Фиг. 2 представлена блок-схема последовательности операций примерного способа 200 связывания MDIL кода приложения, чтобы установить приложение на вычислительном устройстве. В различных вариантах осуществления иллюстрируемые блоки способа на Фиг. 2 может быть объединены, разделены на суб-блоки или опущены.

[022] На Фиг. 2, вычислительное устройство получает MDIL код для приложения из онлайн-провайдера в блоке 210. Например, вычислительное устройство может получить один или более файлов, которые включают в себя код для приложения, причем код выполнен на IL, который является машинно-зависимым, таким как код на машинно-зависимом промежуточном языке, как описано в опубликованной патентной заявке США № US 2011/0258615 или на другом машинно-зависимом промежуточном языке. Поскольку MDIL код принимается в вычислительном устройстве от онлайн-провайдера, MDIL код может генерироваться из компиляции онлайн-провайдера или в другом месте в облаке. Например, онлайн-провайдер компилирует предварительный код для приложения, чтобы генерировать MDIL код для приложения.

[023] В некоторых реализациях, принимается установочный пакет для приложения, причем установочный пакет включает в себя MDIL код для приложения. Установочный пакет для приложения может включать в себя другую информацию для приложения, такую как файлы ресурсов для приложения. MDIL код для приложения может сохраняться на вычислительном устройстве для установки приложения и последующего обновления приложения.

[024] В блоке 220, приложение устанавливается на вычислительном устройстве путем генерации собственного образа приложения, что включает в себя связывание MDIL кода приложения. Например, компоновщик может связывать MDIL код с одной или более библиотеками, доступными на вычислительном устройстве, чтобы генерировать собственный образ для приложения, которое является исполняемым. Приложение может быть управляемым приложением, исполняемым с использованием механизма среды выполнения или механизма исполнения инструкций, такого как Common Language Runtime архитектуры .NET или другим механизмом среды выполнения. Механизм среды

выполнения и/или механизм исполнения инструкций может быть слоем программного обеспечения, на котором приложение в конечном счете работает. В некоторых реализациях, управляемое приложение представляет собой приложение, которое исполняется с использованием архитектуры, которая включает в себя одну или более совместно используемых библиотек и одну или более управляемых сред среды выполнения кода. Например, библиотеки являются библиотеками базовых классов, которые являются общими для разных приложений. В некоторых реализациях, библиотека на вычислительном устройстве может быть образом библиотеки в собственном и/или машинном коде, который может быть загружен в память для использования управляемым приложением, которое также загружается в память. В некоторых реализациях, генерируемый собственный образ может быть сгенерирован компоновщиком, так что собственный образ является исполняемым посредством механизма среды выполнения или механизмом исполнения инструкций, который используется для запуска приложения на вычислительном устройстве. Например, компоновщик может использовать и/или ссылаться на механизм среды выполнения для приложения при связывании (компоновке), так что связывание является подходящим для исполнения с использованием механизма среды выполнения.

[025] В некоторых реализациях, связывание, выполняемое компоновщиком, разрешает MDIL код и генерирует собственный код для приложения, который включается в собственный образ для приложения. Собственный код может быть машинным кодом, который является исполняемым на конкретном процессоре и который использует инструкции из набора инструкций процессора. Набор инструкций процессора может быть набором инструкций для процессора ARM, процессора x86 или другого процессора. В некоторых реализациях, при генерации собственного образа, для каждого MDIL файла в наборе файлов, полученных для приложения, соответствующий MDIL файл и одна или более библиотек, которые должны быть связаны с MDIL кодом файла, предоставляются компоновщику. Например, один или более путей файлов к MDIL файлам и/или файлам библиотек могут быть предоставлены компоновщику. Компоновщик связывает соответствующие MDIL файлы с библиотеками, предоставленными компоновщику, чтобы сгенерировать собственный образ для приложения. Собственный образ может включать в себя собственный код, такой как машинный код, который является исполняемым процессором вычислительного устройства. (В дополнение к собственному образу, по меньшей мере некоторый код для приложения может поддерживаться в форме IL, для “точно к нужному моменту” компиляции или другой компиляции, прежде чем приложение запускается, или когда приложение запускается. Это может быть полезно, когда ресурсы часто меняются, или когда значение типа или другая ссылка не может быть разрешена, когда приложение устанавливается, и остальной MDIL код связывается с библиотеками на устройстве.)

[026] В некоторых реализациях, приложение может быть установлено на устройстве с помощью менеджера пакетов. Например, во время установки приложения, принятый установочный пакет для приложения может быть распакован с помощью менеджера пакетов. Если установочный пакет и/или один или более файлов, включенных в установочный пакет, сжаты, то менеджер пакетов может распаковать такие файлы. Если один или более файлов не сжаты, то менеджер пакетов не распаковывает такие файлы. Менеджер пакетов может предоставить каждый из MDIL файлов, которые должны связываться, на компоновщик вместе с библиотеками, которые должны быть связаны с соответствующими MDIL файлами. В некоторых реализациях менеджер пакетов предоставляет MDIL файлы и/или одну или более библиотек путем

предоставления ссылок на MDIL файлы и/или одну или более библиотек компоновщику. Компоновщик связывает соответствующие MDIL файлы с предоставленными библиотеками для соответствующих MDIL файлов для генерации собственного образа приложения. Собственный образ может быть сгенерирован компоновщиком так, что собственный образ является исполняемым с использованием механизма среды выполнения, доступного на вычислительном устройстве для запуска приложения.

[027] В блоке 230, собственный образ приложения сохраняется на вычислительном устройстве для использования при загрузке приложения для исполнения. Например, сгенерированный собственный образ загружается для исполнения приложения (и MDIL код, который не является исполняемым, не используется для исполнения приложения). В некоторых реализациях, собственный образ определяется как доверительный перед загрузкой для исполнения. В других реализациях, вычислительное устройство не выполняет никакого определения относительно того, является ли собственный образ доверительным, прежде чем собственный образ загружается для исполнения.

Примерная система для генерации собственного кода из предоставленного MDIL кода для установки и исполнения приложения

[028] Фиг. 3 является схемой, иллюстрирующей примерного онлайн-провайдера/рынок и примерное вычислительное устройство 350, которое может устанавливать одно или более управляемых приложений путем генерации одного или более собственных образов из кода на промежуточном языке, таком как MDIL, который предоставляется и генерируется онлайн-провайдером (или другими онлайн-провайдерами). На Фиг. 3, онлайн-провайдер 305 использует компилятор 310, чтобы компилировать предварительный код 315 в один или более MDIL файлов 320, которые являются файлами, которые включают в себя код в MDIL. Набор из одного или более MDIL файлов 320 упакован в установочный пакет 325 для приложения. Каждый из MDIL файлов может быть подписан как доверительный, как показано в блоке 330. Один или более файлов 335, которые включают в себя информацию поддержки, такие как файлы ресурсов или контент приложения, могут быть включены в установочный пакет 325. Установочный пакет 325 хранится онлайн-провайдером 305 и предлагается для загрузки, как показано в блоке 340. Прежде чем установочный пакет 325 будет загружен от онлайн-провайдера 305, установочный пакет 325 сам может быть подписан как доверительный, как показано в блоке 345.

[029] Таким образом, установочный пакет 325 предлагается в качестве части каталога приложений в онлайн-рынке. Различные версии установочного пакета 325 могут быть предложены для различных типов вычислительных устройств. Например, различные типы MDIL кода (адаптированные для разных типов процессоров) включаются в различные установочные пакеты для различных типов вычислительных устройств. Установочные пакеты, доступные для разных типов вычислительных устройств, могут включать в себя версию с не-MDIL кодом (например, MSIL код в XAP файле) в целях обратной совместимости.

[030] Вычислительное устройство 350 загружает один или более установочных пакетов, включая установочный пакет 325, от онлайн-провайдера в блоке 355. В качестве части установки приложения, менеджер 360 пакетов вычислительного устройства 350 распаковывает один или более MDIL файлов 320 и один или более файлов 335, которые включают в себя информацию поддержки для приложения. Менеджер пакетов также вызывает компоновщик 365. Для каждого MDIL файла набора из одного или более MDIL файлов 320, менеджер 360 пакетов предоставляет соответствующий MDIL файл, например MDIL файл 322, компоновщику 365 вместе с одной или более библиотеками,

подлежащими связыванию с соответствующим MDIL файлом, такими как одна или более библиотек 370. Компоновщик 365 связывает соответствующий MDIL файл с одной или более библиотеками, предоставленными для соответствующего MDIL файла, чтобы генерировать собственный код. Библиотеки, предоставленные для MDIL файлов (такие, как одна или более библиотек 370), включены в набор библиотек 375, которые находятся на вычислительном устройстве 350. Используя собственный код, сгенерированный из связывания одного или более MDIL файлов 320, компоновщик генерирует собственный образ 380 для приложения. Собственный образ 380 может быть сгенерирован компоновщиком 365 таким образом, что собственный образ 380 является выполняемым с использованием механизма среды выполнения на вычислительном устройстве, которое может запустить приложение. Собственный образ 380 сохраняется как часть установки приложения на устройстве, как показано в блоке 385. Вычислительное устройство 350 также может установить одно или более других приложений и сохранить сгенерированные собственные образы, соответствующие соответственным другим приложениям.

[031] Как показано в блоке 390, собственный образ 380 может быть подписан как доверительный код, который был сгенерирован из доверительного кода компоновщиком 365. Собственный образ 380 может быть подписан, прежде чем он будет сохранен и/или загружен для использования при исполнении приложения. В блоке 395, собственный образ 380 загружается для исполнения приложения. При загрузке, собственный код собственного образа исполняется вычислительным устройством 305, чтобы обеспечить функциональность приложения.

Примерный способ установки и обновления приложения посредством генерации собственного кода с использованием кода в MDIL

[032] На Фиг. 4 представлена блок-схема последовательности операций примерного способа 400 для установки и обновления приложения посредством генерации собственного кода с использованием кода в MDIL. На Фиг. 4, как показано в блоке 410, генерируется установочный пакет, который включает в себя MDIL код. В некоторых реализациях, MDIL код могут быть включен в один или более файлов. В некоторых реализациях, предварительный код может быть скомпилирован для генерации MDIL кода.

[033] В некоторых реализациях, MDIL код для приложения включает в себя собственный код на основе набора инструкций процессора, но также может включать в себя псевдо инструкции. Например, псевдо инструкция включает в себя символическую ссылку и/или ключевое поле, которые не могут быть разрешены, когда генерируется MDIL. Таким образом, в некоторых реализациях, псевдо инструкции могут включать в себя символические ссылки на поля, типы или т.п. Например, символические ссылки могут быть маркерами, которые идентифицируют поле или тип для использования без специального указания смещения поля или размера типа. В некоторых реализациях, символическая ссылка может быть маркером, который может интерпретироваться компоновщиком, чтобы определять собственный код, который должен генерироваться для символической ссылки. Например, для маркера, такого как маркер поля, который ссылается на поле объекта, компоновщик на конкретном вычислительном устройстве может (1) компоновать объект, как он доступен на вычислительном устройстве, через библиотеку и (2) определять соответствующее смещение для поля объекта, при генерации исполняемых инструкций для вычислительного устройства. Таким образом, в некоторых реализациях, смещения поля объектов в MDIL коде могут быть символическими, а не жестко кодированными как численные смещения (так, как в соответствующем

собственном коде).

[034] Псевдо инструкция может абстрагировать детали реализации, которые должны быть определены и/или разрешены, когда псевдо инструкция связана с соответствующей библиотекой на вычислительном устройстве. В некоторых реализациях, MDIL код включает в себя псевдо инструкции, которые включают символические ссылки, которые могут быть преобразованы компоновщиком для получения собственных инструкций, исполняемых процессором. Например, MDIL код может включать в себя псевдо инструкции, которые могут быть преобразованы компоновщиком в машинные инструкции для доступа к полю объекта, что управляется механизмом среды выполнения, таким как Common Language Runtime (CLR) архитектуры .NET и т.п. В некоторых реализациях, псевдо инструкции в MDIL коде могут быть подобны машинному коду на собственном языке, так что регистры уже распределены, но смещения не включены в одну или более библиотек и/или классов. Например, смещение поля, включенное в собственную инструкцию, как преобразованную из псевдо инструкции, может быть определено компоновщиком и может зависеть от версии доступной библиотеки и/или от вычислительного устройства. В некоторых реализациях, MDIL код может включать в себя псевдо инструкции, которые могут быть преобразованы компоновщиком в машинные инструкции для поиска реализации или экземпляра обобщенного типа или метода.

[035] Со ссылкой на Фиг. 4, в блоке 420, вычислительное устройство получает MDIL код для приложения. Например, вычислительное устройство может получить установочный пакет для приложения, который включает в себя MDIL файлы, которые содержат MDIL код для приложения. В некоторых реализациях, установочный пакет может включать в себя список связывания, который включает в себя перечень MDIL файлов приложения, которые должны быть связаны, чтобы генерировать собственный образ для приложения, и перечень набора из одной или более библиотек, которые должны быть связаны с соответствующими MDIL файлами. В некоторых реализациях одна или более из библиотек, подлежащих связыванию с MDIL кодом приложения, являются библиотеками, которые были объявлены и/или включены в исходный код приложения. Одна или более других библиотек, подлежащих связыванию с MDIL кодом приложения, могут быть библиотеками, которые не объявлены и/или не включены в исходный код приложения.

[036] В блоке 430, приложение устанавливается на вычислительном устройстве путем генерации собственного образа приложения. В некоторых реализациях, приложение может быть установлено с помощью менеджера пакетов. Например, менеджер пакетов может определить, что установочный пакет подписан как доверительный, прежде чем использовать его файлы для установки приложения на вычислительном устройстве, и если установочный пакет не подписан как доверительный, менеджер пакетов не использует установочный пакет для установки приложения на вычислительном устройстве. Менеджер пакетов может предоставить каждый из MDIL файлов, которые должны быть связаны, компоновщику вместе с библиотеками, которые должны быть связаны с соответствующими MDIL файлами. В некоторых реализациях, вместо или в дополнение к определению того, весь ли установочный пакет подписан как доверительный, MDIL файлы могут определяться как подписанные в качестве доверительных, прежде чем использоваться для генерации собственного образа для приложения, и MDIL файлы, которые определены как не подписанные в качестве доверительных, не используются для создания собственного образа для приложения. В некоторых реализациях менеджер пакетов может получать и использовать список

связывания, чтобы автоматически определять и идентифицировать MDIL файлы, которые должны быть связаны, и соответствующие библиотеки, которые должны быть связаны с кодом соответствующих MDIL файлов для приложения. Список связывания может быть в машиночитаемом формате, таком как язык программирования, сценариев или разметки. Например, список связывания может быть включен в файл, который включает в себя информацию, организованную с использованием расширяемого языка разметки (XML).

[037] В некоторых реализациях менеджер пакетов предоставляет MDIL файлы и/или одну или более библиотек компоновщику путем предоставления ссылок на MDIL файлов и/или одну или более библиотек компоновщику. Компоновщик связывает соответствующие MDIL файлы с предоставленными библиотеками для соответствующих MDIL файлов для создания собственного образа приложения. На высоком уровне, компоновщик считывает MDIL инструкции, преобразует IL инструкции в собственные инструкции и проходит через инструкции, которые уже находятся в собственной форме. Например, компоновщик может связывать MDIL код с одной или более библиотеками на вычислительном устройстве, чтобы генерировать собственный образ для приложения, которое является исполняемым. Собственный образ может генерироваться компоновщиком так, что собственный образ является исполняемым с использованием механизма среды выполнения на вычислительном устройстве, который доступен для компоновщика, то есть для работы приложения. Механизмом среды выполнения может быть слой программного обеспечения, который в конечном итоге может запустить приложение на вычислительном устройстве. В некоторых реализациях, компоновщик может быть запущен с использованием информации из сценария (скрипта), пользовательского ввода и/или другой информации. В некоторых реализациях кода связывания, MDIL код предоставляется компоновщику. Компоновщик разрешает псевдо инструкции для генерации соответствующих собственных инструкций. Например, из символических ссылок (например, маркера поля) в псевдо инструкциях, компоновщик может генерировать собственный код путем определения и специфицирования численных смещений для полей на основе формата типа. В некоторых реализациях, псевдо инструкции MDIL могут быть использованы, чтобы создавать собственные инструкции, которые могут отличаться по длине от псевдо инструкции. Альтернативно, собственные инструкции могут не отличаться по длине от псевдо инструкций. В некоторых реализациях, по меньшей мере, часть MDIL кода для приложения может быть близкой к машинному коду, включенному в собственный образ приложения; однако другие части MDIL кода могут обеспечивать уровень абстракции, который позволяет связывать часть MDIL кода с библиотекой, которая может быть доступна в одной или более версиях. Через связывание, MDIL код может быть использован для генерации исполняемого кода, который исполняется с использованием конкретной версии библиотеки. Например, связывание MDIL кода с первым набором библиотек может генерировать исполняемый код для приложения. Или связывание MDIL код с набором библиотек с одной или более библиотеками, находящимися в другой версии, может также генерировать исполняемый код для приложения. Наборы версий одной или более библиотек могут иметь перекрывающиеся версии библиотек.

[038] В некоторых реализациях, связывание MDIL кода генерирует код, который может быть непосредственно выполняемым процессором без дополнительной компиляции. Например, связывание MDIL кода с библиотекой включает в себя генерацию собственной инструкции из инструкции в MDIL коде. В некоторых реализациях, MDIL код включает в себя псевдо инструкции, которые включают в себя символические

ссылки, которые преобразуются и/или связываются компоновщиком для создания собственных инструкций, исполняемых процессором. Например, компоновщик может преобразовать псевдо инструкции в инструкции собственного кода для доступа к полю объекта, который может управляться механизмом среды выполнения на вычислительном устройстве. В некоторых реализациях, части MDIL кода могут быть аналогичны генерируемому собственному коду таким образом, что регистры являются распределенными, но в MDIL коде сдвиги не были включены для одной или более библиотек и/или классов. В некоторых реализациях, компоновщик может конвертировать псевдо инструкции в инструкции собственного кода для поиска реализации или экземпляра обобщенного типа или метода.

[039] В некоторых реализациях, символические инструкции MDIL кода и сгенерированные соответствующие собственные инструкции являются соответствующими в отношении виртуального назначения сегментов метода и/или формата поля объекта. Например, MDIL код и исполняемый собственный код могут быть соответствующими по отношению к формату поля объекта, так что часть MDIL кода для доступа к объектам, которая включает в себя маркер поля, может быть использована, чтобы генерировать собственные инструкции (например, машинные инструкции) с использованием численных смещений поля. В некоторых реализациях, компоновщик может генерировать численное смещение из символической ссылки, чтобы генерировать одну или более исполняемых машинных инструкций. Например, символическая ссылка в MDIL коде преобразуется компоновщиком в конкретную ссылку поля таким образом, что конкретная ссылка поля включает в себя численный размер и/или смещение.

[040] В некоторых реализациях связывания MDIL кода, компоновщик может интерпретировать символическую ссылку, которая включает в себя маркер, для определения собственного кода, подлежащего генерации для символической ссылки. Например, для маркера поля, который ссылается на поле объекта, компоновщик может компоновать объект, как он может быть доступен на вычислительном устройстве, и определять соответствующее смещение для поля объекта для генерации исполняемых инструкций для вычислительного устройства. В некоторых реализациях связывания MDIL кода, компоновщик, который может связывать MDIL код, является более сложным, чем линкер. Например, при связывании MDIL кода, компоновщик может настраивать один или более переходов в результирующем собственном коде, так что переходы соответствуют их намеченным целям.

[041] В дополнение к собственному коду в собственном образе, по меньшей мере некоторый код для приложения может поддерживаться в IL форме. Код в IL форме дополнительно обрабатывается в течение “точно к нужному моменту” (динамической) компиляции или другой компиляции, прежде чем приложение запускается, или когда приложение запускается. Это может быть полезно, когда механизм среды выполнения, библиотека или другой ресурс часто меняются. Или это может быть полезно, когда значение типа или ссылка не могут быть разрешены в течение процесса связывания, в котором остающийся MDIL код связывается с библиотеками на вычислительном устройстве.

[042] В блоке 440, собственный образ сохраняется для использования при загрузке приложения для исполнения. Например, когда пользователь запускает приложение на вычислительном устройстве, собственный образ приложения загружается посредством механизма исполнения инструкций или среды выполнения, такого как CLR архитектуры .NET или другого механизма среды выполнения. В некоторых реализациях,

вычислительное устройство может включать MSIL код для приложения, который может компилироваться на вычислительном устройстве с использованием “точно к нужному моменту” компиляции, чтобы исполнять приложение, но вместо исполнения приложения с использованием MSIL кода, собственный образ для приложения загружается и исполняется, чтобы запустить приложение. В некоторых реализациях, когда собственный образ загружается, загружаются один или более образов или компоновок библиотек, которые используются кодом приложения как включенные в собственный образ. Например, собственный образ может быть загружен для исполнения с использованием механизма среды выполнения вычислительного устройства. Собственный образ для приложения может быть подписан компоновщиком перед его сохранением, чтобы показать, что приложение является доверительным.

[043] В блоке 450, вычислительное устройство обновляется по меньшей мере путем обновления наборов библиотек на вычислительном устройстве. Например, одна или более из библиотек в наборе библиотек вычислительного устройства могут быть обновлены, заменены или изменены. Это может нарушить работу приложений, которые были связаны с предыдущей версией набора библиотек. Таким образом, после того, как одна или более библиотек обновлены, заменены или изменены, одно или более из приложений на вычислительном устройстве могут быть обновлены, так что может быть сгенерирован обновленный собственный образ для соответствующих приложений. Обновленный собственный образ для приложения может быть сгенерирован путем связывания MDIL кода для приложения (которое сохранено в MDIL форме на вычислительном устройстве даже после установки) с по меньшей мере одной из набора одной или более библиотек, которые были обновлены. Повторное связывание MDIL кода для приложения с одной или более библиотеками на вычислительном устройстве, после обновления библиотек, может позволить собственному образу корректно ссылаться на одну или более библиотек, доступных на вычислительном устройстве после обновления устройства и/или обновления библиотек устройства. В то же время, полная повторная компиляция кода для приложения не требуется каждый раз, когда библиотека изменяется, поскольку MDIL код для приложения может быть использован повторно. Когда измененная библиотека используется многими приложениями на устройстве, повторное связывание с использованием ранее сохраненного MDIL кода для соответствующих приложений может значительно упростить процесс обновления собственных образов для соответствующих приложений.

[044] Например, предположим, что имеется семь приложений, каждое из которых зависит от библиотеки, которая была изменена. Для каждого из семи приложений, офлайн-компиляция в облаке (например, онлайн-провайдером) может сформировать действующий MDIL код для этого приложения с использованием компиляции, которая является более сложной, и/или “точно к нужному моменту” компиляции или компиляции на устройстве для целей обновления. Вместо того чтобы повторно компилировать все семь приложений при изменении библиотек, что будет отнимать много времени и может привести к менее эффективному собственному образу, соответствующие приложения повторно связываются с библиотеками на вычислительном устройстве, что намного быстрее, так как "тяжелый труд" по оптимизации компиляции был выполнен ранее в облаке.

[045] В некоторых реализациях, вычислительное устройство обновляется таким образом, что обновление включает в себя обновление механизма среды выполнения вычислительного устройства. Например, механизм среды выполнения вычислительного устройства, который исполняет одно или более установленных приложений с

использованием собственного образа для приложений, может быть обновлен, заменен или изменен. Механизмом среды выполнения может быть слой программного обеспечения, который в конечном итоге исполняет обновленное приложение так, что когда механизм среды выполнения изменяется, обновленный собственный образ для обновленного приложения может генерироваться компоновщиком, чтобы исполняться с использованием обновленного механизма среды выполнения, как доступного для компоновщика и на вычислительном устройстве. В некоторых реализациях, как механизм среды выполнения, так и библиотеки вычислительного устройства изменяются при обновлении устройства, и приложение может быть обновлено, так что оно может исполняться и/или работать должным образом с обновленным механизмом среды выполнения и обновленными библиотеками.

Примерный способ генерации установочного пакета для приложения, которое может быть предоставлено для загрузки

[046] На Фиг. 5 показан примерный способ 500 генерации установочного пакета для приложения, которое может быть предоставлено для загрузки. В различных реализациях, иллюстрируемые блоки способа на Фиг. 5 могут объединяться, разделяться на суб-блоки или опускаться. На Фиг. 5, принимается предварительный код для приложения. Например, разработчик приложения может послать код для приложения на предварительном языке и/или состоянии, и предварительный код для приложения может приниматься онлайн-провайдером, например, на онлайн-рынке. В некоторых реализациях, предварительный код для приложения является исходным кодом приложения. Например, исходный код может быть кодом, написанным на одном или более языках программирования, таких как C++, C#, Visual Basic®, J#, J++ или на других языках программирования. В некоторых реализациях, предварительный код приложения выполнен в промежуточном представлении, например, в IL. Например, исходный код для приложения может быть скомпилирован компилятором и/или генератором в промежуточное представление в IL. В некоторых реализациях, IL может представлять собой Microsoft Intermediate Language (MSIL). В некоторых реализациях, IL не включает собственные инструкции, которые являются машинно-зависимыми. Однако код в IL может быть скомпилирован в код в MDIL, который включает в себя собственные инструкции, которые являются машинно-зависимыми.

[047] В некоторых реализациях другая информация принимается для приложения, а также предварительный код для приложения. Например, могут быть получены файлы ресурсов для приложения, которые используются приложением, когда оно исполняется. Файлы ресурсов для приложений могут быть одним или более файлами, которые включают в себя аудио информацию, информацию музыки, информацию графики, видео информацию, другую медиа информацию, информацию баз данных, текстовую информацию и т.п. В некоторых реализациях, предварительный код может быть упакован и/или приниматься в одном или более сжатых файлов, таких как ZIP файл или XAP файл. Альтернативно, предварительный код не упакован и/или не принимается в одном или более сжатых файлов. В некоторых реализациях один или более файлов с MDIL кодом для приложения принимаются онлайн-провайдером, но MDIL файлы, принимаемые от разработчика, могут рассматриваться как недоверительные файлы, которые не генерировались онлайн-провайдером. Альтернативно, онлайн-провайдер не отбрасывает никакие MDIL файлы, принимаемые онлайн-провайдером. В некоторых реализациях, предварительный код может быть получен в одной или более компоновках. Например, предварительный код (такой как код в MSIL, включенный в компоновку) может быть скомпилирован в код, который находится на двоичном уровне. Компоновка,

которая включает в себя MSIL код, может быть MSIL компоновкой. В некоторых реализациях, MSIL компоновка может содержать один или более файлов.

[048] В блоке 520, предварительный код приложения компилируется для получения MDIL кода, который является кодом на промежуточном языке, который является машинно-зависимым. Как правило, решения или ссылки внутри IL файла разрешаются в процессе компиляции, но решения или ссылки на библиотеки, внешний источник, внешний тип и т.д. остаются неразрешенными. MDIL файл, таким образом, является смесью (a) инструкций компоновки, собственного кода или других зависимых от устройства инструкций и (b) инструкций псевдо кода или других независимых от устройства инструкций, которые должны быть разрешены позже. Таким образом, MDIL код, скомпилированный из предварительного кода, может быть на промежуточном языке, который является языком более низкого уровня, чем язык предварительного кода. Например, предварительный код является кодом на промежуточном языке, который находится на более высоком уровне (дальше от машинного кода), и MDIL код находится на более низком уровне, то есть ближе к машинному коду. Тем не менее, MDIL код обычно не является непосредственно исполняемым без связывания, так как MDIL код включает символические ссылки, которые не являются собственным кодом. Вместо этого, MDIL код может конвертироваться в исполняемый собственный код компоновщиком. В некоторых реализациях, предварительный код для приложения не является непосредственно исполняемым, но может быть скомпилирован в исполняемый код при компиляции с помощью “точно к нужному моменту” (динамического) компилятора. Например, код для приложения в MSIL может динамически компилироваться для запуска приложения из MSIL кода. В некоторых реализациях, код для приложения в MSIL может быть предоставлен компилятору для генерации MDIL кода для приложения. Как правило, MSIL код не включает собственные инструкции, а также не включает в себя псевдо инструкции, включенные в MDIL код, которые должны разрешаться и/или связываться компоновщиком, чтобы генерировать собственный код для приложения.

[049] MDIL код может быть включен в один или более MDIL файлов. В некоторых реализациях, MDIL файл может иметь имя и/или расширение файла такое, что ".mdil" входит в имя и/или расширение MDIL файла. В других реализациях, MDIL файл может иметь другое соглашение об именах, так что ".mdil" не входит в имя и/или расширение MDIL файла.

[050] В некоторых реализациях, когда предварительный код скомпилирован, в дополнение к псевдо инструкциям, компилятор формирует одну или более компоновок для MDIL кода. Например, MDIL код, включенный в компоновку, может быть скомпилирован в код, который находится на двоичном уровне. Компоновка, которая включает в себя MDIL код, может быть MDIL компоновкой. В некоторых реализациях, MDIL компоновка может содержать один или более файлов.

[051] Компиляция предварительного кода в MDIL код для приложения, прежде чем код приложения будет принят на целевом вычислительном устройстве (на котором приложение должно выполняться), может обеспечить различные преимущества. Например, время компилирования может быть исключено из времени, необходимого для установки приложения на целевом вычислительном устройстве. Исключение компиляции из установки приложения на целевом устройстве также может позволить оптимизации по времени компиляции кода, подлежащие выполнению в облаке, не требуя добавления этого времени к времени, которое требуется для установки приложения на целевом устройстве. В частности, это может помочь избежать затрат

времени повторной компиляции на целевом вычислительном устройстве для множества приложений, которые зависят от библиотеки, всякий раз, когда библиотека изменяется, и/или когда механизм среды выполнения изменяется.

[052] В некоторых реализациях компиляция предварительного кода может сформировать один или более файлов собственного кода для приложения, которые включают в себя собственный код приложения и не включают MDIL код.

[053] В некоторых реализациях, для каждого файла MSIL компоновки, принятого для приложения, соответствующий MDIL файл генерируется компилятором. Предварительный код может быть скомпилирован, чтобы генерировать MDIL код, который зависит от набора инструкций для первого процессора (для первого типа целевого вычислительного устройства), и другой MDIL код может быть сгенерирован, который зависит от набора инструкций для второго процессора (для второго типа целевого вычислительного устройства). Например, компиляция предварительного кода для приложения может сформировать MDIL код, который может быть связан, чтобы запускаться на устройстве, которое имеет первый процессор, и сформировать другой MDIL код для приложения, который может быть связан, чтобы запускаться на другом устройстве с другим вторым процессором. Например, первый процессор может быть процессором ARM, а второй процессор может быть процессором x86.

[054] В блоке 530, MDIL код приложения подписывается как доверительный. Например, файлы MDIL кода, сгенерированные из предварительного кода, могут быть подписаны ключом издателя, чтобы указать, что подписанные файлы включают доверительный код. В некоторых реализациях, собственный код, генерируемый путем компиляции предварительного кода, может быть подписан как доверительный. Например, файлы собственного кода, которые генерируются компилятором, могут быть подписаны с помощью ключа издателя, чтобы указать, что подписанные файлы включают доверительный код.

[055] В блоке 540 генерируется установочный пакет, который включает в себя MDIL код. Например, MDIL код может быть упакован в контейнер, такой как один или более файлов, которые можно загрузить для установки приложения. В одной реализации MDIL код в одном или более файлах MDIL кода упакован в один или более сжатых файлов, таких как ZIP файл или XAP файл. Сжатый установочный пакет может снизить количество данных, которые должны загружаться для приложения от онлайн-провайдера. Альтернативно, установочный пакет может быть не в сжатой форме. Различные типы MDIL кода (адаптированные для различных типов процессоров) могут быть включены в различные установочные пакеты для различных типов вычислительных устройств. Установочные пакеты для разных типов вычислительных устройств могут также включать в себя версию с не-MDIL кодом (например, MSIL код в XAP файле) в целях обратной совместимости.

[056] В некоторых реализациях, установочный пакет может содержать другую информацию для приложения, такую как файлы ресурсов для приложения. Например, могут приниматься файлы ресурсов для приложения, которые могут быть использованы приложением, когда оно исполняется. Файлы ресурсов для приложений могут быть одним или более файлами, которые включают звуковую информацию, информацию музыки, информацию графики, видео информацию, другую медиа информацию, информацию базы данных, текстовую информацию и т.п. Установочный пакет может также включать в себя список файлов, которые включены в установочный пакет. Список файлов может содержать информацию о файлах, включенных в установочный пакет. Например, установочный пакет может включать в себя файл, который включает в себя

информацию, организованную с использованием расширяемого языка разметки (XML). XML-файл может содержать список файлов, которые включены в установочный пакет. Для каждого из файлов в списке, XML-файл может содержать информацию, связанную с указанным файлом. Например, XML-файл может перечислять каждый из файлов MDIL кода, включенного в установочный пакет, и указывать, что каждый из файлов MDIL кода может быть связан. XML-файл также может указывать одну или более библиотек, подлежащих связыванию с кодом соответствующих перечисленных файлов MDIL кода.

[057] В некоторых реализациях, сгенерированный установочный пакет подписывается как доверительный. Например, установочный пакет может быть подписан с помощью ключа издателя, чтобы указывать, что он включает в себя код и/или другую информацию, которая является доверительной. В некоторых реализациях, установочный пакет может быть зашифрован.

[058] В блоке 550 установочный пакет предоставляется для загрузки. Например, онлайн-провайдер и/или партнер онлайн-провайдера может хранить установочный пакет для загрузки. Онлайн-провайдером может быть онлайн-рынок, из которого одно или более приложений могут быть загружены для установки на одно или более вычислительных устройств. Приложение может быть включено в каталог приложений, которые доступны для загрузки из онлайн-рынка. Например, приложение может быть загружено вычислительным устройством, используя Интернет и/или облако. Приложение может быть загружено таким образом, что установочный пакет для приложения загружается. В некоторых реализациях, приложение может быть загружено на вычислительное устройство после того, как приложение было куплено и авторизовано для загрузки в вычислительное устройство. В некоторых реализациях, приложение может быть загружено без приобретения. Например, приложение может быть предоставлено для бесплатной загрузки.

ПРИМЕРНАЯ СИСТЕМА ДЛЯ ГЕНЕРАЦИИ ОБНОВЛЕННЫХ СОБСТВЕННЫХ ОБРАЗОВ ДЛЯ УСТАНОВЛЕННЫХ ПРИЛОЖЕНИЙ

[059] На Фиг. 6 показана схема вычислительного устройства 600, которое может генерировать один или более обновленных собственных образов 605 для одного или более установленных приложений. На Фиг. 6, вычислительное устройство включает в себя один или более сгенерированных собственных образов 610 для одного или более установленных приложений на вычислительном устройстве. Вычислительное устройство 600 включает в себя обновленные библиотеки 620, которые включают в себя одну или более библиотек, которые были обновлены после связывания с соответствующим MDIL кодом приложений, чтобы генерировать один или более собственных образов 610. Поскольку по меньшей мере одна из библиотек, которые были связаны с одним или более собственными образами 610, была обновлена, вычислительное устройство 600 обновляет приложения, установленные на нем, путем генерации одного или более обновленных собственных образов 605 для установленных приложений. Вычислительное устройство также включает в себя обновленный механизм 670 среды выполнения.

[060] При формировании одного или более обновленных собственных образов 605 для установленных приложений, компоновщик 625 генерирует обновленный собственный образ для каждого из одного или более установленных приложений. Как показано на Фиг. 6, для каждого установленного приложения, которое имеет собственный образ, который был связан с по меньшей мере одной библиотекой, которая была изменена и/или обновлена, менеджер 630 пакетов предоставляет компоновщику 625 MDIL файлы соответствующего приложения для повторного связывания для

генерации обновленного собственного образа для обновляемого приложения, например, обновленного собственного образа 635. Обновленные собственные образы могут быть сгенерированы компоновщиком 625 таким образом, что собственные образы являются исполняемыми с использованием обновленного механизма 670 среды выполнения.

[061] Для генерации обновленного собственного образа 635, для каждого MDIL файла для обновляемого приложения, компоновщику 625 предоставляется соответствующий MDIL файл приложения, такой как MDIL файл 640, и одна или более библиотек, таких как одна или более библиотек 645, которые должны быть связаны с MDIL файлом. Одна или более библиотек, предоставляемых для связывания с кодом MDIL файла, могут включать в себя по меньшей мере одну библиотеку, которая была обновлена и включена в набор обновленных библиотек 620. В некоторых реализациях одна или более библиотек, связанных с кодом MDIL файла обновляемого приложения, не были обновлены.

[062] После того как сгенерирован, обновленный собственный образ для приложения, например, обновленный собственный образ 635 хранится в виде по меньшей мере одного из одного или более собственных обновленных образов 605 для установленных приложений на вычислительном устройстве. Обновленный собственный образ может быть подписан доверительным кодом, как показано в блоке 650, прежде чем обновленный собственный образ будет использован для исполнения приложения. В блоке 655, вычислительное устройство загружает сгенерированный обновленный собственный образ для приложения, чтобы исполнять приложение, вместо сгенерированного ранее собственного образа для приложения.

[063] В дополнение к обработке обновления библиотек на устройстве, устройство может обновить приложение, когда приложение изменились (например, MDIL код приложения изменился). В этом случае, устройство обновляет приложение путем генерации обновленного собственного образа для приложения, включая связывание новой части MDIL кода с одной или более библиотеками (без изменений) на устройстве. Обновленный собственный образ для приложения может генерироваться компоновщиком, так что собственные образы являются выполняемыми с использованием обновленного механизма среды выполнения.

[064] Как показано в блоке 660, сгенерированный ранее собственный образ может быть временно сохранен в течение генерации обновленного собственного образа и/или обновления вычислительного устройства 600 до тех пор, пока обновление приложения и/или вычислительного устройства 600 не будет подтверждено как успешное. Если обновление на вычислительном устройстве и/или приложении безуспешно, то ранее сгенерированный собственный образ может быть восстановлен из временного хранилища и снова использован в качестве собственного образа, загружаемого с помощью вычислительного устройства для выполнения приложения.

ПРИМЕРНЫЙ СПОСОБ ОБНОВЛЕНИЯ ПРИЛОЖЕНИЯ ПОСРЕДСТВОМ ГЕНЕРАЦИИ СОБСТВЕННОГО КОДА НА MDIL

[065] На Фиг. 7 показан примерный способ 700 для обновления приложения путем генерации собственного образа для приложения из кода в MDIL. В различных реализациях проиллюстрированные блоки способа на Фиг. 7 могут объединены, разделены на суб-блоки или опущены. На Фиг. 7, набор из одной или более библиотек обновляется на вычислительном устройстве, как показано в блоке 710. Например, часть или все из одной или более библиотек на вычислительном устройстве изменяются, модифицируются или обновляются иным образом. В некоторых реализациях, набор из одной или более библиотек обновляется, и механизм среды выполнения устройства

обновляется как часть обновления вычислительного устройства. В некоторых реализациях, для одного или более приложений на вычислительном устройстве, по меньшей мере одна библиотека из одной или более библиотек, которые были связаны с MDIL кодом для соответствующего приложения (для создания собственного образа для соответствующего приложения во время установки), изменены, модифицированы или обновлены иным образом. В некоторых реализациях, библиотеки обновлены и/или изменены таким образом, что код собственного образа для приложения, который связан с библиотекой, уже не соответствует должным образом по меньшей мере одному смещению для библиотеки. Например, смещение для по меньшей мере одного поля объекта изменилось в библиотеке, и т.п. Набор библиотек на вычислительном устройстве может быть обновлен как часть обновления операционной системы. Или, библиотеки на вычислительном устройстве могут быть обновлены как часть обновления в архитектуре и/или обновления механизма исполнения инструкций. Например, вычислительное устройство может использовать архитектуру для исполнения управляемого приложения, такую как архитектуру .NET, и библиотеки, которые являются частью архитектуры, изменены и/или обновлены, и/или механизм среды выполнения, который является частью архитектуры, изменен и/или обновлен. В архитектуре .NET, библиотеки базовых классов могут быть изменены или обновлены до отличающейся и/или более новой версии библиотек архитектуры. В некоторых случаях не все библиотеки на вычислительном устройстве обновляются и/или изменяются. В других случаях, все библиотеки на вычислительном устройстве обновляются и/или изменяется.

[066] В блоке 720, вычислительное устройство генерирует обновленный собственный образ для приложения. Например, приложение, которое было установлено на вычислительном устройстве, путем генерации собственного образа из MDIL кода для приложения, может быть обновлено таким образом, что новый собственный образ (например, обновленный собственный образ) генерируется для приложения путем связывания MDIL кода для приложения с одной или более библиотеками после того, как по меньшей мере одна из одной или более библиотек была обновлена на вычислительном устройстве. Генерация обновленного собственного образа может быть аналогична генерации собственного образа, когда приложение было первоначально установлено. Например, обновленный собственный образ может быть сгенерирован для приложения с использованием по меньшей мере одной из обновленных библиотек вычислительного устройства, как описано выше, в процессе установки. Таким образом, генерирование обновленного собственного образа происходит после того, как первый собственный образ для приложения уже был сгенерирован для приложения, и одна или более библиотек, связанных при генерации первого собственного образа, были обновлены и/или изменены на вычислительном устройстве. В некоторых реализациях, когда механизм среды выполнения был обновлен вместе с библиотеками вычислительного устройства, связывание MDIL кода обновляемого приложения может быть сделано таким образом, что сгенерированный собственный образ является исполняемым с использованием обновленного механизма среды выполнения и доступных обновленных библиотек на вычислительном устройстве.

[067] Или, обновленный собственный образ может генерироваться, когда MDIL код для приложения обновлен, но библиотеки и/или механизмы среды выполнения не изменились. Например, вычислительное устройство принимает новый MDIL код для приложения как часть обновления приложения. Принятый новый MDIL код для приложения может быть установлен, как описано здесь, например, с использованием

способа 200, описанного выше. Например, вычислительное устройство обновляет приложение путем генерации обновленного собственного образа для приложения, включая связывание части нового MDIL кода с одной или более библиотеками (без изменений) на вычислительном устройстве. Собственный код, сгенерированный для приложения компоновщиком, может быть сгенерирован так, что он является исполняемым с использованием механизма среды выполнения, доступного на устройстве для выполнения приложения.

[068] В некоторых реализациях, для обновляемого приложения, MDIL код для приложения, сохраненного на вычислительном устройстве для приложения, может быть связан с одной или более библиотек устройства. Например, компоновщик может связывать MDIL код с одной или более библиотеками, доступными на вычислительном устройстве для генерации исполняемого собственного образа, который обновляется для приложения. Связывание разрешает MDIL код и генерирует собственный код для приложения, который включен в обновленный собственный образ для приложения. В некоторых реализациях, при генерации обновленного собственного образа, для каждого MDIL файла в наборе MDIL файлов для приложения, соответствующие MDIL файлы и одна или более библиотек, которые должны быть связаны с MDIL кодом файла, предоставляются компоновщику. Компоновщик связывает соответствующие MDIL файлы с библиотеками, предоставленными компоновщику, чтобы генерировать обновленный собственный образ для приложения. Обновленный собственный образ может включать в себя собственный код, такой как машинный код, который является исполняемым процессором вычислительного устройства, и обновленный собственный образ, как правило, не включает в себя несвязанные псевдо инструкции, как в MDIL коде. В некоторых реализациях, сгенерированные обновленные собственные образы могут быть подписаны как доверительный код.

[069] В некоторых реализациях, обновленный собственный образ заменяет ранее сгенерированный собственный образ для приложения в качестве собственного образа, загружаемого для выполнения приложения. Обновленный собственный образ загружается для исполнения приложения, так как обновленный собственный образ включает в себя собственный код, который является исполняемым с использованием по меньшей мере одной из обновленных библиотек вычислительного устройства. Сгенерированный ранее собственный образ может быть неустойчивым и/или ненадежным, так что он не работает должным образом или как ожидалось, потому что он может неправильно ссылаться на более раннюю версию одной из библиотек, доступных на вычислительном устройстве.

[070] В некоторых реализациях, когда генерируется обновленный собственный образ, ранее сгенерированный собственный образ сохраняется и/или резервируется временно до тех пор, пока приложение и/или обновление на вычислительном устройстве не будет подтверждено как успешное. Например, обновление для вычислительного устройства, которое обновляет библиотеки вычислительного устройства, может вызвать регенерацию и/или обновление собственных образов для приложений. Если обновление на вычислительном устройстве или обновление собственного образа для приложения безуспешно, то безуспешное обновление может выполнить откат назад, и ранее сгенерированный собственный образ приложения, который сохранен, может быть использован в качестве собственного образа приложения, когда оно исполняется. В некоторых реализациях, данные приложения также сохраняются. Например, пользователь приложения может генерировать пользовательские данные и/или данные приложения, прежде чем приложение будет обновлено, и, после того, как приложение

обновлено, пользовательские данные и/или данные приложения могут быть сохранены, так что генерация обновленного собственного образа не изменяет пользовательские данные и/или данные приложения из предыдущей версии приложения.

[071] В блоке 730, вычислительное устройство генерирует обновленный собственный образ для одного или более других приложений, установленных на устройстве. Например, для каждого дополнительного приложения, установленного на вычислительном устройстве, которое имеет собственный образ, который был создан путем связывания MDIL кода с одной или более библиотеками, которые были обновлены, обновленный собственный образ для соответствующего приложения генерируется путем связывания MDIL кода с одним или более обновленных библиотек. Пунктирные линии, как показано блоком 730, указывают, что блок 730 может быть опущен из способа или входит в способ в различных реализациях способа 700. В некоторых реализациях, после того как одна или более библиотек в устройстве обновляются (например, в обновлении архитектуры), приложения устройства могут автоматически обновляться, и обновленные собственные образы для приложений могут генерироваться, так что приложения исполняются должным образом с использованием обновленных библиотек. Если установленное приложение не использует общие библиотеки, которые были обновлены и/или изменены, приложение не требуется обновлять, и обновленный собственный образ не требуется генерировать для приложения, так как ранее сгенерированный собственный образ для приложения может функционировать должным образом с библиотеками, доступными на устройстве. В некоторых реализациях, если механизм среды выполнения был обновлен вместе с библиотеками вычислительного устройства, обновленные собственные образы для приложений могут генерироваться таким образом, что приложение является исполняемым на вычислительном устройстве с использованием обновленного механизма среды выполнения и обновленных библиотек, доступных на вычислительном устройстве.

[072] В блоке 740, обновленный собственный образ сохраняется для использования при загрузке приложения для исполнения приложения на вычислительном устройстве. Например, загружается обновленный собственный образа, вместо собственного образа приложения, сгенерированного перед генерацией обновленного собственном образа. Обновленный собственный образ загружается для выполнения приложения, потому что обновленный собственный образ может правильно ссылаться на библиотеки, которые он использует. Путем генерации обновленных собственных образов, путем повторного связывания уже загруженного MDIL кода для приложения, вычислительное устройство, которое использует собственные образы для запуска приложений, может быть обновлено без необходимости загружать обновленную версию приложения, без повторной установки приложения из вновь загруженной информации для приложений и без перекомпиляции приложения.

ПРИМЕРНОЕ МОБИЛЬНОЕ УСТРОЙСТВО

[073] Фиг. 8 представляет собой структурную схему, изображающую примерное мобильное устройство 800, включающее в себя множество опциональных аппаратных и программных компонентов, как показано в целом позицией 802. В общем, компонент 802 в мобильном устройстве может взаимодействовать с другим компонентом, хотя и не все соединения показаны для простоты иллюстрации. Мобильное устройство может быть любым из различных вычислительных устройств (таким как сотовый телефон, смартфон, планшетный компьютер, карманный компьютер, персональный цифровой помощник (PDA) и т.д.) и может обеспечивать беспроводную двухстороннюю связь с одной или более сетей 804 мобильной связи, таких как сотовая или спутниковая сеть.

[074] Показанное мобильное устройство 800 может включать в себя контроллер или процессор 810 (например, процессор сигналов, микропроцессор, ASIC или другие логические схемы управления и обработки) для выполнения таких задач, как кодирование сигнала, обработка данных, обработка ввода/вывода, управление питанием и/или другие функции. Операционная система 812 может управлять распределением и использованием компонентов 802 и поддерживать одну или более прикладных программ 814 и/или одну или более программ 815 программного обеспечения, таких как те, которые могут реализовывать одну или более технологий, описанных в данном документе, для генерации собственного кода из IL кода на устройстве для приложений. Прикладные программы могут включать в себя общие мобильные вычислительные приложения и программное обеспечение (например, приложения электронной почты, календари, менеджеры контактов, веб-браузеры, приложения обмена сообщениями, механизм среды выполнения), или любые другие вычислительные приложения. Функциональность 813 для доступа к магазину приложений, онлайн-рынку или онлайн-провайдеру также может быть использована для приобретения и обновления кода и/или другой информации для прикладных программ 814.

[075] Показанное мобильное устройство 800 может включать в себя память 820. Память 820 может включать в себя несъемную память 822 и/или съемную память 824. Несъемная память 822 может включать в себя RAM (ОЗУ), ROM (ПЗУ), флэш-память, жесткий диск или другие хорошо известные технологии хранения данных. Съемная память 824 может включать в себя флэш-память или карту модуля идентификации абонента (SIM-карту), что хорошо известно в системах связи стандарта GSM, или другие известные технологии хранения памяти, такие как "смарт-карты". Память 820 может быть использована для хранения данных и/или кода для запуска операционной системы 812 и приложений 814 и 815. Примерные данные могут включать в себя веб-страницы, текст, изображения, звуковые файлы, видео данные или другие наборы данных, подлежащие отправке и/или принимаемые из одного или более сетевых серверов или других устройств через одну или более проводных или беспроводных сетей. Память 820 может быть использована для хранения идентификатора абонента, такого как Международный идентификационный номер мобильного абонента (IMSI), и идентификатора оборудования, такого как Международный идентификатор мобильного абонента (IMEI). Такие идентификаторы могут быть переданы на сетевой сервер для идентификации пользователей и оборудования.

[076] Мобильное устройство 800 может поддерживать одно или более устройств 830 ввода, таких как воспринимающий касание (сенсорный) экран 832, микрофон 834, камеру 836, физическую клавиатуру 838 и/или трекбол 840 и одно или более устройств 850 вывода, например, громкоговоритель 852 и дисплей 854. Другие возможные устройства вывода (не показаны) могут включать в себя пьезоэлектрические или другие устройства сенсорного вывода. Некоторые устройства могут обслуживать более одной функции ввода/вывода. Например, воспринимающий касание экран 832 и дисплей 854 могут быть объединены в одном устройстве ввода/вывода. Устройства 830 ввода могут включать в себя естественный пользовательский интерфейс (NUI). NUI является любой технологией интерфейса, которая позволяет пользователю взаимодействовать с устройством "естественным" образом, свободным от искусственных ограничений, налагаемых устройствами ввода, такими как мыши, клавиатуры, пульты дистанционного управления и тому подобное. Примеры методов NUI включают такие, которые основываются на распознавании речи, распознавании касания и пера, распознавании жестов как на экране, так и рядом с экраном, воздушных жестов, отслеживании головы

и глаз, голосе и речи, зрении, осязании, жестах и машинном интеллекте. Другие примеры NUI включают в себя обнаружение жеста движения с использованием акселерометров/гироскопов, распознавание лиц, 3D-дисплеи, отслеживание головы, глаз и взгляда, системы иммерсивной дополненной реальности и системы виртуальной реальности, все из которых обеспечивают более естественный интерфейс, а также технологии для восприятия активности мозга с помощью электродов восприятия электрического поля (EEG и связанные с ними методы). Таким образом, в одном конкретном примере, операционная система 812 или приложения 814 могут содержать программное обеспечение распознавания речи как часть голосового пользовательского интерфейса, который позволяет пользователю управлять устройством 800 с помощью голосовых команд. Кроме того, устройство 800 может включать в себя устройства ввода и программное обеспечение, которое обеспечивает возможность взаимодействия с пользователем с помощью пространственных жестов пользователя, например, обнаружение и интерпретация жестов для обеспечения ввода в игровое приложение или в другое приложение.

[077] Беспроводной модем 860 может быть соединен с антенной (не показана) и может поддерживать двустороннюю связь между процессором 810 и внешними устройствами, как хорошо известно в данной области техники. Модем 860 показан в общем и может включать в себя сотовый модем для связи с мобильной сетью 804 связи и/или другими радио-модемами (например, Bluetooth-864 или WiFi-862). Беспроводной модем 860 обычно сконфигурирован для связи с одной или более сотовыми сетями, такими как сеть GSM для передачи данных и голосовой связи в пределах одной сотовой сети, между сотовыми сетями или между мобильным устройством и коммутируемой телефонной сетью общего пользования (PSTN).

[078] Мобильное устройство может дополнительно включать в себя по меньшей мере один порт 880 ввода/вывода, блок 882 питания, приемник 884 спутниковой навигационной системы, например, приемник системы глобального позиционирования (GPS), акселерометр 886 и/или физический коннектор 890, который может быть USB-портом, 1394 IEEE (Fire Wire) портом и/или RS-232 портом. Изображенные компоненты 802 не обязательно все должны быть включены, и какие-либо компоненты могут быть удалены, а другие компоненты могут быть добавлены.

ПРИМЕРНАЯ СРЕДА РЕАЛИЗАЦИИ

[079] Фиг. 9 иллюстрирует обобщенный пример подходящей среды 900 реализации, в которой могут быть реализованы описанные варианты осуществления, методы и технологии.

[080] В примерной среде 900, различные виды услуг (например, вычислительные услуги) предоставляются облаком 910. Например, облако 910 может включать в себя совокупность вычислительных устройств, которые могут быть расположены централизованно или распределенным образом, которые обеспечивают облачные услуги различным типам пользователей и устройств, подключенных через сеть, такую как Интернет. Среда 900 реализации может использоваться различными способами для выполнения вычислительных задач. Например, некоторые задачи (например, обработка пользовательского ввода и представление пользовательского интерфейса) могут быть выполнены на локальных вычислительных устройствах (например, подключенных устройствах 930, 940, 950), тогда как другие задачи (например, хранение данных, которые будут использоваться при последующей обработке) могут быть выполнены в облаке 910.

[081] В примерной среде 900, облако 910 предоставляет услуги для подключенных

устройств 930, 940, 950 с различными экранными возможностями. Подключенное устройство 930 представляет собой устройство с компьютерным экраном 935 (например, экраном среднего размера). Например, подключенное устройство 930 может быть персональным компьютером, таким как настольный компьютер, портативный компьютер, ноутбук, нетбук и т.п. Подключенное устройство 940 представляет собой устройство с экраном 945 мобильного устройства (например, экраном малого размера). Например, подключенное устройство 940 может представлять собой мобильный телефон, смартфон, персональный цифровой помощник, планшетный компьютер и т.п. Подключенное устройство 950 представляет собой устройство с большим экраном 955. Например, подключенное устройство 950 может быть телевизионным экраном (например, смарт-телевизор) или другим устройством, подключенным к телевизору (например, приставка или игровая консоль), и т.п. Одно или более из подключенных устройств 930, 940 и 950 могут включать в себя возможности сенсорного экрана. Сенсорные экраны могут принимать ввод по-разному. Например, емкостные сенсорные экраны обнаруживают сенсорный ввод, когда объект (например, кончик пальца или стилус) искажает или прерывает электрический ток, проходящий через поверхность. В качестве другого примера, сенсорные экраны могут использовать оптические датчики для обнаружения сенсорного ввода, когда лучи от оптических датчиков прерываются. Физический контакт с поверхностью экрана не является необходимым, чтобы ввод обнаруживался некоторыми сенсорными экранами. Устройства без экранных возможностей также могут использоваться в примерной среде 900. Например, облако 910 может предоставлять услуги для одного или более компьютеров (например, серверных компьютеров) без дисплея.

[082] Услуги могут быть предоставлены облаком 910 через провайдеров 920 услуг или с помощью других провайдеров онлайн-услуг (не показано). Например, облачные услуги могут быть настроены с учетом размера экрана, возможностей дисплея и/или возможностей сенсорного экрана конкретного подключенного устройства (например, подключенных устройств 930, 940, 950).

[083] В примерной среде 900, облако 910 обеспечивает технологии и решения, описанные в данном документе, для различных подключенных устройств 930, 940, 950, используя, по меньшей мере частично, провайдеров 920 услуг и одного или более онлайн-провайдеров 925. Например, провайдеры 920 услуг могут обеспечивать централизованное решение для различных облачных услуг. Провайдеры 920 услуг могут управлять подписками на услуги для пользователей и/или устройств (например, для подключенных устройств 930, 940, 950 и/или их соответствующих пользователей). Облако 910 может обеспечивать ресурсы для загрузки, передачу или прием одного или более установочных пакетов для одного или более приложений, как описано здесь. Например, промежуточный код языка для приложения может быть компилирован в облаке 910 по меньшей мере одним из онлайн-провайдеров 925. Как показано в блоке 965, собственный образ генерируется для приложения подключенным устройством 930 из IL кода, загруженного из облака 910.

ПРИМЕРНАЯ ВЫЧИСЛИТЕЛЬНАЯ СРЕДА

[084] Фиг. 10 изображает обобщенный пример подходящей вычислительной среды 1000, в которой могут быть реализованы описанные инновации. Вычислительная среда 1000 не предназначена для каких-либо ограничений в отношении объема использования или функциональности, так как инновации могут быть реализованы в различных вычислительных системах общего назначения или специального назначения. Например, вычислительная среда 1000 может быть любым из различных вычислительных устройств

(таких, как настольный компьютер, портативный компьютер, серверный компьютер, планшетный компьютер, медиаплеер, игровая система, мобильное устройство и т.д.)

[085] Как показано на Фиг. 10, вычислительная среда 1000 включает в себя один или более блоков 1010, 1015 обработки и памяти 1020, 1025. На Фиг. 10, эта базовая конфигурация 1030 включена в пределы пунктирной линии. Блоки 1010, 1015 обработки выполняют компьютерно-исполняемые инструкции. Блок обработки может быть центральным процессорным блоком (CPU) общего назначения, процессором на специализированной интегральной схеме (ASIC) или любым другим типом процессора. В многопроцессорной системе, множество блоков обработки выполняют компьютерно-исполняемые инструкции, чтобы повысить вычислительную мощность. Например, на Фиг. 10 показан центральный процессорный блок 1010, а также блок обработки графики или со-процессорный блок 1015. Материальные блоки памяти 1020, 1025 могут представлять собой энергозависимую память (например, регистры, кэш, RAM), энергонезависимую память (например, ROM, EEPROM, флэш-память и т.д.) или некоторую комбинацию обоих, доступную посредством блока(ов) обработки. Память 1020, 1025 хранит программное обеспечение 1080, реализующее одну или более описанных здесь инноваций, в форме компьютерно-исполняемых инструкций, пригодных для исполнения блоком(ами) обработки.

[086] Вычислительная система может иметь дополнительные особенности. Например, вычислительная среда 1000 включает в себя память 1040, одно или более устройств 1050 ввода, одно или более устройств 1060 вывода и одно или более коммуникационных соединений 1070. Механизм межсоединений (не показан), такой как шина, контроллер или сеть, взаимосвязывает компоненты вычислительной среды 1000. Как правило, программное обеспечение операционной системы (не показано) обеспечивает операционную среду для другого программного обеспечения, выполняющегося в вычислительной среде 1000, и координирует действия компонентов вычислительной среды 1000.

[087] Материальная память 1040 может быть съемной или несъемной и включает в себя магнитные диски, магнитные ленты или кассеты, CD-ROM, DVD, или любой другой носитель, который может быть использован для хранения информации не-транзиторным (не-временным) образом, и к которому можно получить доступ в вычислительной среде 1000. Память 1040 хранит инструкции для программного обеспечения 1080, реализующего одно или более инноваций, описанных здесь, например, генерацию собственного кода из IL кода для одного или более приложений.

[088] Устройство 1050 ввода может быть устройством сенсорного ввода, таким как клавиатура, мышь, перо или шаровой манипулятор, устройство голосового ввода, сканирующее устройство или другое устройство, которое обеспечивает ввод в вычислительную среду 1000. Для кодирования видео, устройством 1050 ввода может быть камера, видео карта, карта TV-тюнера или аналогичное устройство, которое принимает ввод видео в аналоговой или цифровой форме, либо CD-ROM или CD-RW, который считывает образцы видео в вычислительной среде 1000. Устройством 1060 вывода может быть дисплей, принтер, динамик, устройство для записи на диски или другое устройство, которое обеспечивает вывод из вычислительной среды 1000.

[089] Коммуникационное соединение 1070 позволяет устанавливать связь по среде связи с другим вычислительным объектом. Среда связи передает информацию, такую как компьютерно выполняемые инструкции, аудио или видео ввод или вывод или другие данные в виде модулированного сигнала данных. Модулированный сигнал данных является сигналом, в котором одна или более из его характеристик установлены или

изменены таким образом, чтобы кодировать информацию в сигнале. В качестве примера, а не ограничения, среда связи может использовать электрический, оптический, RF или другой носитель.

[090] Хотя операции некоторых из описанных способов описаны, в частности, в последовательном порядке для удобства изложения, следует понимать, что этот способ описания включает перегруппировку, если конкретный порядок не требуется конкретным описанием, изложенным ниже. Например, операции, описанные последовательно, в некоторых случаях могут быть переупорядочены или могут выполняться одновременно. Кроме того, для простоты, прилагаемые чертежи могут не показывать различные способы, в которых описанные способы могут быть использованы в сочетании с другими способами.

[091] Любые из описанных способов могут быть реализованы в виде исполняемых компьютером инструкций, хранящихся на одном или более считываемых компьютером носителях (например, не-временных считываемых компьютером носителях, таких как один или более оптических медиа дисков, компонентах энергозависимой памяти (такие как DRAM или SRAM), или компонентах энергонезависимой памяти (например, флэш-памяти или жестких дисков)) и выполняются на компьютере (например, любом коммерчески доступном компьютере, включая смартфоны или другие мобильные устройств, которые включают в себя вычислительные аппаратные средства). Как легко понять, термин “считываемые компьютером носители” не включает в себя коммуникационные соединения, такие как модулированные сигналы данных. Любые исполняемые компьютером инструкции для реализации раскрытых методов, а также любые данные, создаваемые и используемые в ходе реализации раскрытых вариантов осуществления, могут быть сохранены на одном или более считываемых компьютером носителях (например, не-временных считываемых компьютером носителях, которые исключают распространяющиеся сигналы). Выполняемые компьютером инструкции могут быть частью, например, приложением специализированного программного обеспечения или приложением программного обеспечения, которое доступно или загружается через веб-браузер или другое приложение программного обеспечения (например, удаленное вычислительное приложение). Такое программное обеспечение может исполняться, например, на одном локальном компьютере (например, любой подходящем коммерчески доступном компьютере) или в сетевой среде (например, через Интернет, глобальную сеть, локальную сеть, сеть клиент-сервер (такую как облачная вычислительная сеть) или другая такая сеть) с использованием одного или более сетевых компьютеров.

[092] Для ясности, описаны только определенные выбранные аспекты основанной на программном обеспечении реализации. Другие детали, которые хорошо известны в данной области техники, опущены. Например, должно быть понятно, что описанная выше технология не ограничивается каким-либо конкретным языком или компьютерной программой. Например, раскрытая технология может быть реализована с помощью программного обеспечения, написанного на C++, C#, J++, Java, Perl, JavaScript, Adobe Flash или любом другом подходящем языке программирования. Кроме того, раскрытая технология не ограничивается каким-либо конкретным компьютером или типом аппаратных средств. Некоторые детали подходящих компьютеров и аппаратных средств хорошо известны и их не требуется подробно описывать в данном раскрытии.

[093] Следует также хорошо понимать, что любые функциональности, описанные здесь, могут быть выполнены, по меньшей мере частично, одним или более логическими компонентами аппаратных средств, а не программного обеспечения. Например, и без

ограничения, иллюстративные типы логических компонентов аппаратных средств, которые могут быть использованы, включают в себя программируемые пользователем вентильные матрицы (FPGA), программно-специализированные интегральные схемы (ASIC), программно-специализированные стандартные продукты (ASSP), однокристальные системы (SOC), комплексные программируемые логические устройства (CPLD) и т.д.

[094] Кроме того, любые из основанных на программном обеспечении вариантов осуществления (содержащих, например, исполняемые компьютером инструкции, чтобы предписывать компьютеру выполнять любой из описанных способов) могут выгружаться, загружаться или быть удаленно доступными через подходящие средства связи. Такие подходящие средства связи включают в себя, например, Интернет, всемирную паутину, интранет, приложения программного обеспечения, кабели (включая волоконно-оптический кабель), магнитную связь, электромагнитную связь (в том числе RF, микроволновую и инфракрасную связь), электронные коммуникации или другие подобные средства связи.

[095] Описанные способы, устройства и системы не должны быть истолкованы как ограничивающие каким-либо образом. Вместо этого, настоящее изобретение направлено на все новые и неочевидные признаки и аспекты различных раскрытых вариантов осуществления, отдельно и в различных сочетаниях и под-комбинациях друг с другом. Описанные способы, устройство и системы не ограничиваются каким-либо конкретным аспектом или признаком или их комбинацией, и раскрытые варианты осуществления не требуют, чтобы любое одно или более конкретных преимуществ присутствовало, или проблемы были решены. Ввиду многих возможных вариантов осуществления, к которым могут быть применены принципы раскрытого изобретения, следует иметь в виду, что проиллюстрированные варианты осуществления являются только предпочтительными примерами осуществления изобретения и не должны рассматриваться как ограничивающие объем изобретения. Скорее, объем изобретения определяется следующей формулой изобретения. Поэтому в качестве изобретения заявляется все, что входит в объем этих пунктов формулы изобретения и их эквивалентов.

(57) Формула изобретения

1. Способ генерации собственного кода из кода на промежуточном языке для приложения, содержащий:

прием (420), в вычислительном устройстве (350), характерного для типа устройства установочного пакета (325) для приложения от онлайн-провайдера (305), причем пакет включает в себя файлы на машинно-зависимом промежуточном языке (MDIL-файлы (320)), которые включают в себя MDLI-код, генерируемый онлайн-провайдером для приложения, список связывания, который включает в себя перечень MDIL-файлов (320) приложения, которые подлежат связыванию для генерации собственного образа (380) для приложения, и перечень набора из одной или более библиотек (370), которые подлежат связыванию с соответствующими MDIL-файлами,

при этом MDIL-код включает в себя машинно-зависимые инструкции, основанные на наборе инструкций процессора, неразрешенные псевдоинструкции с символьными ссылками, которые должны быть разрешены, чтобы генерировать собственные инструкции, посредством связывания, и при этом MDIL-код был ранее скомпилирован удаленно от вычислительного устройства;

в вычислительном устройстве, установку (220) приложения на вычислительном

устройстве путем генерации собственного образа для приложения путем связывания MDIL-кода, причем генерация собственного образа содержит связывание (430) части MDIL-кода с одной или более библиотеками на вычислительном устройстве, в том числе разрешение неразрешенных символьных ссылок, чтобы сгенерировать собственные инструкции для собственного образа; и

сохранение (440) собственного образа на вычислительном устройстве для использования при загрузке приложения для исполнения.

2. Способ по п. 1, дополнительно содержащий обновление (450) вычислительного устройства (350), причем обновление вычислительного устройства содержит:

обновление набора библиотек на вычислительном устройстве, причем набор библиотек включает в себя по меньшей мере одну библиотеку из одной или более библиотек, связанных с частью MDIL-кода;

обновление по меньшей мере одного механизма среды выполнения вычислительного устройства;

после обновления набора библиотек автоматическую генерацию обновленного собственного образа (380) для приложения путем связывания части MDIL-кода с упомянутой по меньшей мере одной библиотекой из упомянутой одной или более библиотек после того, как упомянутая одна или более библиотек обновлены на вычислительном устройстве, причем обновленный собственный образ для приложения генерируется таким образом, что обновленный собственный образ является исполняемым с использованием обновленного по меньшей мере одного механизма среды выполнения; и

сохранение обновленного собственного образа на вычислительном устройстве для использования при загрузке приложения для исполнения.

3. Способ по п. 1, дополнительно содержащий загрузку собственного образа (380) во время исполнения приложения, при этом загрузка собственного образа выполняется посредством среды выполнения общего языка.

4. Способ по п. 1, в котором псевдоинструкции кодированы на двоичном уровне.

5. Способ по п. 4, в котором генерация собственных инструкций содержит генерацию численного смещения поля на основе псевдоинструкций.

6. Способ по п. 5, в котором псевдоинструкции включают в себя маркер, который идентифицирует поле, и собственные инструкции включают в себя численное смещение поля для ссылки на поле при исполнении.

7. Способ по п. 1, дополнительно содержащий:

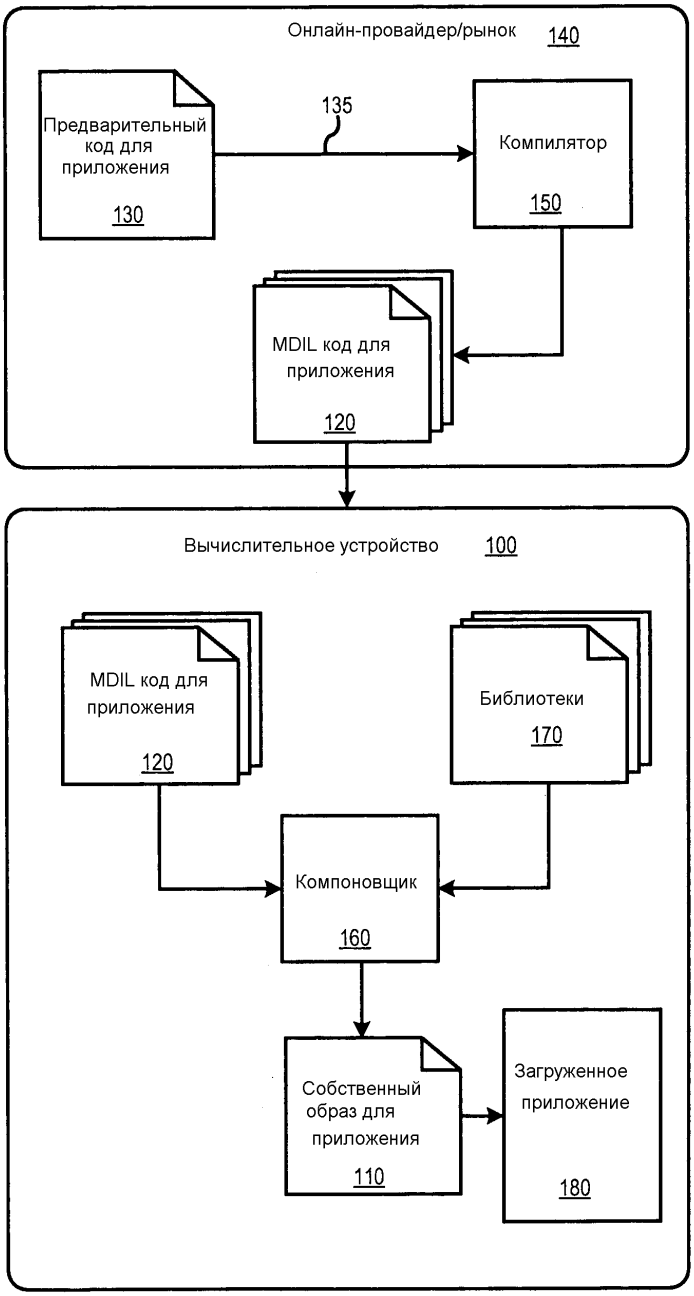
прием нового MDIL-кода для приложения;

обновление приложения на вычислительном устройстве (350) путем генерации обновленного собственного образа для приложения, в том числе связывание новой части MDIL-кода с упомянутой одной или более библиотеками (370) на вычислительном устройстве; и

сохранение обновленного собственного образа на вычислительном устройстве для использования при загрузке приложения для исполнения.

8. Вычислительное устройство (1000), которое включает в себя процессор (1010) и память (1020, 1025, 1040), причем память хранит исполняемые компьютером инструкции (1080), чтобы побуждать вычислительное устройство выполнять способ по любому из предшествующих пунктов.

1


Онлайн-провайдер/рынок 140
Предварительный код для приложения 130
135
Компилятор 150
MDIL код для приложения 120
Вычислительное устройство 100
MDIL код для приложения 120
Библиотеки 170
Компоновщик 160
Собственный образ для приложения 110
Загруженное приложение 180

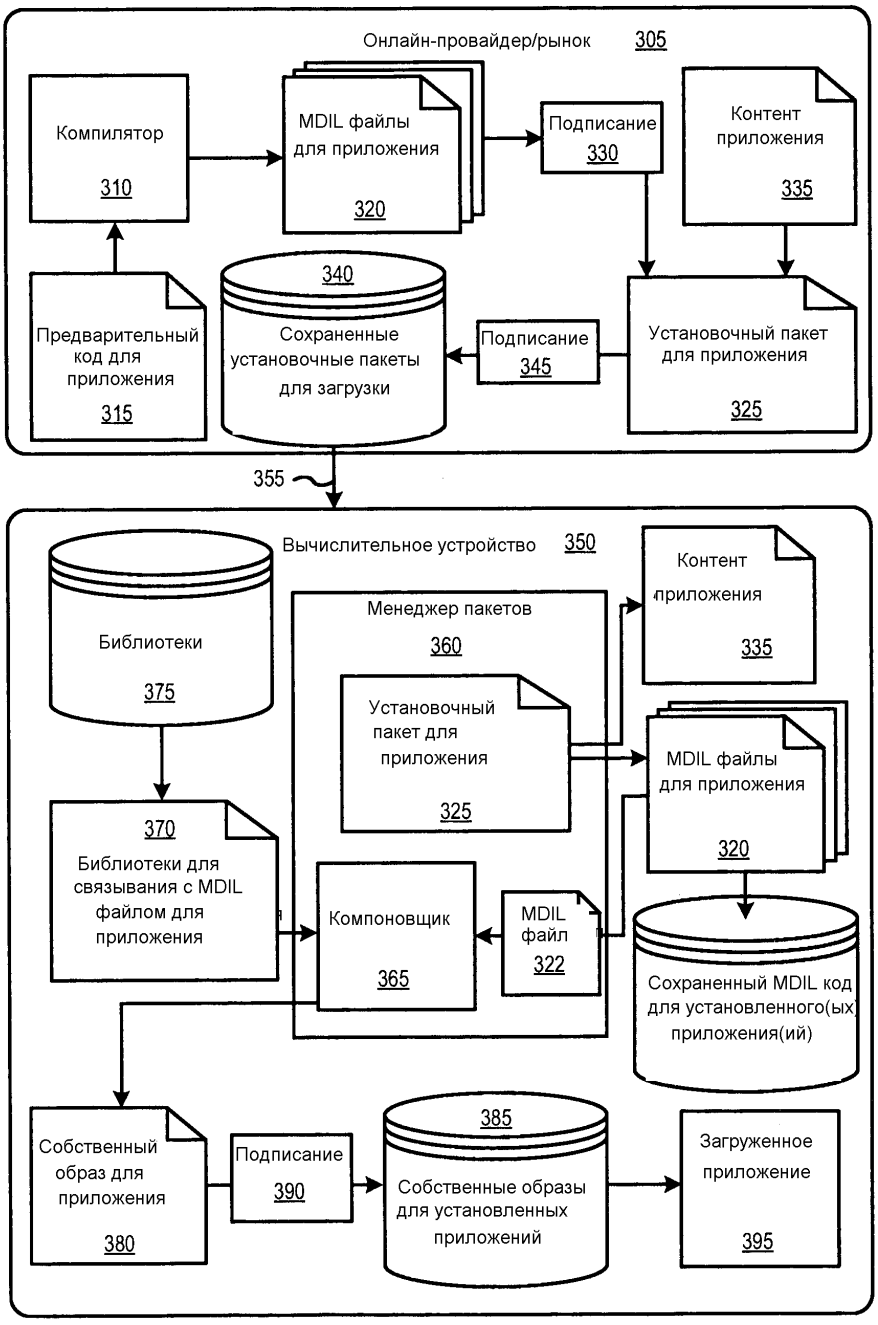
ФИГ.1

2


200
Прием в вычислительном устройстве MDIL кода для приложения от онлайн-провайдера
210
Установка приложения на вычислительном устройстве путем генерации собственного образа для приложения путем связывания MDIL кода
220
Сохранение собственного образа на вычислительном устройстве для использования при загрузке приложения для исполнения
230

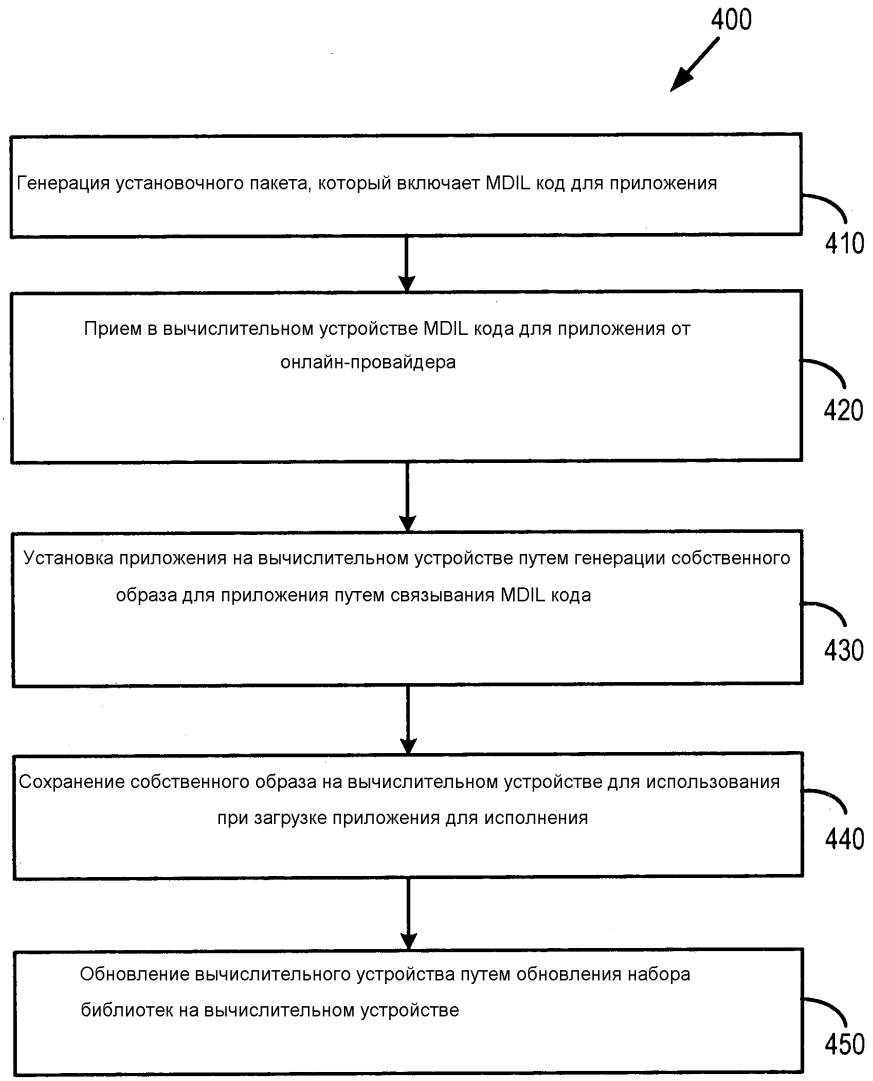
ФИГ.2

3

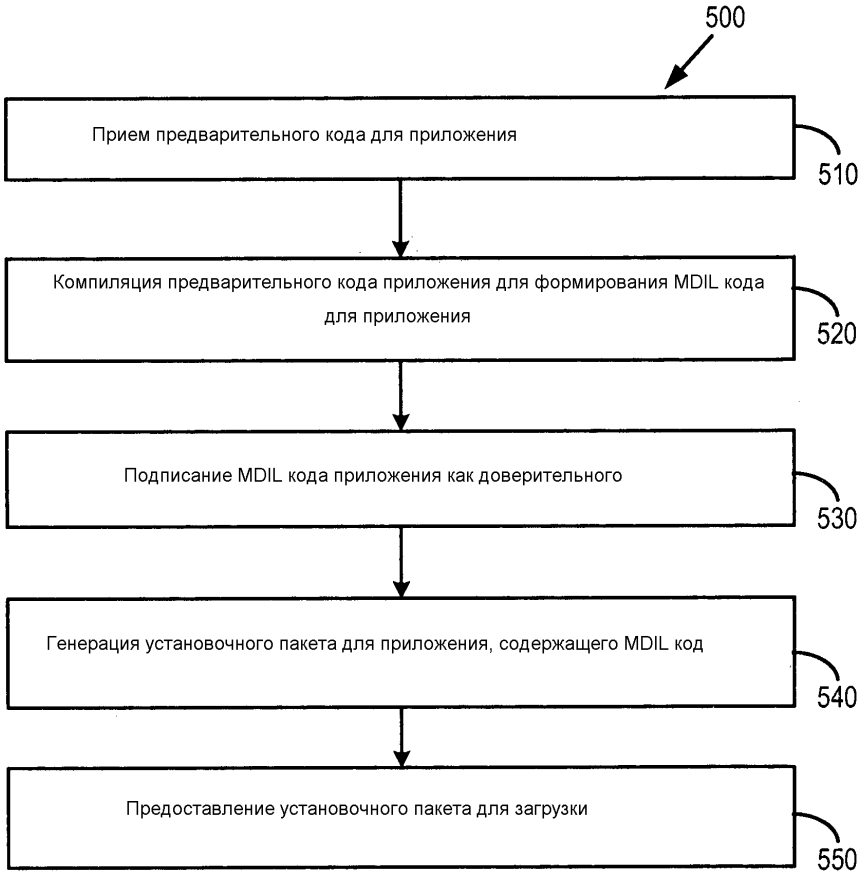

Онлайн-провайдер/рынок 305
Компилятор 310
MDIL файлы для приложения 320
Подписание 330
Контент приложения 335
Предварительный код для приложения 315
340
Сохраненные установочные пакеты для загрузки
Подписание 345
Установочный пакет для приложения 325
355
Вычислительное устройство 350
Библиотеки 375
Менеджер пакетов 360
Установочный пакет для приложения 325
Контент приложения 335
MDIL файлы для приложения 320
370
Библиотеки для связывания с MDIL файлом для приложения
Компоновщик 365
MDIL файл 322
Сохраненный MDIL код для установленного(ых) приложения(ий)
Собственный образ для приложения 380
Подписание 390
385
Собственные образы для установленных приложений
Загруженное приложение 395

ФИГ.3

4

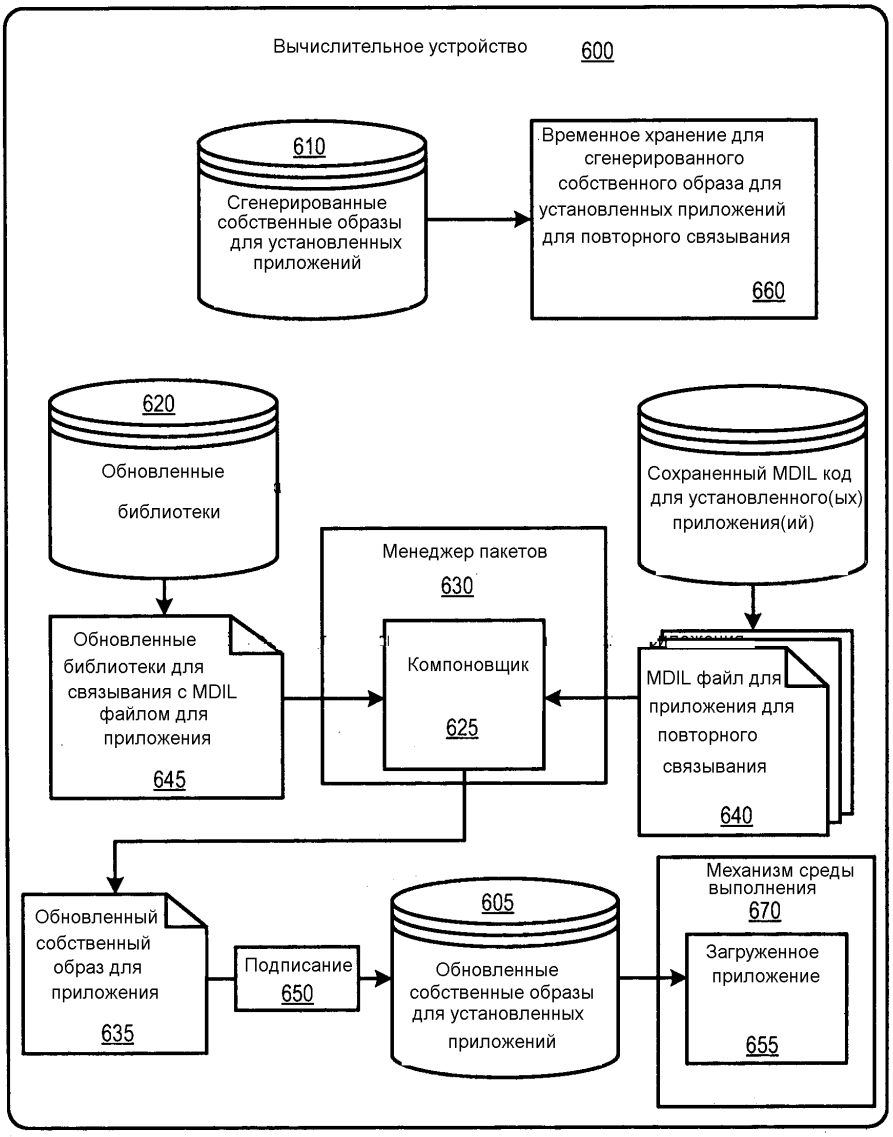

400
Генерация установочного пакета, который включает MDIL код для приложения
410
Прием в вычислительном устройстве MDIL кода для приложения от онлайн-провайдера
420
Установка приложения на вычислительном устройстве путем генерации собственного образа для приложения путем связывания MDIL кода
430
Сохранение собственного образа на вычислительном устройстве для использования при загрузке приложения для исполнения
440
Обновление вычислительного устройства путем обновления набора библиотек на вычислительном устройстве
450

ФИГ.4

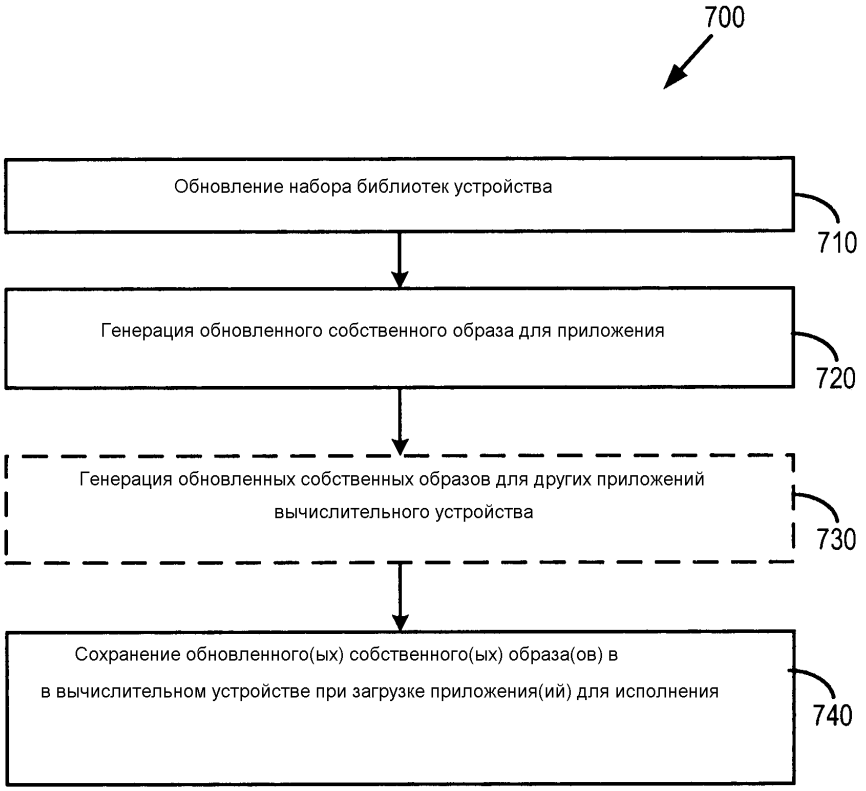

500
Прием предварительного кода для приложения
510
Компиляция предварительного кода приложения для формирования MDIL кода для приложения
520
Подписание MDIL кода приложения как доверительного
530
Генерация установочного пакета для приложения, содержащего MDIL код
540
Предоставление установочного пакета для загрузки
550

ФИГ.5

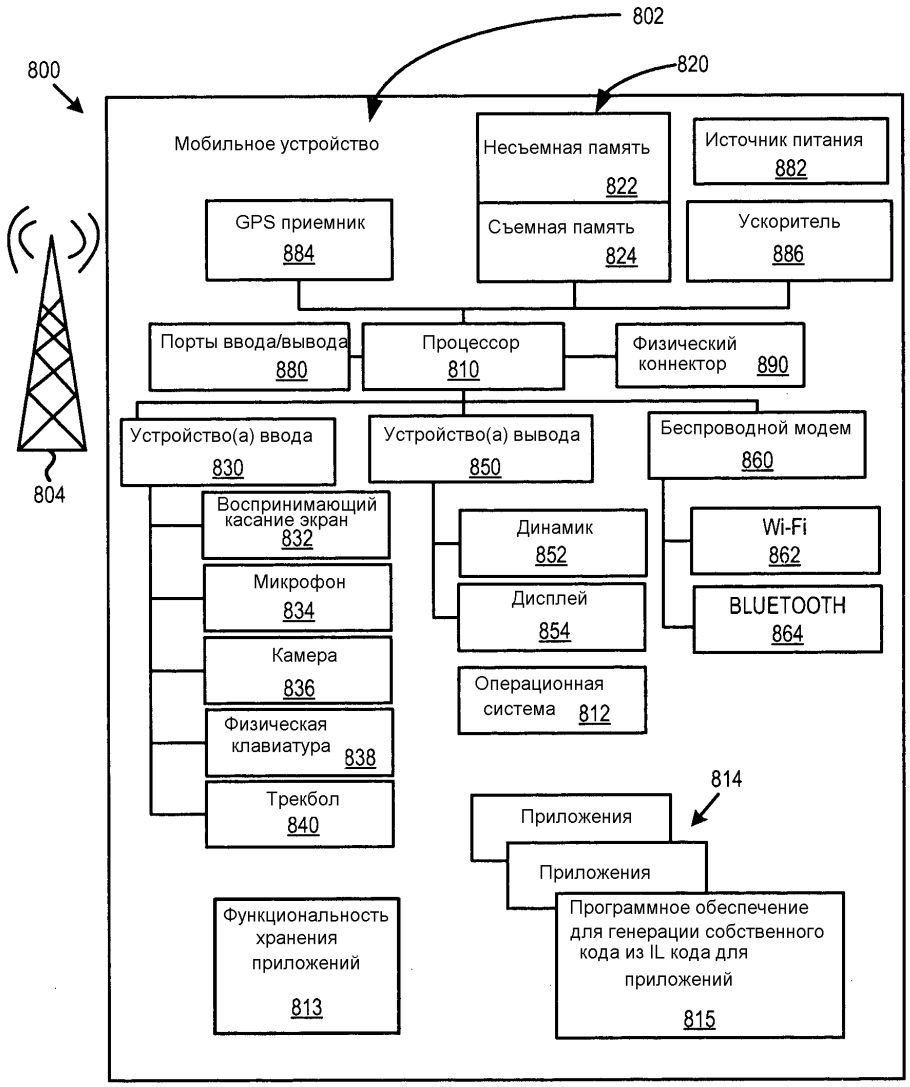
6


Вычислительное устройство 600
610
Сгенерированные собственные образы для установленных приложений
Временное хранение для сгенерированного собственного образа для установленных приложений для повторного связывания
660
620
Обновленные библиотеки
Сохраненный MDIL код для установленного(ых) приложения(ий)
Менеджер пакетов
630
Обновленные библиотеки для связывания с MDIL файлом для приложения
645
Компоновщик
625
MDIL файл для приложения для повторного связывания
640
Механизм среды выполнения
670
Обновленный собственный образ для приложения
635
Подписание
650
605
Обновленные собственные образы для установленных приложений
Загруженное приложение
655

ФИГ.6

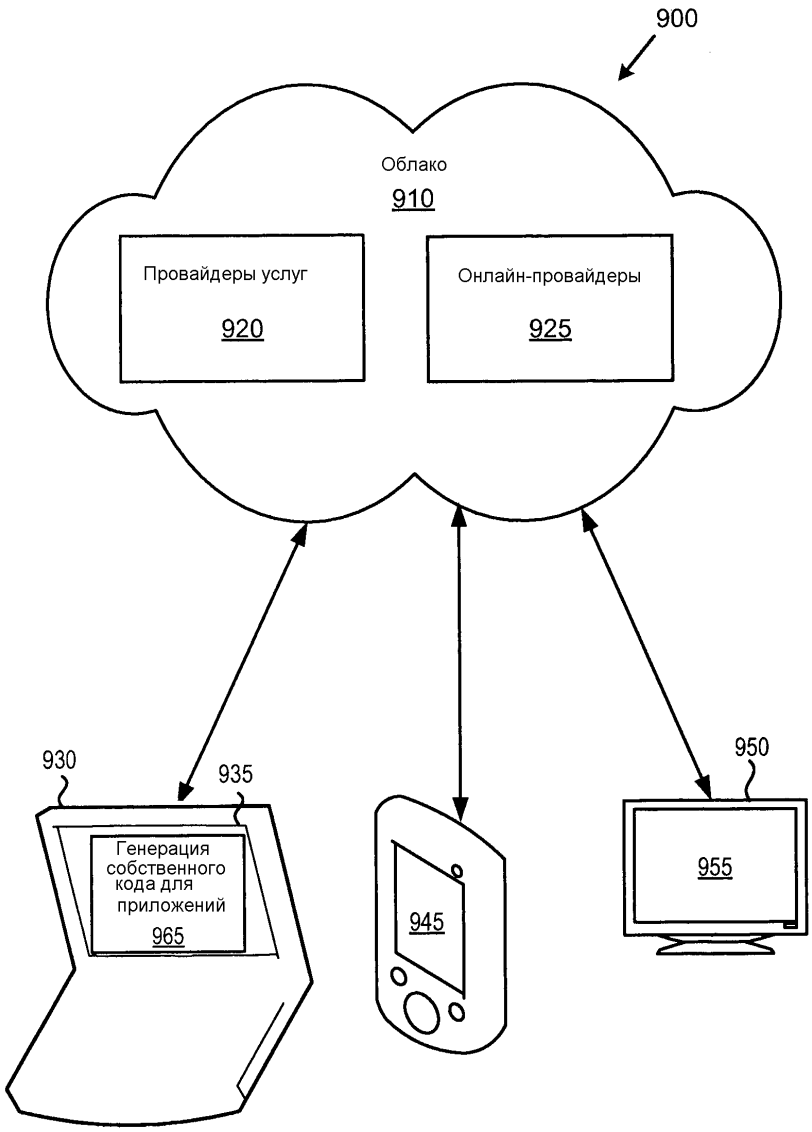

700
Обновление набора библиотек устройства
710
Генерация обновленного собственного образа для приложения
720
Генерация обновленных собственных образов для других приложений вычислительного устройства
730
Сохранение обновленного(ых) собственного(ых) образа(ов) в в вычислительном устройстве при загрузке приложения(ий) для исполнения
740

ФИГ.7


802
820
800
Мобильное устройство
Несъемная память 822
Источник питания 882
GPS приемник 884
Съемная память 824
Ускоритель 886
Порты ввода/вывода 880
Процессор 810
Физический коннектор 890
Устройство(а) ввода 830
Устройство(а) вывода 850
Беспроводной модем 860
Воспринимающий касание экран 832
Динамик 852
Wi-Fi 862
Микрофон 834
Дисплей 854
BLUETOOTH 864
Камера 836
Операционная система 812
Физическая клавиатура 838
814
Трекбол 840
Приложения
Приложения
Функциональность хранения приложений 813
Программное обеспечение для генерации собственного кода из IL кода для приложений 815
804

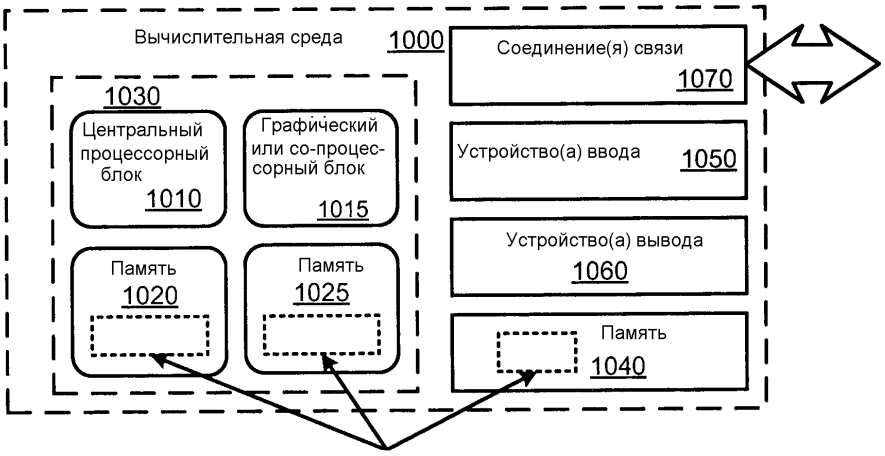
ФИГ.8

9


900
Облако
910
Провайдеры услуг
920
Онлайн-провайдеры
925
930
935
Генерация собственного кода для приложений
965
945
950
955

ФИГ.9

10


Вычислительная среда
1000
1030
Центральный процессорный блок
1010
Графический или со-процес-сорный блок
1015
Память
1020
Память
1025
Соединение(я) связи
1070
Устройство(а) ввода
1050
Устройство(а) вывода
1060
Память
1040
Программное обеспечение, реализующее описанные технологии
и собственного кода из IL кода для приложений

ФИГ.10