(19) Евразийское патентное ведомство

(11) 044257 (13) B1

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ЕВРАЗИЙСКОМУ ПАТЕНТУ

(45) Дата публикации и выдачи патента
2023.08.08

(21) Номер заявки
202290809

(22) Дата подачи заявки
2020.09.04

(51) Int. Cl. ***G05B 19/4155*** (2006.01)
G06F 9/44 (2006.01)
B65G 43/00 (2006.01)

(54) СИСТЕМЫ, УСТРОЙСТВО И СПОСОБЫ КОНВЕЙЕРНОЙ ОБРАБОТКИ

(31) **62/896,682**
(32) **2019.09.06**
(33) **US**
(43) **2022.06.07**
(86) **PCT/US2020/070502**
(87) **WO 2021/046581 2021.03.11**

(71)(73) Заявитель и патентовладелец:
ИНТЕЛЛОДЖЕНС ИНК. (US)

(72) Изобретатель:
Шлаймун Зиа, Шлаймун Нико (US)

(74) Представитель:
Нилова М.И. (RU)

(56) US-A1-20180143777
US-A1-20120245916
US-A1-20140215245
US-A1-20110179252
US-B1-7844758

(57) Предложена реконфигурируемая аппаратная платформа вместо части программного обеспечения, которая использует цепочку реконфигурируемых аппаратных блоков операторов для манипулирования данными по мере того, как данные перемещаются по цепочке. Эта конвейерная архитектура или цепочка блоков операторов перемещает данные от блока операторов к блоку операторов. Этот процессор с конвейерной архитектурой может быть объединен с известным входным интерфейсным процессором для обработки сложной информации или критических циклов в аппаратных средствах, в то время как остальная часть программы обрабатывается в качестве программного обеспечения.

044257 B1

044257 B1

Перекрестная ссылка на родственную заявку

Эта заявка заявляет приоритет и преимущество предварительной заявки на патент США № 62/896,682, поданной 6 сентября 2019, которая посредством ссылки полностью включена в настоящий документ.

Область техники

Настоящее изобретение относится к компьютерам и, в частности, компьютерным процессорам.

Уровень техники

Цифровые вычислительные машины, предназначенные для универсальных вычислений, могут использовать стандартную архитектуру, такую как неймановская архитектура. Разработанная примерно в 1945 году физиком и математиком Джоном фон Нейманом машина с неймановской архитектурой может быть теоретическим проектом цифровой вычислительной машины с хранимой в памяти программой.

Краткое описание чертежей

На фиг. 1 представлена функциональная схема, показывающая пример вычислительной системы.

На фиг. 2 представлена функциональная схема, показывающая вычислительную систему с конвейерной архитектурой.

На фиг. 3 представлен исходный код программы, печатающей числа Фибоначчи.

На фиг. 4 представлен машинный код программы, который исполняется в системе со стандартной архитектурой, для печатания чисел Фибоначчи.

На фиг. 5 представлена блок-схема блоков операторов, которые исполняются системой с конвейерной архитектурой, для печатания чисел Фибоначчи.

На фиг. 6 представлен исходный код, который вычисляет и распечатывает сумму цифр.

На фиг. 7 представлена первая четверть машинного кода, исполняемая системой со стандартной архитектурой, которая вычисляет и распечатывает сумму цифр.

На фиг. 8 представлена вторая четверть машинного кода, исполняемая системой со стандартной архитектурой, которая вычисляет и распечатывает сумму цифр.

На фиг. 9 представлена третья четверть машинного кода, исполняемая системой со стандартной архитектурой, которая вычисляет и распечатывает сумму цифр.

На фиг. 10 представлена четвертая четверть машинного кода, исполняемая системой со стандартной архитектурой, которая вычисляет и распечатывает сумму цифр.

На фиг. 11 представлена блок-схема блоков операторов, исполняемых системой с конвейерной архитектурой, которая вычисляет и распечатывает сумму цифр.

На фиг. 12 представлена блок-схема, показывающая вычислительную систему с конвейерной архитектурой, используемую в сочетании с вычислительной системой со стандартной архитектурой.

На фиг. 13 представлена блок-схема, показывающая, как может быть исполнена программа посредством вычислительной системы с конвейерной архитектурой и вычислительной системой со стандартной архитектурой.

На фиг. 14 представлена блок-схема способа подготовки конвейерной архитектуры.

На фиг. 15 представлена блок-схема, показывающая вычислительную систему и компоненты.

Осуществление изобретения

Ниже приведено подробное описание систем и способов, соответствующих вариантам реализации настоящего изобретения. Хотя описано несколько вариантов реализации, следует понимать, что раскрытие не ограничивается каким-либо одним вариантом реализации, а вместо этого охватывает множество альтернатив, изменений и эквивалентов. Кроме того, несмотря на то, что в последующем описании изложены множество конкретных подробностей для обеспечения полного понимания раскрытых в настоящем документе вариантов реализации, некоторые варианты реализации могут быть осуществлены на практике без некоторых или всех этих подробностей. Кроме того, в целях ясности некоторые технические материалы, известные в данной области техники, подробно не описаны, чтобы избежать излишнего затемнения раскрытия.

Ниже раскрыты технологии, устройства и способы, в которых используется реконфигурируемая аппаратная платформа для взаимного соединения цепочки реконфигурируемых аппаратных блоков операторов для манипулирования данными по мере того, как данные перемещаются вниз по цепочке. Эта конвейерная архитектура или цепочка блоков операторов перемещает данные от блока операторов к блоку операторов. Вместо части программного обеспечения цепочка реконфигурируемых аппаратных блоков операторов может манипулировать данными по мере того, как данные перемещаются по цепочке.

Согласно некоторым вариантам реализации вычислительная система с конвейерной архитектурой может использоваться исключительно для выполнения вычислительных задач.

Множество вычислительных систем с конвейерной архитектурой могут использоваться последовательно или параллельно, например, для распределения рабочей нагрузки между вычислительными системами.

Вычислительная система с конвейерной архитектурой может использоваться в сочетании с вычислительной системой со стандартной архитектурой, например, рабочая нагрузка может быть распределена между вычислительными системами.

Множество вычислительных систем с конвейерной архитектурой могут использоваться последовательно или параллельно и использоваться в сочетании с вычислительной системой со стандартной архитектурой, например, для распределения рабочей нагрузки между вычислительными системами.

Вычислительная система с конвейерной архитектурой может использоваться в сочетании со множеством вычислительных систем со стандартной архитектурой, например, и рабочая нагрузка распределяется между вычислительными системами.

На фиг. 1 представлена функциональная схема, показывающая пример вычислительной системы, которая подобна вычислительной системе, имеющей неймановскую архитектуру, или содержит ее. Вычислительная система содержит вход 102, вычислительную систему 104 и выход 106. Вход 102 принимается (например, посредством шины и т.п.) в вычислительную систему 104, в которой он перед передачей (например, посредством шины и т.п.) из вычислительной системы 104 в качестве выхода 106. В вычислительной системе 104 содержатся запоминающее устройство произвольного доступа (ОЗУ) 108, которое связано с центральным процессором (ЦП) 112 посредством общей шины 110. Кроме того, центральный процессор 112 содержит арифметический логический блок (АЛУ) 116, блок 114 управления, регистры 118 и стеки 120.

Программы, исполняемые вычислительной системой со стандартной архитектурой, могут содержать набор команд, которые исполняются в конкретной последовательности, для управления данными.

После загрузки программы в ОЗУ 108 центральный процессор 112 может выполнять последовательность циклов "выборка-декодирование-исполнение", в результате чего содержимое ячеек ОЗУ 108 считывается, дешифруется и затем исполняется в конкретной последовательности, как указано программой. Поскольку ячейки ОЗУ 108 содержат команды и данные, центральный процессор 112 считывает и дешифрует команды для определения того, что делать с информацией, и затем исполняет их с получением результата. Некоторые команды предписывают центральному процессору 112 записать результат операции назад в ячейку ОЗУ 108, а другие команды предписывают центральному процессору 112 перейти к конкретной ячейке в ОЗУ 108, в зависимости от результата предыдущей команды.

Проблема этой архитектуры может состоять в том, что программная команда и данные содержатся в том же самом ОЗУ 108. Информация в ОЗУ 108 может быть считана по одной ячейке за такт и дешифрована, что приводит к неэффективности архитектуры и ограничению производительности. Кроме того, общая шина 110 не может обеспечить возможность для центрального процессора 112 считывать и записывать информацию одновременно. Это называется узким местом и может еще больше ограничивать производительность системы.

На фиг. 2 показана схема, иллюстрирующая вычислительную систему с конвейерной архитектурой, содержащую вход 202, вычислительную систему 204 и выход 206. Вход 202 принимается (например, посредством шины и т.п.) вычислительной системой 204, в которой он обрабатывается перед передачей (например, посредством шины и т.п.) из вычислительной системы 204 в качестве выхода 206. В вычислительной системе 204 содержится реконфигурируемая аппаратная платформа 208 (например, программируемая пользователем вентильная матрица (ППВМ)), которая содержит множество реконфигурируемых блоков 210, 212, 214, 216 и 218 операторов, которые соединены потоками 220, 222, 224 и 226 данных в одном направлении и потоком 228 данных в противоположном направлении.

Вместо центрального процессора, связанного с ОЗУ посредством шины, конвейерная архитектура может использовать реконфигурируемую аппаратную платформу, такую как ППВМ 208, для взаимного соединения цепи реконфигурируемых блоков 210, 212, 214, 216 и 218 операторов с целью управления данными по мере того, как данные проходя по цепи от блока операторов к блоку операторов в потоках 220, 222, 224 и 226 данных в одном направлении и потоке 228 данных в противоположном направлении.

Согласно одному варианту реализации в каждом блоке 210, 212, 214, 216 и 218 операторов выполняется операция или группа операций для управления данными перед тем, как данные будут переданы к следующему блоку операторов указанной цепи в потоках 220, 222, 224 и 226 данных в одном направлении и потоке 228 данных в противоположном направлении.

Программа транслируется и затем копируется в реконфигурируемую аппаратную платформу 208 (например, ППВМ и т.п.). Каждая команда или группа команд назначаются блоку 210, 212, 214, 216 и 218 операторов, и ход выполнения программы определяется взаимосвязью этих блоков операторов.

Данными манипулируют в каждом блоке 210, 212, 214, 216 и 218 операторов по мере того, как они проходят по цепи от блока операторов к блоку операторов в потоках 220, 222, 224 и 226 данных.

В случае команды "перехода" поток данных может быть изменен/перенаправлен блоком операторов в противоположном направлении или к некоторому другому блоку операторов в отдельном потоке 228 данных. В этом примере переход на основании условия, которое соблюдено, показан от блока 4 (216) операторов назад к блоку 2 (212) операторов.

Кроме того, блок 210, 212, 214, 216 и 218 операторов может быть автономным и может быть выполнен с возможностью обработки данных либо асинхронно, либо синхронно, когда он принимает их от предыдущего блока операторов в цепи.

Согласно одному варианту реализации автономной операции конвейерная архитектура обеспечивает возможность исполнения множества команд в одиночном цикле процессора.

Конвейерная архитектура может быть более эффективной, чем стандартная архитектура, поскольку она не требует, чтобы программа была считана из ОЗУ и дешифрована.

Конвейерная архитектура может предотвращать возникновение узких мест, связанных с известной компьютерной архитектурой, поскольку она не основана на потоке общей шины, и каждый набор блоков операторов имеет свой собственный поток данных.

Конвейерная архитектура может обеспечивать возможность более высокой производительности и вычислительной мощности. Дополнительное преимущество этой архитектуры состоит в том, что при работе в синхронном режиме конвейерная архитектура может быть способна упаковывать данные с большей плотностью в реконфигурируемой аппаратной платформе путем постановки данных в очередь на входе каждого блока операторов, готовых к загрузке в последующий блок операторов, по мере их поступления.

Программные команды могут содержаться в блоках операторов в виде аппаратных логических вентилей, а не программного обеспечения, что значительно ускоряет выполнение инструкций по сравнению с программным аналогом.

Еще одно преимущество конвейерной архитектуры состоит в том, что программу может быть труднее взломать. Программа может храниться в виде аппаратного средства, и любое изменение программы хакером может разорвать цепочку конвейера и привести к перезагрузке системы. Перезагрузка системы может привести к тому, что исходная (неизмененная) программа будет автоматически перезагружена системой в реконфигурируемую аппаратную платформу.

На фиг. 3-11 показаны различия между этими двумя архитектурами. Исходный код С и откомпилированные выходы как из более традиционной вычислительной системы, так и из вычислительной системы с конвейерной архитектурой исследуются для двух различных программ.

На фиг. 3 показан исходный код для печатания чисел Фибоначчи. Исходный код С показан для программы, используемой для распечатки чисел Фибоначчи в диапазоне от 0 до 255.

На фиг. 4 показан машинный код для печатания чисел Фибоначчи. Исходный код С, показанный на фиг. 3, может быть откомпилирован для исполнения традиционной вычислительной системой. Результирующий машинный язык может выглядеть подобно распечатке, показанной на фиг. 4. Традиционная вычислительная система может использовать по меньшей мере 85 тактов тактового генератора центрального процессора для завершения первой итерации цикла вычисления и печати. После этого традиционная вычислительная система может использовать по меньшей мере 56 тактов центрального процессора для завершения последующих итераций цикла вычисления и печати.

На фиг. 5 показана блок-схема блоков операторов для печатания чисел Фибоначчи. По сравнению с фиг. 4, исходный код С, показанный на фиг. 3, может быть откомпилирован для исполнения вычислительной системой с конвейерной архитектурой. Результирующие блоки операторов, используемые для исполнения программы, могут выглядеть подобно показанным на фиг. 5.

Блок 1 502 операторов (ОВ #1) назначает значения переменных "x=0" и "y=1". Блок 2 504 операторов (ОВ #2) выполняет функцию "printf". Блок 3 506 операторов (ОВ #3) суммирует содержание x и y и назначает результат переменной z. Он также назначает значение y значению x и значение z к значению y. Блок 4 508 операторов (ОВ #4) выполняет условный переход назад к началу блока 2 504 операторов, если результатом "x<255" является истина и назад к началу блока 1 502 операторов, если результат "x<255" ложен.

Согласно данному варианту реализации множество команд могут быть сгруппированы в одиночном блоке 502, 504, 506 и 508 операторов, что обеспечивает возможность проведения множества операций в отношении данных прежде, чем эти данные будут переданы к следующему блоку операторов. При работе в синхронном режиме конвейерная архитектура может использовать четыре такта процессора для завершения первой и последующих итераций цикла вычисления и печати. Это может обеспечить возможность для машины с конвейерной архитектурой в этом примере работать в 14 раз быстрее, чем известная машина с подобным тактовым циклом (т.е. 56 тактов против 4 тактов).

На фиг. 6 показан исходный код для нахождения суммы цифр. Исходный код С для программы для нахождения суммы цифр числа с использованием рекурсии показан на фиг. 6.

На фиг. 7-10 показаны машинные коды для нахождения суммы цифр. Исходный код С, показанный на фиг. 6, может быть откомпилирован для исполнения неймановской вычислительной системой. Результирующий машинный язык может выглядеть подобно распечаткам, показанным на фиг. 7-10. "Основной" цикл осуществляет вызов отдельного цикла 802 "суммы" для вычисления и возврата результата 1002. В цикле суммы содержится условный оператор 902 "если". В зависимости от результата условного оператора "если" цифровая вычислительная машина может использовать либо 113 тактовых циклов центрального процессора, либо 191 такт тактового цикла для обработки одной итерации.

На фиг. 9 показана блок-схема блоков операторов для нахождения суммы цифр. По сравнению с фиг. 7-10 исходный код С, показанный на фиг. 6, может быть откомпилирован для исполнения вычислительной системой с конвейерной архитектурой. Результирующие блоки операторов, используемые для исполнения программы, могут выглядеть подобно показанным на фиг. 11.

Блок 1 1102 операторов (ОВ #1) выполняет функцию "printf" для печатания на устройстве вывода

"Ведите число". Блок 2 1104 операторов (ОВ #2) выполняет функцию "scanf" для ввода числа из устройства ввода. Блок 3 1106 операторов (ОВ #3) выполняет условный оператор "если", который сравнивает введенное число с 0, и затем перенаправляет программу либо к блоку 4 1108 операторов (ОВ #4), если результат положителен, или блоку 5 1110 операторов (ОВ #5), если результат отрицателен. Блок 4 операторов 1108 (ОВ #4) выполняет вычисление. Блок 5 1110 операторов (ОВ #5) возвращает 0. Блок 6 1112 операторов (ОВ #6) назначает число, возвращенное либо блоком ОВ #4 (1108), либо блоком ОВ #5 (1110) переменной "сумма". Блок 7 1114 операторов (ОВ #7) выполняет функцию "printf" для распечатки суммы на устройстве вывода. Кроме того, выход этого блока 1114 операторов соединен с входом блока ОВ #1 (1102), чтобы программа могла быть зациклена на неопределенный срок.

Согласно данному варианту реализации блок операторов может перенаправлять цепочку программы в зависимости от результата условия. Работая в синхронном режиме, вычислительная система с конвейерной архитектурой может использовать шесть тактовых циклов процессора для завершения итерации программы, независимо от результата команды "если". Вычислительная система с конвейерной архитектурой в этом примере может работать в 18 раз быстрее, чем вычислительная система со стандартной архитектурой для подобного тактового цикла (т.е. 113 тактов против 6 тактов).

Вычислительная система с конвейерной архитектурой может быть значительно быстрее, чем вычислительная система со стандартной архитектурой в зависимости от приложения. Например, вычислительная система с конвейерной архитектурой может работать быстрее в приложениях, в которых обрабатывается большое количество данных. Преимущество производительности вычислительной системы с конвейерной архитектурой по сравнению с вычислительной системой со стандартной архитектурой может зависеть от исполняемой программы. В ходе тестирования было замечено, что в некоторых приложениях преимущество может находиться в диапазоне от 100 до 2000%.

На фиг. 12 показана блок-схема, иллюстрирующая вычислительную систему с конвейерной архитектурой, используемую в сочетании с вычислительной системой со стандартной архитектурой. Согласно данному варианту реализации входной интерфейс 1202 вычислительной системы со стандартной архитектурой соединен с выходным интерфейсом 1204 вычислительной системы с конвейерной архитектурой посредством общей шины 1206.

Входной интерфейс вычислительной системы со стандартной архитектурой содержит следующие компоненты: центральный процессор (ЦП) 1208; динамическое запоминающее устройство произвольного доступа (динамическое ОЗУ) 1210; адаптер 1212 локальной сети (ЛВС); базовая система ввода-вывода (BIOS) 1214 и жесткий диск (HDD) 1216, которые все соединены вместе посредством устройства 1206 общей шины. В случае HDD 1216 это соединение осуществлено посредством интерфейса (I/F) 1218.

Также согласно данному варианту реализации показан блок 1220 графического процессора (GPU) и дополнительный сопроцессор 1222.

Выходной интерфейс вычислительной системы с конвейерной архитектурой имеет встроенную ППВМ 1224, которая соединена с остальными компонентами в общей системе посредством общей шины 1206.

Поскольку некоторые программы могут бездействовать в течение значительной части рабочего времени, нет особого смысла выполнять этот код бездействия в вычислительной системе с конвейерной архитектурой. Вместо этого только конкретные разделы программы (например, критические циклы, критические потоки) могут транслироваться и исполняться вычислительной системой с конвейерной архитектурой для выполнения "тяжелой работы". Остальная часть программы без конкретных разделов может по-прежнему исполняться вычислительными системами со стандартной архитектурой. Совместное использование этих двух архитектур позволяет избежать необходимости транслирования всей программы для исполнения вычислительной системой с конвейерной архитектурой. Это может препятствовать использованию ценных ресурсов вычислительной системы с конвейерной архитектурой (например, адресного пространства программы), которое может не принести какой-либо ощутимой выгоды. Кроме того, использование этих двух архитектур обеспечивает совместимость с существующими программами, которые предназначены для исполнения вычислительной системой со стандартной архитектурой.

Согласно некоторым вариантам реализации вычислительная система с конвейерной архитектурой может использоваться в соединении с вычислительными системами со стандартной архитектурой. На фиг. 13 представлена блок-схема, показывающая, как программа может быть исполнена вычислительной системой с конвейерной архитектурой и вычислительными системами со стандартной архитектурой. Входной интерфейс 1302 вычислительной системы со стандартной архитектурой соединен с выходным интерфейсом 1304 вычислительной системы с конвейерной архитектурой посредством шины 1306. Затем основное тело программы 1308 вызывает подпрограмму A 1312 и подпрограмму B 1314 вычислительной системы с конвейерной архитектурой посредством функций 1316 и 1320 вызова и возвращает результаты 1318 и 1322 соответственно.

На фиг. 14 представлена блок-схема способа подготовки конвейерной архитектуры. Способ может быть осуществлен системами и/или компонентами, описанными в настоящем изобретении, включая систему 204 на фиг. 2. На этапе 1402 система с конвейерной архитектурой может принять программу, выполненную с возможностью исполнения в качестве программного обеспечения. На этапе 1404 система с

конвейерной архитектурой может определить, что первая часть программы должна исполняться аппаратными средствами, а вторая часть программы должна исполняться как программное обеспечение. На этапе 1406 система с конвейерной архитектурой на основании первой части может определить множество взаимосвязанных перепрограммируемых блоков операторов, содержащих одну или более функций преобразования, которые берут входные данные из шины с предшествующими данными, выполняет одно или более преобразований входных данных и выводит преобразованные входные данные посредством шины выходных данных. На этапе 1408 система с конвейерной архитектурой может конфигурировать множество взаимосвязанных перепрограммируемых блоков операторов для исполнения одним или более перепрограммируемых процессоров. На этапе 1410 система с конвейерной архитектурой может исполнять вторую часть посредством одного или более входных интерфейсных процессоров. На этапе 1412 система с конвейерной архитектурой может отправить первые данные от одного или более входных интерфейсных процессоров к одному или более перепрограммируемых процессоров. На этапе 1414 система с конвейерной архитектурой может исполнить первую часть посредством одного или более перепрограммируемых процессоров. На этапе 1416 система с конвейерной архитектурой может отправить вторые данные от одного или более перепрограммируемых процессоров к одному или более входных интерфейсных процессоров. На этапе 1418 система с конвейерной архитектурой на основании первых данных и вторых данных может определить результирующие данные.

Система с конвейерной архитектурой на основании вычислительной сложности может определить первую часть программы, которая должна исполняться аппаратными средствами.

Результирующие данные могут быть результатом исполнения указанной программы. Вторые данные могут быть выведены из первых данных на основании одного или более преобразований. Результирующие данные могут быть выведены из вторых данных, а вторые данные могут быть выведены из первых данных. Один или более перепрограммируемых процессоров могут представлять собой программируемую пользователем вентильную матрицу. Отправка первых данных от одного или более входных интерфейсных процессоров к одному или более перепрограммируемых процессоров может дополнительно включать передачу первых данных посредством шины расширения от одного или более входных интерфейсных процессоров к одному или более перепрограммируемых процессоров. Прием программы, выполненной с возможностью исполнения в качестве программного обеспечения, может дополнительно включать компилирование программы в исполняемый код программного обеспечения, конфигурирование аппаратных средств и код связи для передачи данных между исполняемым кодом программного обеспечения и конфигурацией аппаратных средств.

Конвейерный процессор может содержать шину входных данных, множество взаимосвязанных перепрограммируемых блоков операторов и шину выходных данных. Множество взаимосвязанных перепрограммируемых блоков операторов может содержать: шину входных данных первого перепрограммируемого блока операторов, соединенную с шиной выходных данных второго перепрограммируемого блока операторов или шиной входных данных; шину выходных данных первого перепрограммируемого блока операторов, соединенную с шиной входных данных третьего перепрограммируемого блока операторов или шиной выходных данных; и одну или более функций преобразования, которые берут входные данные от шины предшествующих данных, выполняют одно или более преобразований в отношении входных данных и выводят преобразованные входные данные через шину выходных данных.

Ширина шин множества взаимосвязанных перепрограммируемых блоков операторов может быть различной. Выход последующего блока может быть входом предыдущего блока. Второй перепрограммируемый блок операторов и третий перепрограммируемый блок операторов могут быть одинаковыми. Третий перепрограммируемый блок операторов может предшествовать первому перепрограммируемому блоку операторов в порядке исполнения. Конвейерный процессор может дополнительно содержать программный интерфейс, выполненный с возможностью приема команд для создания множества взаимосвязанных перепрограммируемых блоков операторов.

Система обработки данных может содержать множество процессоров и функцию управления, при этом функция управления выполнена с возможностью назначения данных каждому процессору. Каждый процессор может содержать шину входных данных, множество взаимосвязанных перепрограммируемых блоков операторов и шину выходных данных. Взаимосвязанные перепрограммируемые блоки операторов могут содержать: шину входных данных первого перепрограммируемого блока операторов, соединенную с шиной выходных данных второго перепрограммируемого блока операторов или шиной входных данных; шину выходных данных первого перепрограммируемого блока операторов, соединенную с шиной входных данных третьего перепрограммируемого блока операторов или шиной выходных данных; и одну или более функций преобразования, которые берут входные данные от шины предшествующих данных, выполняют одно или более преобразований в отношении входных данных и выводят преобразованные входные данные через шину выходных данных.

Функция управления может быть дополнительно выполнена с возможностью реконфигурирования множества взаимосвязанных перепрограммируемых блоков операторов. Функция управления может содержать команды сохранения в запоминающем устройстве для создания множества взаимосвязанных перепрограммируемых блоков операторов из множества процессоров. Система согласно настоящему

изобретению может дополнительно содержать по меньшей мере один входной интерфейсный процессор с архитектурой, отличающейся от архитектуры множества процессоров. По меньшей мере один входной интерфейсный процессор может содержать процессор общего назначения. Функция управления может содержать: защищенный интерфейс, выполненный с возможностью приема изменений конфигурации; и незащищенный интерфейс, выполненный с возможностью назначения данных одному или более процессоров из множества процессоров.

На фиг. 15 представлена блок-схема, показывающая компоненты согласно некоторым приведенным для примера вариантам реализации, выполненные с возможностью считывания команд из машиночитаемого или компьютерочитаемого носителя (например, машиночитаемого носителя данных) и осуществления любой одной или более методологий, описанных в настоящем документе. В частности, на фиг. 15 показано схематическое представление аппаратных ресурсов 1500, включая один или более процессоров (или процессорных ядер) 1510, одно или более запоминающих устройств/накопительных устройств 1520 и один или более коммуникационных ресурсов 1530, каждый из которых связан с возможностью обмена данными через шину 1540.

Процессоры 1510 (например, центральный процессор (ЦП), процессор вычислений с сокращенным набором команд (RISC), процессор вычислений со сложным набором команд (CISC), графический процессор (GPU), процессор цифровых сигналов (DSP), такой как процессор основной полосы частот, специализированная интегральная схема (ASIC), радиочастотная интегральная схема (RFIC), другой процессор или любое подходящее сочетание вышеперечисленного) могут включать в себя, например, процессор 1512 и процессор 1514. Запоминающее/накопительное устройства 1520 могут включить в себя основное запоминающее устройство, дисковый накопитель или любое подходящее сочетание вышеперечисленного.

Коммуникационные ресурсы 1530 могут включать в себя межсоединительные и/или сетевые интерфейсные компоненты или другие подходящие устройства для связи с одним или более периферийных устройств 1504 и/или одной или более баз 1506 данных по сети 1508. Например, ресурсы 1530 связи могут включать в себя проводные компоненты связи (например, для соединения по универсальной последовательной шине (USB)), компоненты сотовой связи, компоненты связи ближнего поля (NFC), компоненты Bluetooth® (например, Bluetooth® с низким энергопотреблением), компоненты Wi-Fi® и другие компоненты связи.

Команды 1550 могут содержать программное обеспечение, программу, приложение, апплет, прикладной код или другой исполняемый код для принуждения по меньшей мере одного из процессоров 1510 выполнять любую одну или более из методологий, описанных в настоящем документе. Команды 1550 могут полностью или частично находиться по меньшей мере в одном из процессоров 1510 (например, в кэш-памяти процессора), запоминающего/накопительного устройств 1520 или любом подходящем сочетании вышеперечисленного. Кроме того, любая часть команд 1550 может быть передана аппаратным ресурсам 1500 из любого сочетания периферийных устройств 1504 и/или баз данных 1506.

Соответственно, запоминающее устройство процессоров 1510, запоминающее/накопительное устройства 1520, периферийные устройства 1504 и базы 1506 данных является примерами компьютерочитаемых и машиночитаемых носителей.

Используемый в настоящем изобретении термин "схема" может относиться к специализированной интегральной схеме (ASIC), электронной схеме, процессору (общему, выделенному или групповому) и/или запоминающему устройству (общему, выделенному или групповому), быть частью вышеперечисленных устройств или включать в себя вышеперечисленные устройства, которые исполняют одно или более из программ или микропрограмм, комбинационной логической схемы и/или других подходящих аппаратных компонентов, которые обеспечивают описанные функциональные средства. Согласно некоторым вариантам реализации электронная схема может быть реализована в виде электронной схемы или может функционировать в соединении с электронной схемой, которая может быть реализована в одном или более из программных или микропрограммных модулей. Согласно некоторым вариантам реализации электронная схема может включать в себя логические средства, по меньшей мере частично действующие в аппаратных средствах.

Варианты реализации и осуществления систем и способов, описанных в настоящем документе, могут включать в себя различные операции, которые могут быть осуществлены в машинно-исполняемых командах, которые будут исполнены вычислительной системой. Вычислительная система может включать в себя один или более компьютеров общего или специального назначения (или других электронных устройств). Вычислительная система может включать в себя аппаратные компоненты, которые включают в себя конкретные логические средства для выполнения операций, или может включать в себя сочетание аппаратных средств, программного обеспечения и/или встроенных программ.

Вычислительные системы и компьютеры в вычислительной системе могут быть соединены посредством сети. Подходящие сети для конфигурирования и/или использования, как описано в настоящем документе, включают в себя одну или более локальных сетей, глобальных сетей, городских сетей и/или сети Интернет, или IP сетей, таких как Всемирная паутина (World Wide Web), частная сеть Интернет, защищенная сеть Интернет, сеть с дополнительными платными услугами, виртуальная частная сеть, экстранет (extranet), интранет или даже автономные машины, которые связаны с другими машинами путем

физической транспортировки носителей. В частности, подходящая сеть может быть образована из частей или целых двух или более других сетей, включая сети, использующие различные аппаратные средства и технологии сетевой связи.

Одна подходящая сеть включает в себя сервер и один или более клиентов; другие подходящие сети могут содержать другие сочетания серверов, клиентов и/или одноранговых узлов, а данная вычислительная система может функционировать как в качестве клиента, так и сервера. Каждая сеть включает в себя по меньшей мере два компьютера или две вычислительные системы, такие как сервер и/или клиенты. Вычислительная система может включать в себя рабочую станцию, портативный компьютер, отключаемый мобильный компьютер, сервер, мейнфрейм (mainframe), кластер, так называемый "сетевой компьютер" или "тонкий клиент", планшет, смартфон, персональный цифровой помощник или другое портативное вычислительное устройство, "умное" устройство или устройство бытовой электроники, медицинское устройство или сочетание вышеперечисленного.

Подходящие сети могут включать в себя коммуникационное или сетевое программное обеспечение, такое как программное обеспечение, доступное в компаниях Novell®, Microsoft® и у других поставщиков, и могут работать с использованием протоколов TCP/IP, SPX, IPX и других протоколов с использованием витой пары, коаксиального или волоконно-оптического кабелей, телефонных линий, радиоволн, спутников, радиорелейных станций, модулированных линий электропередачи переменного тока, передачи физических носителей и/или других "проводных" способов передачи данных, известных специалистам в данной области техники. Сеть может охватывать сети меньшего размера и/или может быть соединена с другими сетями посредством шлюза или подобного механизма.

Различные технологии или определенные аспекты или их части могут принимать вид программного кода (т.е. команд), встроенных в материальные носители, такие как гибкие дискеты, диски CD-ROM, жесткие диски, магнитные или оптические карты, твердотельные запоминающие устройства, энергонезависимый компьютерочитаемый носитель для хранения или любой другой машиночитаемый носитель для хранения, при этом, когда программный код загружается в машину и исполняется машиной, такой как компьютер, указанная машина становится устройством для практического осуществления различных способов. В случае исполнения программного кода программируемыми компьютерами вычислительное устройство может включать в себя процессор, носитель для хранения, читаемый процессором (включая энергозависимое и энергонезависимое запоминающее устройство и/или элементы хранения), по меньшей мере одно устройство ввода и по меньшей мере одно устройство вывода. Энергозависимое и энергонезависимое запоминающее устройство и/или элементы хранения могут быть ОЗУ, программируемым ПЗУ, флэш-диском, оптическим приводом, магнитным жестким диском или другим носителем для хранения электронных данных. Одна или более программ, которые могут реализовывать или использовать различные описанные в настоящем документе способы, могут использовать интерфейс прикладного программирования (API), повторно используемые элементы управления и т.п. Такие программы могут быть реализованы на высокоуровневом процедурном или объектно-ориентированном языке программирования для связи с вычислительной системой. Однако при необходимости программа(программы) может быть реализована на ассемблере или на машинном языке. В любом случае язык может быть компилируемым или интерпретируемым языком и может сочетаться с аппаратными реализациями.

Каждая вычислительная система включает в себя один или более процессоров и/или запоминающих устройств; вычислительные системы также могут включать в себя различные устройства ввода и/или устройства вывода. Процессор может включать в себя устройство общего назначения, такое как процессор Intel®, AMD® или другой "готовый" микропроцессор. Процессор может включать в себя устройство обработки специального назначения, такое как специализированная интегральная схема ASIC, SoC, SiP, FPGA, PAL, PLA, FPLA, PLD или другое индивидуализированное или программируемое устройство. Запоминающее устройство может включать в себя статическое ОЗУ, динамическое ОЗУ, флэш-память, один или более триггеров, ПЗУ, CD-ROM, DVD, жесткий диск, ленту или магнитный, оптический или другой компьютерный носитель для хранения. Устройство (устройства) ввода может включать в себя клавиатуру, "мышь", сенсорный экран, световое перо, планшет, микрофон, датчик или другие аппаратные средства с соответствующей прошивкой и/или программным обеспечением. Устройство (устройства) вывода может включить в себя монитор или другое отображающее устройство, принтер, речевой или текстовый синтезатор, переключатель, сигнальную линию или другие аппаратные средства с соответствующей прошивкой и/или программным обеспечением.

Следует понимать, что множество функциональных блоков, описанных в этой спецификации, могут быть реализованы в виде одного или более компонентов, что является выражением, используемым для более конкретного подчеркивания их независимости от реализации. Например, компонент может быть реализован в виде аппаратной схемы, содержащей заказные схемы сверхбольшой интеграции (СБИС) или матрицы вентилей, или готовые полупроводники, такие как логические чипы, транзисторы или другие дискретные компоненты. Компонент также может быть реализован в программируемых аппаратных устройствах, таких как программируемые пользователем вентильные матрицы, программируемые логические матрицы, программируемые логические устройства или тому подобное.

Компоненты также могут быть реализованы в программном обеспечении для исполнения процессорами различных типов. Идентифицированный компонент исполняемого кода может, например, содержать один или несколько физических или логических блоков компьютерных инструкций, которые могут быть, например, организованы как объект, процедура или функция. Тем не менее, исполняемые программы идентифицированного компонента не обязательно должны быть физически расположенным вместе, но могут содержать разрозненные инструкции, хранящиеся в разных местах, которые при логическом соединении вместе образуют компонент и достигают заявленной цели для компонента.

Действительно, компонент исполняемого кода может быть одиночной командой, или множеством команд, и даже может быть распределен по нескольким различным сегментам кода, между разными программами и между несколькими запоминающими устройствами. Точно так же рабочие данные могут быть идентифицированы и проиллюстрированы в настоящем документе в составе компонентов и могут быть реализованы в любой подходящей форме и организованы в структуре данных любого подходящего типа. Рабочие данные могут быть собраны как единый набор данных или могут быть распределены по разным местам, в том числе по разным устройствам хранения, и могут существовать, по крайней мере частично, просто как электронные сигналы в системе или сети.

Компоненты могут быть пассивными или активными, могут включать в себя агенты, способные выполнять необходимые функции.

Несколько аспектов описанных вариантов реализации будут показаны как программные модули или компоненты. Используемый в настоящем изобретении термин "программный модуль" или "компонент" может включать в себя машинную команду или исполняемый компьютером код любого типа, расположенные в запоминающем устройстве. Программный модуль, например, может включать в себя один или более физических или логических блоков машинных команд, которые могут быть организованы в виде подпрограммы, программы, объекта, компонента, структуры данных и т.п., которые выполняют одну или более задач или реализуют данные конкретных типов. Следует понимать, что программный модуль может быть реализован в виде аппаратных средств и/или встроенных программ вместо программного обеспечения или в дополнение к нему. Один или более из функциональных, описанных в настоящем изобретении, модулей может быть разделен на субмодули, и/или они могут быть объединены в один или несколько модулей.

В определенных вариантах реализации конкретный программный модуль может включать в себя различные команды, хранящиеся в различных ячейках запоминающего устройства, различных запоминающих устройствах или различных компьютерах, которые вместе реализуют описанные функциональные средства данного модуля. Действительно, модуль может включать в себя одиночную команду или множество команд и может быть распределен среди нескольких различных сегментов кода, среди различных программ и нескольких запоминающих устройств. Некоторые варианты реализации могут быть осуществлены в распределенной вычислительной среде, в которой задачи выполняются дистанционным обрабатывающим устройством, связанным через сеть связи. В распределенной вычислительной среде программные модули могут быть расположены в локальном и/или удаленном запоминающих устройствах хранения. Кроме того, данные, связываемые или предоставляемые вместе в записи базы данных, могут быть резидентными в том же самом запоминающем устройстве или в нескольких запоминающих устройствах, и могут быть связаны в полях записи в базе данных по сети.

Ссылка по всей этой спецификации на "пример", означает, что конкретный признак, структура или характеристика, описанные в соединении с указанным примером, включены по меньшей мере в один вариант реализации настоящего изобретения. Таким образом, появление фразы "в примере" в различных местах по всей этой спецификации не обязательно все относятся к тому же самому варианту реализации.

Как используется в настоящем документе, множество узлов, элементов структуры, композиционных элементов и/или материалов могут быть представлены в общем списке для удобства. Однако эти списки должны рассматриваться таким образом, как если каждый член списка индивидуально идентифицирован в качестве отдельного и уникального члена. Таким образом, никакой индивидуальный член такого списка не должен рассматриваться как фактический эквивалент какого-либо другого члена того же самого списка исключительно на основании его присутствия в общей группе, если не указано иное. Кроме того, различные варианты реализации и примеры настоящего изобретения могут быть упомянуты в настоящем документе наряду с альтернативами его различных компонентов.

Подразумевается, что такие варианты реализации, примеры и альтернативы не должны рассматриваться как фактические эквиваленты друг друга, но должны рассматриваться как отдельные и автономные представления настоящего изобретения.

Кроме того, описанные признаки, структуры или характеристики могут быть объединены любым подходящим способом в один или более вариант реализации. В последующем описании представлены множество конкретных подробностей, таких как примеры материалов, частот, размеров, длин, ширин, форм и т.п., для обеспечения полного понимания вариантов реализации настоящего изобретения. Однако специалист в данной области техники признает, что настоящее изобретение может быть осуществлено без одного или более конкретных подробностей или с использованием других способов, компонентов, материалов и т.п. В других случаях известные структуры, материалы или операции не показаны или опи-

саны подробно, чтобы не затемнять аспекты настоящего изобретения.

Следует признать, что описанные в настоящем документе системы включают в себя описания конкретных вариантов реализации. Эти варианты реализации могут быть объединены в автономные системы, частично объединены в другие системы, разбиты на множество систем или разделены или объединены другими способами. Кроме того, следует понимать, что параметры/признаки/аспекты/и т.п. одного варианта реализации могут использоваться в другом варианте реализации. Параметры/признаки/аспекты/и т.п. просто описаны в одном или более вариантах реализации для ясности, и понятно, что параметры/признаки/аспекты/и т.п. могут быть объединены с параметрами/признаками/аспектами/и т.п. другого варианта реализации или заменены, если конкретно не указано иное в настоящем документе.

Хотя вышеизложенное было описано достаточно подробно для целей ясности, очевидно, что могут быть сделаны некоторые изменения и модификации без отклонения от его принципов. Следует отметить, что существует множество альтернативных способов реализации описанных здесь процессов и устройств. Соответственно, настоящие варианты реализации следует считать иллюстративными, а не ограничивающими, и настоящее изобретение не должно ограничиваться подробностями, приведенными в настоящем документе, но может быть изменено в пределах объема охраны и эквивалентов приложенной формулы.

Специалистам в данной области техники понятно, что в подробности описанных выше вариантов реализации может быть внесено множество изменений без отклонения от основополагающих принципов настоящего изобретения. Следовательно, объем охраны настоящего изобретения должен определяться только приложенной формулой.

ФОРМУЛА ИЗОБРЕТЕНИЯ

1. Способ конфигурирования процессора, включающий:

прием программы, выполненной с возможностью исполнения в качестве программного обеспечения;

определение первой части программы для запуска в аппаратных средствах и второй части программы для запуска в качестве программного обеспечения;

определение, выполняемое на основании первой части, множества взаимосвязанных перепрограммируемых блоков операторов, содержащих одну или более функций преобразования, которые берут входные данные от шины предшествующих данных, выполняют одно или более преобразований в отношении входных данных и выводят преобразованные входные данные через шину выходных данных;

конфигурирование множества взаимосвязанных перепрограммируемых блоков операторов для исполнения одним или более перепрограммируемых процессоров;

исполнение второй части посредством одного или более входных интерфейсных процессоров;

передачу первых данных от одного или более входных интерфейсных процессоров к одному или более перепрограммируемых процессоров;

исполнение первой части посредством одного или более перепрограммируемых процессоров;

передачу вторых данных от одного или более перепрограммируемых процессоров к одному или более входных интерфейсных процессоров и

определение результирующих данных на основании первых данных и вторых данных.

2. Способ по п.1, согласно которому определение первой части дополнительно включает определение, выполняемое на основании вычислительной сложности, первой части программы для запуска в аппаратных средствах.

3. Способ по п.1, согласно которому результирующие данные являются результатом исполнения программы.

4. Способ по п.1, согласно которому вторые данные получают из первых данных на основании одного или более преобразований.

5. Способ по п.1, согласно которому результирующие данные получают из вторых данных, а вторые данные получают из первых данных.

6. Способ по п.1, согласно которому один или более перепрограммируемых процессоров является программируемой пользователем вентильной матрицей.

7. Способ по п.1, согласно которому передача первых данных от одного или более входных интерфейсных процессоров к одному или более перепрограммируемых процессоров дополнительно включает отправку первых данных через шину расширения от одного или более входных интерфейсных процессоров к одному или более перепрограммируемых процессоров.

8. Способ по п.1, согласно которому прием программы, выполненной с возможностью исполнения в качестве программного обеспечения, дополнительно включает компилирование программы в исполняемый код программного обеспечения, конфигурирование аппаратных средств и код связи для передачи данных между исполняемым кодом программного обеспечения и конфигурацией аппаратных средств.

9. Конвейерный процессор, используемый в способе по любому из пп.1-8 и содержащий:

шину входных данных;

множество взаимосвязанных перепрограммируемых блоков операторов, содержащих:

шину входных данных первого перепрограммируемого блока операторов, соединенную с шиной выходных данных второго перепрограммируемого блока операторов или шиной входных данных;

шину выходных данных первого перепрограммируемого блока операторов, соединенную с шиной входных данных третьего перепрограммируемого блока операторов или шиной выходных данных; и

одну или более функций преобразования, которые берут входные данные от шины предшествующих данных, выполняют одно или более преобразований в отношении входных данных и выводят преобразованные входные данные через шину выходных данных;и

шину выходных данных.

10. Конвейерный процессор по п.9, в котором ширина шины множества взаимосвязанных перепрограммируемых блоков операторов не является одинаковой.

11. Конвейерный процессор по п.9, в котором выход последующего блока является входом предыдущего блока.

12. Конвейерный процессор по п.9, в котором второй перепрограммируемый блок операторов и третий перепрограммируемый блок операторов являются одинаковыми.

13. Конвейерный процессор по п.9, в котором третий перепрограммируемый блок операторов предшествует первому перепрограммируемому блоку операторов в порядке исполнения.

14. Конвейерный процессор по п.9, дополнительно содержащий программный интерфейс, выполненный с возможностью приема команд для создания множества взаимосвязанных перепрограммируемых блоков операторов.

15. Система для обработки данных, содержащая:

множество конвейерных процессоров по любому из пп.9-14 и

функцию управления, выполненную с возможностью назначения данных каждому из указанного множества конвейерных процессоров.

16. Система по п.15, в которой функция управления дополнительно выполнена с возможностью реконфигурирования множества взаимосвязанных перепрограммируемых блоков операторов.

17. Система по п.15, в которой функция управления содержит команды по сохранению в запоминающем устройстве для создания множества взаимосвязанных перепрограммируемых блоков операторов указанного множества конвейерных процессоров.

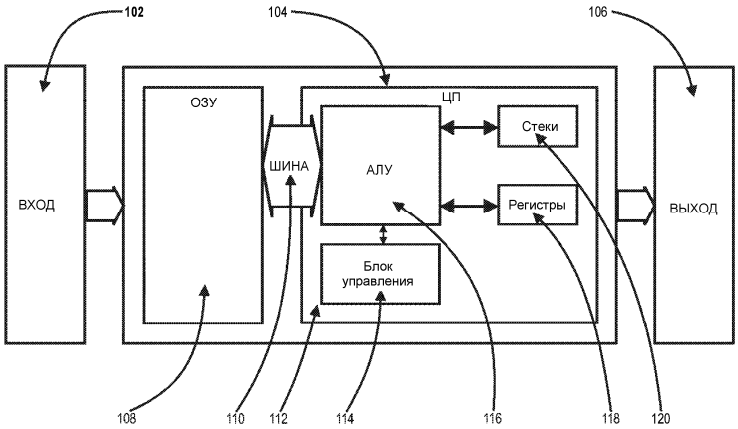
18. Система по п.15, дополнительно содержащая по меньшей мере один входной интерфейсный процессор с архитектурой, отличающейся от архитектуры указанного множества конвейерных процессоров.

19. Система по п.18, в которой по меньшей мере один входной интерфейсный процессор содержит процессор общего назначения.

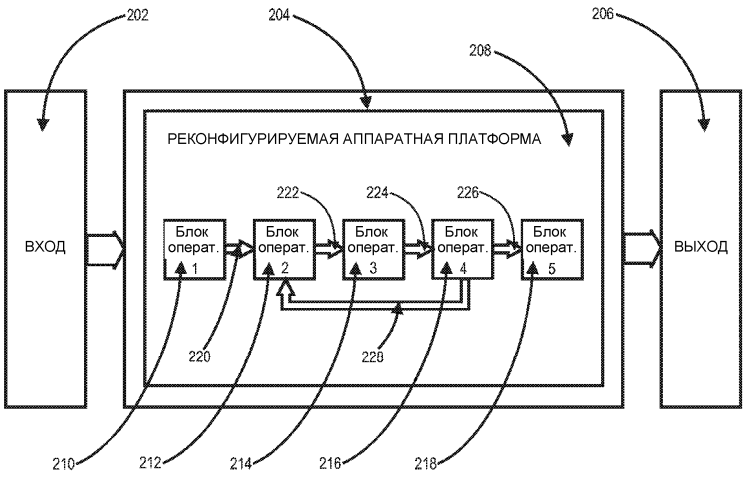
20. Система по п.15, в которой функция управления содержит:

защищенный интерфейс, выполненный с возможностью приема изменений конфигурации; и

незащищенный интерфейс, выполненный с возможностью назначения данных одному или более процессоров из указанного множества конвейерных процессоров.



Фиг. 1



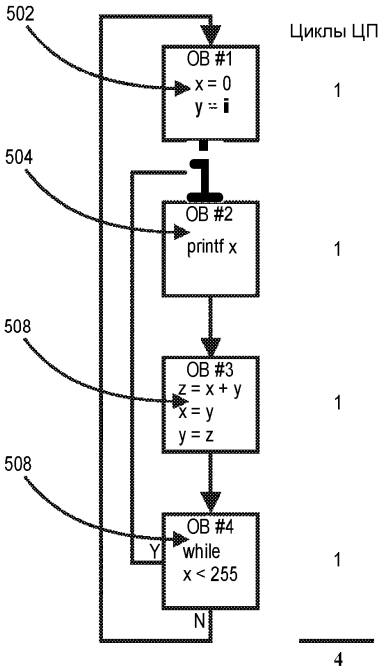
Фиг. 2

```
% cat fib.c
#include <stdio.h>

int main (void) {
        int x, y , z;

        While (1) {
                x = 0;
                y = 1;

                do {
                        printf("%d\n"

                        z = x + y;
                        x = y;
                        y = z;
                } while (x < 255);
        }
}
```

Фиг. 3

Исходный код С	Ячейка ОЗУ	Команда	Операнды	Циклы ЦП: Первая итерация	Циклы ЦП: Последующие итерации
	000000010000020	pusbq	%rbp	1	
Initialize	**000000010000021**	movq	%rsp, %rbp	3	
	0000000100000124	subq	$0x20, %rsp	4	
	000000010000028	movi	$0x8, -Qx4(%rbp)	7	
x = 0;	**0000000100000121**	mov!	$0x0, -Qx8(%rbp)	7	
y = i ;	0000000100000136	movi	$0x1, -0xc(%rbp)	7	
	000000010000013d	ieaq	0x56((%rip), %rd!	7	7
printir%d\n", x);	0000000100000144	movi	-0x8(%rbp), %esi	7	7
	0000000100000147	movb	$0x8, %al	3	3
	0000000100000149	cailq	0x1000000178	2	2
	000000010000014e	movi	-Qx8(%rbp), %esi	7	7
z = x + y;	0000000100000151	addi	-0xc(%rbp), %esi	2	2
	0000000100000154	movi	%esi, -0x1Q(%rbp)	3	3
x = y;	0000000100000157	movi	-0xc(%rbp), %esi	3	3
	000000010000015a	movi	%esl, -0x8(%rbp)	3	3
y = z;	0000000100000fSd	movi	-0x10(%rbp), %esi	3	3
	0000000100000160	movi	%esi, -0xc{%rbp)	3	3
	0000000100000163	movi	%eax, -Qx14(%rbp)	3	3
} while (x < 255);	000000010000026	cmpi	$0x!1, -0x8(%rbp)	3	3
	0000000100000fSd	jl	0x100000013d	7	7
}	0000000100000173	jmp	0x100000021		
				85	55

Фиг. 4



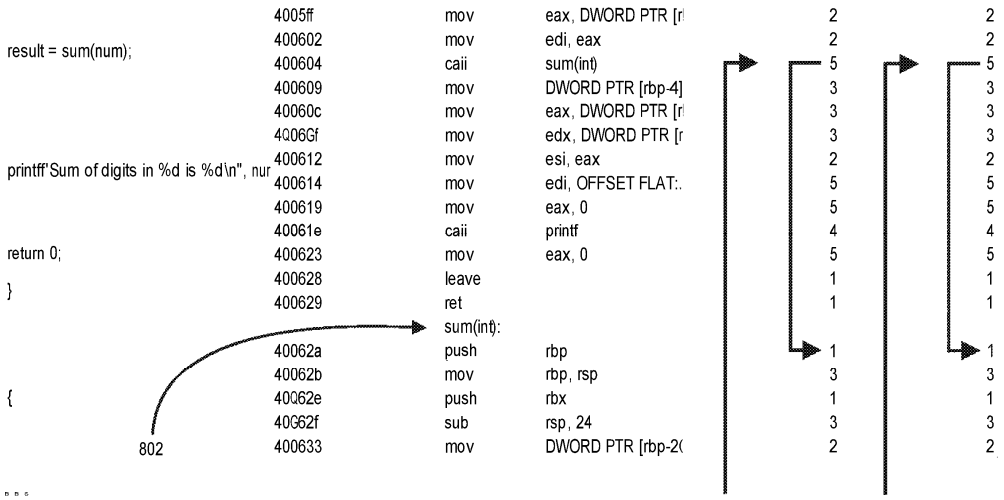
Фиг. 5

```
include <stdio.h>
intsum (int a);

int main()
{
        int num, result;
        printffEnter the number: ");
        scanf("%d", &num);
        result = sum(num);
        printff'Sum of digits in %d is %d\n", num, result);
        return 0;
}

intsum (int num)
{
        if (num != 0)
        {
                return (num % 10 + sum (num / 10));
        }
        else
        {
                return 0;
        }
}
```

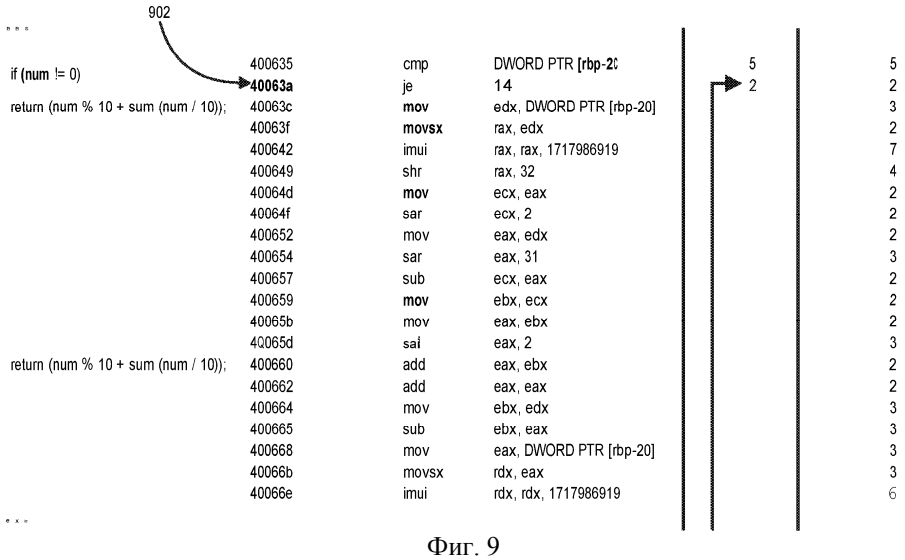
Фиг. 6

Исходный код С	Ячейка ОЗУ	Команда	Операнды	Циклы Оператор "если" = Истина	Оператор "если" = Ложь
		.LC0:			
		.string "Enter the number: "			
Initialize		.LC1:			
		.string "%d"			
		.LC2:			
		.string "Sum of digits in %d is %d\n"			
		main:			
	4005d2	push rbp	rbp	1	1
{	4005d3	mov	rbp, rsp	3	3
	4005d6	sub	rsp, 18	4	4
	4005da	mov	edi, OFFSET FLAT:.	5	5
printfffEnter the number: ");	4005df	mov	eax, 0	5	5
	4005e4	call	printf	5	5
	4005e9	lea	rax, [rbp-8]	4	4
	4005ed	mov	rsi, rax	2	2
scanf("%d", &num);	400510	mov	edi, OFFSET FLAT:.	5	5
	400515	mov	eax, 0	5	5
	40051a	cail	scant	5	5

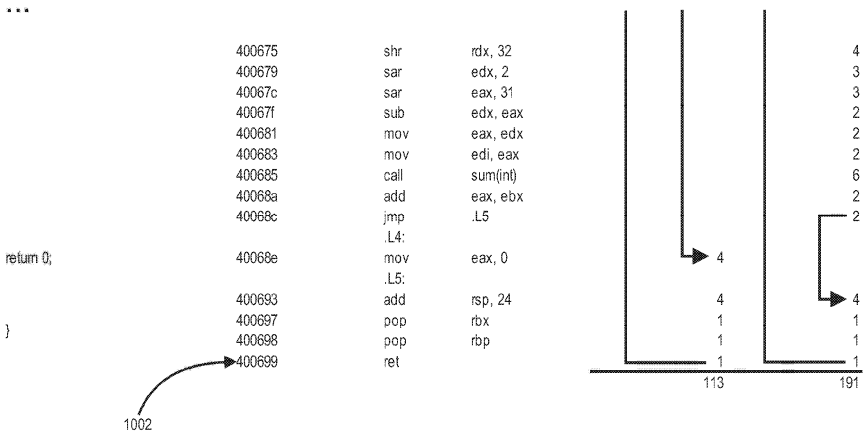
Фиг. 7



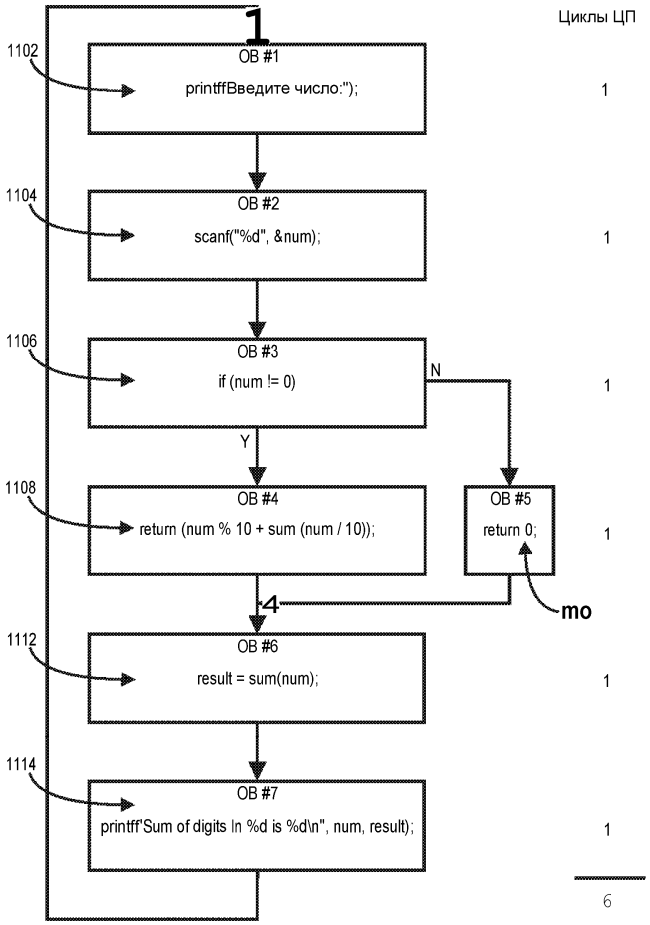
Фиг. 8



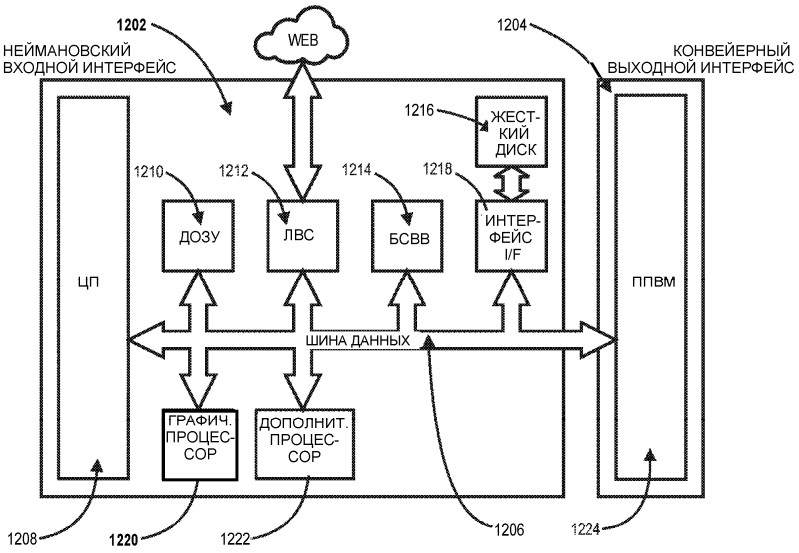
Фиг. 9


...
return 0;
}
400675 shr rdx, 32
400679 sar edx, 2
40067c sar eax, 31
40067f sub edx, eax
400681 mov eax, edx
400683 mov edi, eax
400685 call sum(int)
40068a add eax, ebx
40068c jmp .L5
.L4:
40068e mov eax, 0
.L5:
400693 add rsp, 24
400697 pop rbx
400698 pop rbp
400699 ret
1002
4 3 3 2 2 2 6 2 2 4 1 1 1 1
113 191

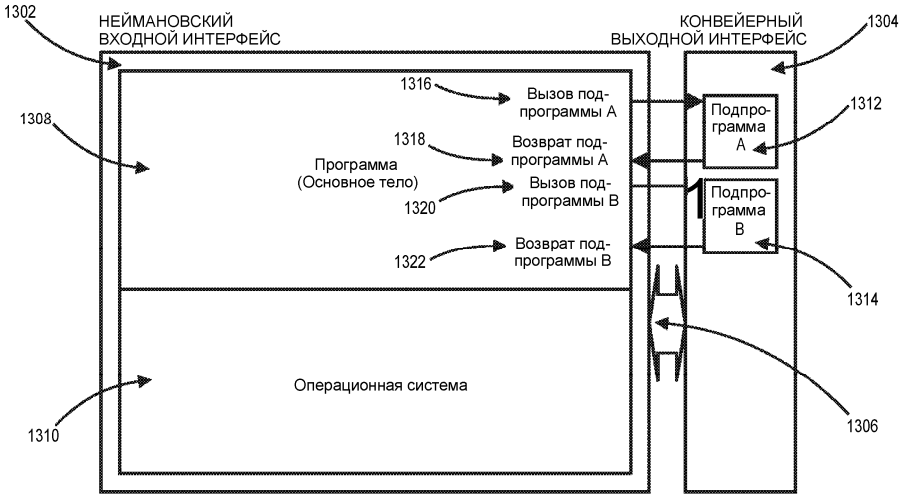
Фиг. 10


Циклы ЦП
1
1102
OB #1
printffВведите число:");
1
1104
OB #2
scanf("%d", &num);
1
1106
OB #3
if (num != 0)
N
1
Y
1108
OB #4
return (num % 10 + sum (num / 10));
OB #5
return 0;
1
4
mo
1112
OB #6
result = sum(num);
1
1114
OB #7
printff'Sum of digits In %d is %d\n", num, result);
1
6

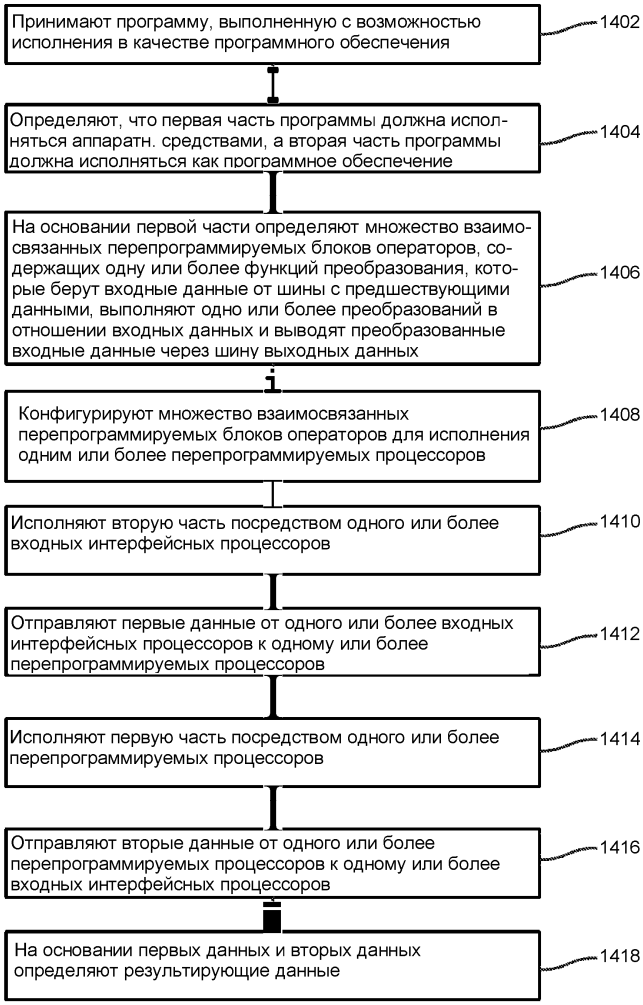
Фиг. 11


1202
НЕЙМАНОВСКИЙ ВХОДНОЙ ИНТЕРФЕЙС
WEB
1204
КОНВЕЙЕРНЫЙ ВЫХОДНОЙ ИНТЕРФЕЙС
1216
ЖЕСТ-КИЙ ДИСК
1210
1212
1214
1218
ДОЗУ
ЛВС
БСВВ
ИНТЕР-ФЕЙС I/F
ЦП
ППВМ
ШИНА ДАННЫХ
ГРАФИЧ. ПРОЦЕС-СОР
ДОПОЛНИТ. ПРОЦЕС-СОР
1208
1220
1222
1206
1224

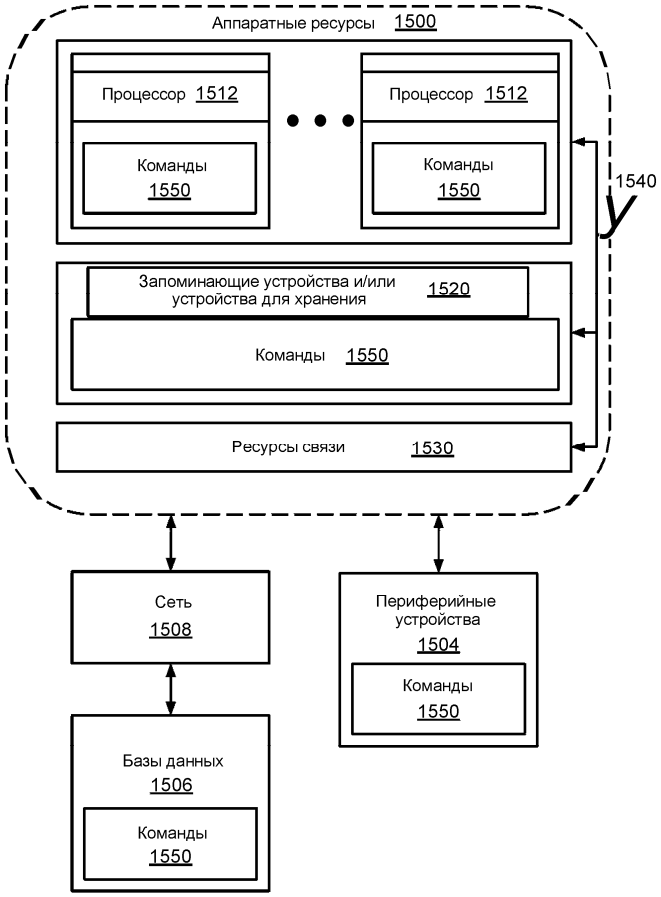
Фиг. 12


1302
НЕЙМАНОВСКИЙ ВХОДНОЙ ИНТЕРФЕЙС
КОНВЕЙЕРНЫЙ ВЫХОДНОЙ ИНТЕРФЕЙС
1304
1316
Вызов под-программы A
1308
1318
Возврат под-программы A
Программа (Основное тело)
1320
Вызов под-программы B
1322
Возврат под-программы B
Подпро-грамма A
1312
Подпро-грамма B
1314
Операционная система
1310
1306

Фиг. 13


Принимают программу, выполненную с возможностью исполнения в качестве программного обеспечения
1402
Определяют, что первая часть программы должна исполняться аппаратн. средствами, а вторая часть программы должна исполняться как программное обеспечение
1404
На основании первой части определяют множество взаимосвязанных перепрограммируемых блоков операторов, содержащих одну или более функций преобразования, которые берут входные данные от шины с предшествующими данными, выполняют одно или более преобразований в отношении входных данных и выводят преобразованные входные данные через шину выходных данных
1406
Конфигурируют множество взаимосвязанных перепрограммируемых блоков операторов для исполнения одним или более перепрограммируемых процессоров
1408
Исполняют вторую часть посредством одного или более входных интерфейсных процессоров
1410
Отправляют первые данные от одного или более входных интерфейсных процессоров к одному или более перепрограммируемых процессоров
1412
Исполняют первую часть посредством одного или более перепрограммируемых процессоров
1414
Отправляют вторые данные от одного или более перепрограммируемых процессоров к одному или более входных интерфейсных процессоров
1416
На основании первых данных и вторых данных определяют результирующие данные
1418

Фиг. 14


Аппаратные ресурсы 1500
Процессор 1512
Команды 1550
Процессор 1512
Команды 1550
1540
Запоминающие устройства и/или устройства для хранения 1520
Команды 1550
Ресурсы связи 1530
Сеть 1508
Базы данных 1506
Команды 1550
Периферийные устройства 1504
Команды 1550

Фиг. 15