РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 361 267** (13) **C2**

(51) МПК
G06F 12/00 (2006.01)
G06F 9/32 (2006.01)
G06F 9/04 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2004134452/09, 25.11.2004**

(24) Дата начала отсчета срока действия патента:
25.11.2004

(30) Конвенционный приоритет:
26.11.2003 US 10/723,823

(43) Дата публикации заявки: **10.05.2006**

(45) Опубликовано: **10.07.2009** Бюл. № **19**

(56) Список документов, цитированных в отчете о поиске: **US 2003/0200402 A1, 23.10.2003. RU 2182722 C2, 20.05.2002. US 6182195 B1, 30.01.2001. US 2002/0156989 A1, 24.10.2002. US 5956754 A, 21.09.1999. RU 2157000 C2, 27.09.2000.**

Адрес для переписки:
129090, Москва, ул. Б.Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", пат.пов. Ю.Д.Кузнецову, рег.№ 595

(72) Автор(ы):
КОУХЕН Эрнест С. (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ КОРПОРЕЙШН (US)

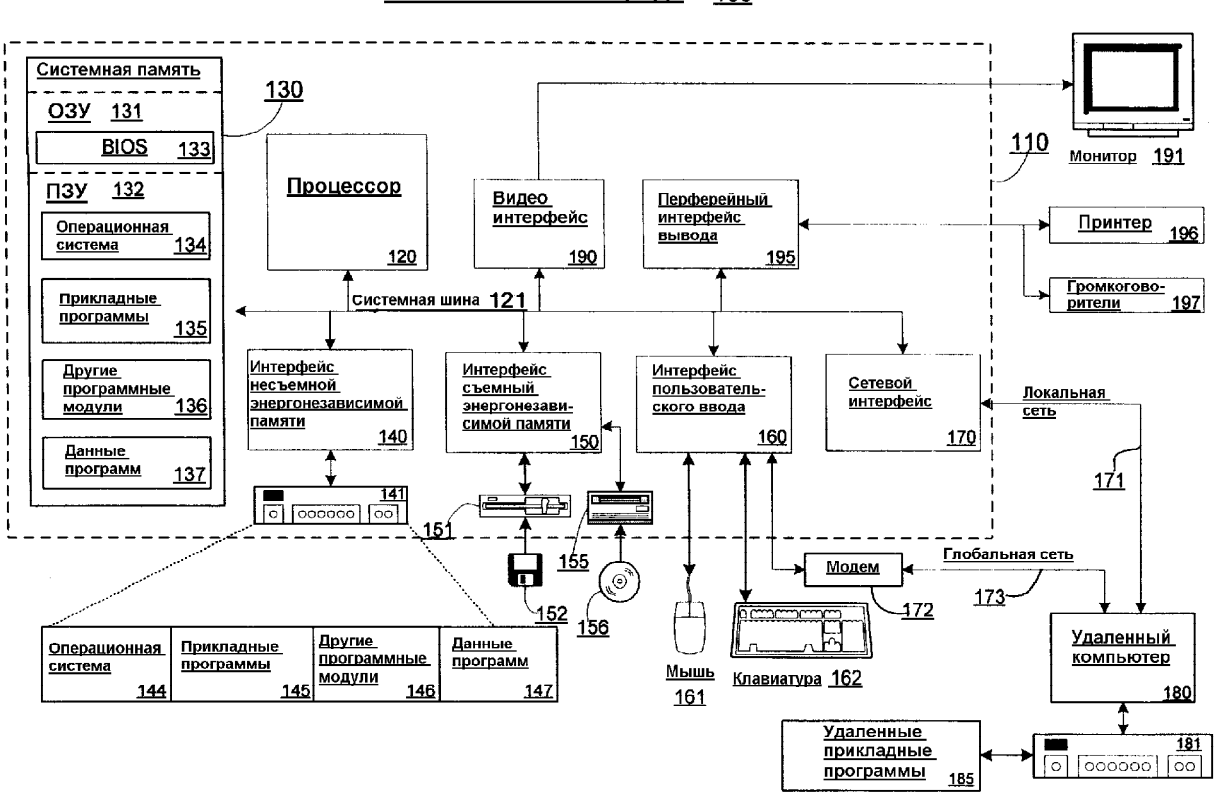
RU 2361267 C2

(54) ОТЛОЖЕННАЯ ОЧИСТКА БУФЕРОВ БЫСТРОГО ПРЕОБРАЗОВАНИЯ АДРЕСОВ

(57) Реферат:

Изобретение относится к области управления памятью, а более конкретно к очистке кэш-буфера преобразования адресов. Техническим результатом является расширение функциональных возможностей. Управление преобразованием адресов (ATC) ограничивает отображения между виртуальными и физическими адресами для того, чтобы реализовать политику доступа к памяти. Каждый процессор в многопроцессорной системе поддерживает буфер быстрого преобразования адресов (TLB), который кэширует отображения для ускорения преобразования виртуальных адресов. Каждый процессор также поддерживает счетчик. Каждый раз, когда TLB процессора очищается, счетчик процессора увеличивается. Когда ссылка на страницу удаляется из карты преобразования адресов, значение счетчиков для всех процессоров записывается. Когда процессор осуществляет доступ к странице, записанные значения счетчиков сравниваются с текущим значением счетчика процессора для определения того, очищался ли TLB процессора с того момента, когда ссылка на страницу была удалена из карты. Затратная операция очистки TLB откладывается до тех пор, пока это не потребуется, но, тем не менее, осуществляется достаточно заблаговременно для предотвращения того, чтобы недействительная запись в TLB была использована для нарушения политики доступа. 4 н. и 7 з.п. ф-лы, 6 ил.

RU 2361267 C2



Фиг.1

RU 2361267 C2

RU 2361267 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 361 267** (13) **C2**

(51) Int. Cl.
G06F 12/00 (2006.01)
G06F 9/32 (2006.01)
G06F 9/04 (2006.01)

(12) ABSTRACT OF INVENTION

(21), (22) Application: **2004134452/09, 25.11.2004**

(24) Effective date for property rights:
25.11.2004

(30) Priority:
26.11.2003 US 10/723,823

(43) Application published: **10.05.2006**

(45) Date of publication: **10.07.2009 Bull. 19**

Mail address:
129090, Moskva, ul. B.Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", pat.pov. Ju.D.Kuznetsovu, reg.№ 595

(72) Inventor(s):
KOUKhEN Ehrnest S. (US)

(73) Proprietor(s):
MAJKROSOFT KORPOREJShN (US)

RU 2361267 C2

(54) DELAYED CLEARING OF TRANSLATION LOOKASIDE BUFFERS

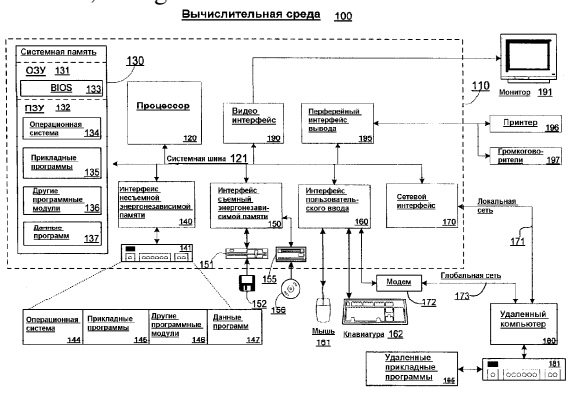
(57) Abstract:

FIELD: physics; computer engineering.

SUBSTANCE: invention pertains to memory control, more specifically to clearing address translation cache-buffers. Address translation control (ATC) limits imagery between virtual and physical addresses, so as to implement the memory access policy. Each processor in a multiprocessor system supports a translation lookaside buffer (TLB), which caches imagery to speed up translation of virtual addresses. Each processor also supports a counter. Each time the TLB of a processor is cleared, the processor counter increases. When a link on page is deleted from the address translation map, the counter value for all processors is recorded. When the processor accesses the page, the recorded counter values are compared with the current counter value of the processor to determine whether the processor TLB has been cleared since the link on the page was deleted from the map. The costly operation of clearing TLB is delayed until it is required, but, it is however done in time to prevent use of invalid records in TLB to violate access policy.

EFFECT: increase of functional capabilities.

11 cl, 6 dwg



Фиг.1

RU 2361267 C2

ОБЛАСТЬ ТЕХНИКИ, К КОТОРОЙ ОТНОСИТСЯ ИЗОБРЕТЕНИЕ

Данное изобретение относится, в общем случае, к области управления памятью и более конкретно к очистке кэш-буфера преобразования адресов.

ПРЕДШЕСТВУЮЩИЙ УРОВЕНЬ ТЕХНИКИ

Большинство компьютерных систем имеет механизм виртуальной адресации, с помощью которого виртуальные адреса отображаются на физические адреса. Когда запрос на доступ к ячейке памяти сделан с использованием виртуального адреса, виртуальный адрес преобразуется в соответствующий физический адрес целевой ячейки памяти, к которой запрашивается доступ. Совокупность таблиц преобразования адресов определяет отображение виртуальных адресов на физические адреса. Таблицы преобразования адресов обычно хранятся в памяти, так как преобразование адреса требует доступа к памяти для того, чтобы считать таблицы. Эта операция доступа к памяти, требующаяся для чтения таблиц, является дополнительной к операции доступа, подлежащей выполнению в отношении целевой ячейки. Таким образом, когда используется виртуальная адресация, число доступов к памяти, производимых системой, может удваиваться по отношению к их числу, которое имело бы место, если бы все запросы на доступ были сделаны с помощью физических адресов. Некоторые виртуальные адреса бывают многоуровневыми в том смысле, что они требуют отображения, осуществляющего разыменование по стадиям, что означает, что необходимо использовать два или более доступов к памяти для выполнения преобразования адресов (что приводит к увеличению в три или более раз, числа доступов к памяти, которое требуется для выполнения до конца одного базового запроса а доступ).

Для того чтобы уменьшить число доступов к памяти, которые должны иметь место для преобразования адреса, многие системы виртуальной адресации используют тип кэш-буфера, называемый буфер быстрого преобразования адреса (TLB). Так как к страницам памяти, доступ к которым осуществлялся ранее, с большой вероятностью может быть осуществлен доступ в ближайшее время, после того как таблицы преобразования адресов использовались для преобразования дескриптора виртуальной страницы в местоположение физической страницы, соответствие между виртуальной и физической страницей кэшируется в TLB. Каждый раз, когда требуется произвести преобразование адреса, TLB проверяется на предмет определения того, есть ли в ней кэшированное отображение для страницы, на которой расположен запрашиваемый блок памяти. Если соответствующее отображение было ранее кэшировано в TLB, тогда используется кэшированная копия; в противном случае адрес преобразуется на основе таблиц преобразования. Поскольку доступ к TLB быстрее, чем доступ к таблицам преобразования в памяти, использование TLB повышает производительность в случае, когда последовательные доступы к памяти расположены на одной группе страниц, что является обычным случаем.

Буферы TLB создают несколько дополнительных проблем, когда виртуальная память используется для обеспечения защиты памяти. Посредством защиты памяти стараются принудительно обеспечить политику управления безопасностью, которая управляет тем, какие программные компоненты могут осуществлять определенные виды доступа (например, чтение, запись) к определенным страницам физической памяти; эта защита может быть обеспечена виртуальной памятью с помощью управления редактированием преобразования виртуальных адресов в физические. (Это управление может обеспечиваться либо операционной системой, которая создает отображения, либо системой управления преобразованием адресов (ATC), которая

фильтрует изменения к таким отображениям.) Однако когда преобразование адресов изменяется, старые отображения могут все еще существовать в TLB. Таким образом, когда таблицы преобразования адресов редактируются, с тем чтобы аннулировать некоторые права доступа к странице для некоторых программных компонент, компонента может сохранять доступ к странице до тех пор, пока эти старые отображения не будут очищены из буферов TLB. Обычный путь добиться этого - заставлять любые соответствующие TLB производить очистку как часть подобной операции.

Однако очистка TLB - затратная операция, особенно для микропроцессоров с разделяемой памятью. Каждый процессор, который может содержать устаревшее отображение, нарушающее новую политику безопасности, должен получить сигнал о необходимости очистить свой TLB; этот сигнал обычно требует относительно медленного межпроцессорного прерывания (IPI). Кроме того, очистка сама по себе относительно затратная операция.

Имея в виду все вышесказанное, необходим механизм для преодоления недостатков современного уровня техники.

СУЩНОСТЬ ИЗОБРЕТЕНИЯ

Данное изобретение обеспечивает механизм для поддержки отложенной очистки буферов TLB. Для каждого TLB поддерживается счетчик и каждый раз, когда TLB очищается, значение счетчика увеличивается. В многопроцессорной системе, где каждый процессор поддерживает свой собственный TLB, может быть отдельный счетчик для каждого процессора, при этом счетчик для конкретного процессора увеличивается, когда процессор очищает свой TLB. Когда возникает инициирующее событие, значения любых из соответствующих счетчиков записывается после того, как инициирующее событие завершается. «Инициирующее событие» - это событие, которое затрагивает карту преобразования адресов или политику доступа к памяти, которой эта карта подчиняется, таким образом, что устаревшие записи в TLB могут стать причиной нарушений политики; соответствующий счетчик является счетчиком TLB, который, возможно, содержит такие устаревшие записи. Для примера, заданная страница памяти может быть объявлена запрещенной в соответствии с политикой безопасности, и все отображения на эту страницу могут быть удалены из карт преобразования адресов; это удаление ссылок на страницу из карты является примером инициирующего события, и счетчики для любых буферов TLB, которые могут содержать отображение на эту страницу, являются релевантными. Может случиться, что пройдет довольно много времени между возникновением инициирующего события и моментом, когда эффект или результат этого инициирующего события фактически будет использован в процессе преобразования адресов. (Для примера, отображения на заданную страницу памяти могут измениться в заданный момент времени, но миллионы операций могут иметь место перед любой попыткой произвести повторное использование отсоединенной страницы.) Когда результат инициирующего события используется, сохраненные значения счетчиков сравниваются с текущими значениями счетчиков для того, чтобы определить, какие (если такие вообще есть) буферы TLB могли быть не очищены с тех пор, когда произошло инициирующее событие. Если какие-либо соответствующие TLB имеют значения счетчиков, совпадающие с их сохраненными значениями, то все такие TLB были очищены в обычном порядке.

Необходимо отметить, что TLB надежен, т.е. доподлинно свободен от устаревших записей TLB, которые могут стать результатом нарушения применяемой политики

доступа, если значение счетчика изменилось, и TLB ненадежен, если значение счетчика не изменилось. Таким образом, безопасно использовать любой размер счетчика и изменять счетчик произвольным образом (или даже вообще не изменять его) при очистке. Другой возможностью является использование счетчика реального времени для создания временной метки для времени последней очистки, если часы синхронизированы в пределах некоторого временного отклонения, инициатор очистки может легко сохранить время, когда преобразование было модифицировано, и предположить завершение всех очисток, временные метки которых превышают время модификации преобразования, по меньшей мере, на упомянутое временное отклонение.

В системе, где среды с высокой степенью надежности и без высокой степени надежности («правая сторона» и «левая сторона» соответственно) существуют бок о бок и в которой каждая очистка TLB требует перехода на правую сторону, механизм данного изобретения может быть использован, чтобы избежать излишних переходов на правую сторону. Для примера, каждое изменение между правой и левой стороной может привести к очистке TLB, и счетчик для каждого процессора может увеличиваться каждый раз, когда процессор перемещается от правой стороны к левой. Другими словами, поскольку правая сторона достоверно производит (или проверяет) очистку TLB, счетчик отражает последнее значение времени, когда TLB мог включать в себя записи, которые могли бы нарушить схему управления доступом к памяти. Каждый раз, когда страница делается недоступной (или запрещенной для записи) для левой стороны, эта страница становится, по существу, «помеченной временной меткой» со значением счетчика(ов). Когда к странице процессор осуществляет доступ, определяется: (1) находится ли процессор в текущий момент на правой стороне, либо (2) очистил ли процессор достоверным образом свой TLB с момента последнего изменения статуса страницы. Последний из двух вариантов определения может быть осуществлен с помощью удостоверения в том, что значение счетчика процессора больше, чем сохраненное значение для этой страницы.

Другие признаки настоящего изобретения описаны ниже.

ПЕРЕЧЕНЬ ФИГУР ЧЕРТЕЖЕЙ

Приведенное выше краткое описание, также как и следующее ниже подробное описание предпочтительных вариантов осуществления изобретения, будет лучше понимаемо при чтении вместе с приложенными чертежами. С целью иллюстрации изобретения на чертежах показаны примерные схемы изобретения; однако изобретение не ограниченно конкретными способами и раскрытыми инструментальными средствами. На чертежах:

Фиг.1 - блок-схема примерной вычислительной среды, в которой аспекты изобретения могут быть реализованы;

Фиг.2 - блок-схема примерной системы виртуальной адресации;

Фиг.3 - блок-схема двух сред, которые сосуществуют в машине и соответствующая часть памяти которых защищена;

Фиг.4 - блок-схема, показывающая буфер быстрого преобразования адресов (TLB), используемый в процессе преобразования адресов;

Фиг.5 - блок-схема системы, которая включает в себя множество процессоров, где каждый процессор ассоциирован с TLB и счетчиком;

Фиг.6 - блок-схема последовательности операций процесса для отложенной очистки TLB в соответствии с аспектами данного изобретения.

ПОДРОБНОЕ ОПИСАНИЕ ПРИМЕРНЫХ ВАРИАНТОВ ОСУЩЕСТВЛЕНИЯ

Обзор управления преобразованием адресов (ATC) может использоваться для реализации политики доступа к памяти посредством динамического управления отображениями, используемыми для преобразования виртуальных адресов в физические. Когда буфер быстрого преобразования адресов (TLB) используется для кэширования отображений, старые отображения, которые стали несовместимы с текущим состоянием политики доступа, могут все еще сохраняться в TLB, предоставляя при этом память для использования в противоречии с политикой. TLB может быть очищен, но очистка TLB является затратной операцией, которую предпочтительно производить не чаще, чем необходимо. Данное изобретение обеспечивает механизм для отложенной очистки TLB, который позволяет отложить очистку TLB до тех пор, пока в ней не возникнет необходимость. Механизм предпочтительно использовать в мультипроцессорных системах, в которых каждый процессор поддерживает свой собственный TLB, который может быть очищен независимо от буферов TLB других процессоров.

ИЛЛЮСТРАТИВНАЯ КОМПЬЮТЕРНАЯ СИСТЕМА

Фиг.1 показывает иллюстративную вычислительную среду, в которой аспекты изобретения могут быть реализованы. Среда 100 вычислительной системы является только одним примером подходящей вычислительной среды, и не подразумевается, что она налагает какие-либо ограничения на объем использования или функциональные возможности изобретения. Вычислительная среда 100 не должна интерпретироваться как имеющая какие-либо зависимости или требования в отношении любой компоненты или комбинации компонент, приведенных в иллюстративной операционной среде 100.

Изобретение применимо с множеством других сред или конфигураций вычислительных систем общего или специального назначения. Примеры широко известных вычислительных систем, сред и/или конфигураций, которые подходят для использования с изобретением, включают в себя, но не в ограничительном смысле, персональные компьютеры, серверные компьютеры, карманные или переносные устройства, многопроцессорные системы, основанные на микропроцессорах системы, телевизионные приставки, программируемую бытовую электронику, сетевые персональные компьютеры (ПК), мини-компьютеры, универсальные компьютеры (мейнфреймы), встраиваемые системы, распределенные вычислительные среды, включающие в себя любые из описанных выше устройств или систем, и тому подобное.

Изобретение может быть описано в общем контексте машиноисполняемых команд, таких как программные модули, выполняемые на компьютере. Обычно программные модули включают в себя процедуры, программы, объекты, компоненты, структуры данных и т.д., которые выполняют конкретные задания или реализуют определенные абстрактные типы данных. Изобретение может также применяться в распределенных вычислительных средах, где задания выполняются с помощью удаленных устройств обработки данных, которые связаны вместе сетью связи или другой средой передачи данных. В распределенной вычислительной среде программные модули и другие данные могут находиться как на локальных, так и на удаленных компьютерных носителях данных, включая запоминающие устройства.

В соответствии с Фиг.1 иллюстративная система для реализации изобретения включает в себя вычислительное устройство общего назначения в форме компьютера 110. Компоненты компьютера 110 могут включать в себя, но не в ограничительном смысле, устройство 120 обработки данных (процессор), системную память 130 и системную шину 121, которая соединяет различные системные

компоненты, включая системную память, с устройством 120 обработки данных. Устройство 120 обработки данных может представлять собой несколько логических модулей обработки данных, таких как поддерживаемые в многопоточном процессоре. Системная шина 121 может относиться к любому из нескольких типов шинных структур, включающих в себя шину памяти или контроллер памяти, периферийную шину и локальную шину, с использованием любой из многообразия шинных архитектур. Для примера, но не ограничения, такие архитектуры включают в себя шину Архитектуры Промышленного Стандарта (ISA), шину Микроанальной Архитектуры (MCA), расширенную шину ISA (EISA), локальную шину Видео Электронной Ассоциации Стандартов (VESA) и шину Межсоединения Периферийных Устройств (PCI) (известную также, как мезонинная шина). Системная шина 121 может также быть реализована, как двухточечное соединение, устройство коммуникации или нечто подобное, из между числа осуществляющими связь устройствами.

Компьютер 110 обычно включает в себя различные машиночитаемые носители информации. Машиночитаемые носители информации могут быть любыми возможными носителями, к которым компьютер 110 может осуществить доступ, и включают в себя энергозависимые и энергонезависимые, съемные и несъемные носители. Для примера, но не ограничения, машиночитаемые носители информации могут включать в себя компьютерные носители информации и среды передачи. Компьютерные носители информации включают в себя как энергонезависимые, так и энергозависимые, как съемные, так и несъемные, реализованные любым способом или с помощью любой технологии для хранения информации, такой как машиночитаемые команды, структуры данных, программные модули или другие данные. Компьютерные носители включают в себя, но не в ограничительном смысле, оперативное запоминающее устройство (ОЗУ,RAM), постоянное запоминающее устройство (ПЗУ, ROM), электрически стираемое перепрограммируемое ПЗУ (EEPROM), флеш-память или память другой технологии, ПЗУ на компакт-диске (CD-ROM), универсальные цифровые диски (DVD) или другие оптические дисковые носители информации, магнитные кассеты, магнитные ленты, магнитные дисковые носители или другие магнитные устройства для хранения информации, или любые другие носители, которые могут использоваться для хранения желаемой информации и к которым компьютер 110 может осуществить доступ. Среды передачи обычно воплощают машиночитаемые команды, структуры данных, программные модули или другие данные в модулированном информационном сигнале, таком как несущая волна или другой механизм передачи, и включают в себя любые среды доставки информации. Термин «модулированный информационный сигнал» обозначает сигнал, одна или более характеристик которого установлены или изменены так, чтобы обеспечить кодирование информации в этом сигнале. Для примера, но не ограничения, среды передачи включают в себя проводные среды, такие как проводная сеть или прямое проводное соединение, и беспроводные среды, такие как акустические, радиочастотные и другие беспроводные среды. Комбинации любых из вышеперечисленных носителей и сред также охватываются понятием «машиночитаемый носитель информации».

Системная память 130 включает в себя компьютерные носители информации в форме энергозависимых и/или энергонезависимых запоминающих устройств, таких как постоянное запоминающее устройство (ПЗУ) 131 и оперативное запоминающее устройство (ОЗУ) 132. Базовая система ввода/вывода 133 (BIOS), включающая в себя базовые процедуры, помогающие передаче информации между элементами внутри

компьютера 110, например во время запуска, обычно хранится в ROM 131. RAM 132 обычно содержит данные и/или программные модули, которые оперативно доступны для устройства 120 обработки данных и/или обрабатываются устройством 120 обработки данных в текущий момент. Для примера, но не ограничения, Фиг.1 показывает операционную систему 134, прикладные программы 135, другие программные модули 136 и данные 137 программ.

Компьютер 110 может также включать в себя другие съемные/несъемные компьютерные носители информации. Только для примера Фиг.1 показывает накопитель 140 на жестких магнитных дисках, который считывает с несъемного энергонезависимого магнитного носителя или записывает на него, дисковод 151 для магнитного диска, который считывает или со съемного энергонезависимого магнитного диска 152 записывает на него и дисковод 155 для оптического диска, который считывает со съемного энергонезависимого извлекаемого, непостоянного оптического диска 156, такого как CD-ROM или другой оптический носитель, или записывает на него. Другие съемные/несъемные, энергозависимые/энергонезависимые компьютерные носители информации, которые можно использовать в иллюстративной операционной среде, включают в себя, но не в ограничительном смысле, кассеты с магнитной лентой, карты флеш-памяти, универсальные цифровые диски, цифровую видеоленту, твердотельные ОЗУ, твердотельные ПЗУ и тому подобное. Накопитель 141 на жестких магнитных дисках обычно подсоединяется к системной шине 121 с помощью интерфейса энергонезависимой несъемной памяти, такого как интерфейс 140, а дисковод 151 для магнитного диска и дисковод 155 для оптического диска обычно подсоединяются к системной шине 121 с помощью интерфейса энергонезависимой съемной памяти, такого как интерфейс 150.

Накопители, дисководы и используемые ими компьютерные носители информации, рассмотренные выше и показанные на Фиг.1, позволяют хранить машиночитаемые команды, структуры данных, программные модули или другие данные для компьютера 110. На Фиг.1, для примера, накопитель 141 на жестких магнитных дисках показан как хранящий операционную систему 144, прикладные программы 145, другие программные модули 146 и данные 147 программ. Следует отметить, что эти компоненты могут либо быть одинаковыми, либо различаться от операционной системы 134, прикладных программ 135, других программных модулей 136 и 137 программ. Операционной системе 144, прикладным программам 145, другим программным модулям 146 и данным 147 программ присвоены другие ссылочные номера, чтобы проиллюстрировать, что они, как минимум, являются другими копиями. Пользователь может вводить команды и информацию в компьютер 20 с помощью устройств ввода, таких как клавиатура 162 и координатно-указательное устройство, обычно являющееся «мышью», шаровым манипулятором или сенсорной панелью.

Другие устройства ввода (не показанные здесь) могут включать в себя микрофон, джойстик, игровую панель, спутниковую тарелку, сканнер или подобные им устройства. Эти и другие устройства ввода часто подсоединяются к устройству обработки данных 120 с помощью интерфейса 160 пользовательского ввода, связанного с системной шиной, но могут также подсоединяться посредством других структур интерфейсов и шин, таких как параллельный порт, игровой порт или универсальная последовательная шина (USB). Монитор 191 или другой тип устройства отображения также подсоединяется к системной шине 121 через интерфейс, такой как видеоинтерфейс 190. В добавлении к монитору компьютеры могут также включать

другие периферийные устройства вывода, такие как громкоговорители 197 и принтер 196, которые могут подсоединяться через периферийный интерфейс вывода 195.

Компьютер 110 может работать в сетевой среде, используя логические соединения с одним или более удаленных компьютеров, таких как удаленный компьютер 180. Удаленный компьютер 180 может быть персональным компьютером, сервером, маршрутизатором, сетевым ПК, одноранговым устройством или другим общим сетевым узлом и, обычно, включает в себя многие или все элементы, описанные выше по отношению к компьютеру 110, хотя только запоминающее устройство 181 показано на Фиг.1. Логические соединения, изображенные на Фиг.1, включают в себя локальную сеть (LAN) 171 и глобальную сеть (WAN) 173, но также могут включать в себя и другие сети. Такие сетевые среды широко распространены в офисах, сетях масштаба предприятия, интрасетях и сети Интернет.

При использовании в сетевой среде LAN компьютер 110 соединяется с LAN через сетевой интерфейс или адаптер 170. При использовании в сетевой среде WAN компьютер 110 обычно включает в себя модем 172 или другие средства, предназначенные для установления связи через WAN, такую как Интернет. Модем 172, который может быть внутренним или внешним, может подсоединяться к системной шине 121 через интерфейс 160 пользовательского ввода или другое подходящее средство. В сетевой среде программные модули, показанные в отношении к компьютеру 110, или их части могут храниться в удаленном запоминающем устройстве. Для примера, но не ограничения, Фиг.1 показывает удаленные прикладные программы 185 как постоянно хранящиеся на запоминающем устройстве 181. Важно отметить, что сетевые соединения показаны как пример, и другие средства установления линии связи между компьютерами могут быть также использованы.

ИЛЛЮСТРАТИВНАЯ СХЕМА ВИРТУАЛЬНОЙ АДРЕСАЦИИ

Фиг.2 показывает пример системы виртуальной адресации. Пример, показанный на Фиг.2, является схемой виртуальной адресации страничного типа, хотя понятно, что виртуальная адресация может основываться на других моделях, например сегментной. Схема, показанная на Фиг.2, является двухуровневой схемой адресации, такой как одна из схем виртуальной адресации, доступных на процессоре INTEL x86. Схема является двухуровневой в том смысле, что необходимо использовать два уровня преобразования адресов для преобразования идентификатора виртуальной страницы в физическую страницу, как показано ниже.

В этой страничной схеме каталог 202 страниц включает в себя набор записей. Примерная структура записи более детально описана ниже, при рассмотрении Фиг.3, но, по существу, каждая запись определяет физическое местоположение (т.е. номер страничного блока или «PFN») конкретной таблицы страниц, такой как таблица 204(1), 204(2), 204(3). Каждая таблица страниц в свою очередь включает в себя набор записей, каждая из которых определяет физическое местоположение (опять же номер страничного блока) конкретной страницы данных, такой как страница 206(1), 206(2), 206(3) или 206(4). Страницы данных представляют собой смежные части оперативной памяти 132 заданной длины. Страницы данных могут хранить любые типы данных, и, как необходимо отметить здесь, в добавление к хранению обычных данных, страницы данных также используются для хранения содержимого каталога 202 страниц и страниц 204(1) - 204(3). Таким образом, заданная страница может быть каталогом, таблицей, страницей данных или иметь несколько ролей в

качестве любой комбинации этих трех структур.

Схема виртуальной адресации, изображенная на Фиг.2, является двухуровневой схемой виртуальной адресации, поскольку для того, чтобы определить местонахождение конкретной страницы, надо проверить как каталог страниц (уровень 1), так и таблицу страниц (уровень 2). Специалистам в данной области техники должно быть понятно, что можно построить схему виртуальной адресации с произвольным числом уровней, и при этом принципы данного изобретения могут быть применены ко всем таким схемам виртуальной адресации. Как известно, процессор INTEL x86 поддерживает виртуальную адресацию имеющую один, два или три уровня и обычно применяет «гибридную» схему, в которой «малые» страницы (т.е. страницы размером четыре килобайта) используют двухуровневые виртуальные адреса, а «большие» страницы (т.е. страницы размером четыре мегабайта) используют одноуровневые виртуальные адреса.

На страничной схеме по Фиг.2 любой байт на странице может быть идентифицирован посредством виртуального адреса 210, включающего в себя смещение 211 каталога страниц (PD), смещение 212, таблицы страниц (PT) и смещение 213 страницы. Таким образом, для определения физического адреса модуль 220 управления памятью (MMU), предназначенный для преобразования адресов, использует смещение 211 каталога страниц для определения положения конкретной записи в каталоге страниц 202. Для примера, смещение 211 может быть равным нулю, что служит индикатором того, что надо обратиться к нулевой записи в каталоге 202 страниц. Эта запись содержит PFN, в котором хранится таблица страниц, так что MMU 220 использует этот PFN для определения положения одной из таблиц страниц (т.е. таблицы 204(1) страниц). MMU 220 далее использует смещение 212 таблицы страниц как индекс для идентифицированной таблицы страниц и извлекает запись, найденную при этом смещении. Запись содержит PFN страницы данных (т.е. страницы 206(1)), так что MMU 220 добавляет сдвиг 213 страницы к базовому адресу идентифицированной страницы для того, чтобы определить положение конкретного байта физической памяти. MMU 220 может быть также выполнен с возможностью осуществления других различных функций в добавление к простому преобразованию адресов, например, MMU 220 может загружать страницу с диска, если запись этой страницы в таблице помечена как «нет в наличии»; MMU 220 может запрещать доступ на запись, если страница помечена как «только для чтения», и т.д.

В виртуальной схеме адресации по Фиг.2 местоположение (т.е. PFN) самого каталога страниц сохраняется в ячейке 201 запоминающего устройства. MMU 220 использует содержимое этой ячейки для определения положения каталога 202 страниц, когда начинает преобразование виртуального адреса 210. Таким образом, может существовать множество карт страниц, и конкретная карта может быть выбрана для текущего использования посредством задания содержимого ячейки 201 запоминающего устройства таким оразом, чтобы она содержала PFN каталога страниц заданной карты. В примере для процессора INTEL x86, ячейка 201 запоминающего устройства соответствует регистру с именем CR3.

СРЕДА С ВЫСОКОЙ СТЕПЕНЬЮ НАДЕЖНОСТИ И ЗАЩИТА ПАМЯТИ ПОСРЕДСТВОМ УПРАВЛЕНИЯ ПРЕОБРАЗОВАНИЕМ АДРЕСОВ

Необходимо отметить, что одним из признаков системы виртуальной адресации, описанной выше, является то, что для заданной карты преобразования памяти возможен физический адрес, не отображенный в какой-либо виртуальный адрес. Таким образом, в системе, такой как процессор INTEL x86, где почти все запросы на

доступ к памяти производятся с помощью виртуального адреса, можно сделать часть физической памяти запрещенной к доступу для заданного источника посредством гарантирования того, что карта преобразования адресов данного источника не указывает на запрещенную для доступа память. Реализация управления картой преобразования адресов для обеспечения такой запрещенной для доступа памяти называется Управлением Преобразованием Адресов (ATC). Запрещенная для доступа часть памяти часто называется «защищенной памятью», и ATC является одним из путей обеспечения защищенной памяти.

Одним из вариантов использования защищенной памяти является тот случай, когда среды с высокой степенью надежности и без высокой степени надежности сосуществуют в одном устройстве. Среда с высокой степенью надежности может быть обеспечена применением защищенной памяти, к которой среда без высокой степени надежности не может осуществить доступ. Концепция среды с высокой степенью надежности и ее взаимоотношения с защищенной памятью описаны ниже при рассмотрении Фиг.3.

Фиг.3. показывает две среды, сосуществующие на одном устройстве: среда 360 с высокой степенью надежности и среда 350 без высокой степени надежности. Для удобства эти среды названы как правая сторона (RHS) и левая сторона (LHS) соответственно; RHS является средой с высокой степенью надежности, а LHS - средой без высокой степени надежности. «С высокой степенью надежности» означает для среды, что имеется высокий уровень гарантии того, что функции, выполняющиеся в этой среде, будут выполняться корректно. Таким образом, LHS 350 исполняет функции 306(1), 306(2),... 306(n), а RHS 360 исполняет различные функции 308(1), 308(2),... 308(m). LHS 350 ассоциирована со спецификацией 302, которая определяет, то, как функции 306(1) - 306(n) должны себя вести. Кроме того, RHS 360 ассоциирована со спецификацией 304, которая определяет, то, как функции от 308(1) - 308(m) должны себя вести. Спецификации 302 и 304 могут быть или не быть записаны. Для примера, LHS 350 может быть коммерческой операционной системой, которая поставляется с детальным руководством, разъясняющим, как ее различные службы, драйвера, системные вызовы и т.д. должны себя вести. Или может быть просто общее понимание того, как данная среда должна вести себя, и это понимание может составлять спецификацию.

Независимо от того, какую форму спецификация принимает, должно быть понятно, что почти все программное обеспечение включает в себя как известные, так и необнаруженные ошибки, лючки (входы в программу, минующие стандартные процедуры проверки, используемые программистами для отладки, и не удаленные в рабочей версии), логические ошибки и т.д., которые могут стать причиной непредвиденного поведения программного обеспечения. Однако возможно оценить уровень надежности в отношении того, что заданная часть программного обеспечения будет вести себя ожидаемым образом, т.е. уровень гарантии того, что поведение программного обеспечения будет фактически совпадать с описанным в спецификации. В примере по Фиг.3. RHS 360 является средой «с высокой степенью надежности» в отношении того, что существует относительно высокий уровень гарантии того, что функции 308(1)-308(m), для выполнения которых RHS 360 спроектирована, будут фактически исполняться в соответствии со спецификацией 304. Напротив, LHS 350 является средой «без высокой степени надежности» в отношении того, что существует относительно низкий уровень гарантии того, что LHS 350 будет исполнять функции 306(1)-306(n) в соответствии со спецификацией 302. «Относительно

высокий» и «относительно низкий» в данном контексте обозначает, что уровень гарантии того, что RHS 360 будет вести себя в соответствии со спецификацией 304, выше, чем уровень гарантии того, что LHS 350 будет вести себя в соответствии со спецификацией 302.

В общем случае полнофункциональная коммерческая операционная система, такая как одна из операционных систем MICROSOFT WINDOWS, является средой без высокой степени надежности. Такие среды поддерживают открытую архитектуру, в которой драйверы устройств, встраиваемые элементы и другие типы расширений могут быть свободно добавлены, что делает трудным для проверки поведение такой операционной системы с учетом всех возможных условий. Более того, когда желательно применение некоторых видов безопасности, полезно запускать такую полнофункциональную операционную систему параллельно с операционной системой с высокой степенью надежности. Операционная система с высокой степенью надежности является малой операционной системой, обеспечивающей малый набор функций. Поскольку функциональные возможности, обеспечиваемые операционной системой с высокой степенью надежности невелики, имеется малое количество переменных, оказывающих воздействие на ее функционирование, так что поведение такой системы легче проверить на соответствие высокой степени доверия.

В предпочтительном варианте осуществления изобретения среда с высокой степенью надежности включает в себя защищенную память, т.е. часть памяти, которая недоступна из среды без высокой степени надежности. Так, RHS 360 может хранить секретные данные (например, криптографические ключи) в защищенной памяти, не опасаясь того, что есть риск чтения или записи в эту память процессом, работающим в LHS 350. Для примера, спецификация 304 может обеспечивать, что RHS 360 имеет возможность защитить информацию от подделок извне, и защищенная память позволяет RHS 360 осуществлять эту функцию. Более того, код, используемый RHS 360 для осуществления ее различных функций, может храниться в защищенной памяти для того, чтобы препятствовать процессам, работающим в LHS 350, осуществлять перезапись этого кода другим кодом (который может привести RHS 360 к поведению, не соответствующему ее спецификации). Фиг.3 показывает защищенную память 312, доступную для использования RHS 360. Физическое адресное пространство 310 включает в себя все ячейки физической памяти, доступные на заданном вычислительном устройстве. Защищенная память 312 является подмножеством этой совокупности ячеек. Как показано на Фиг.3, RHS 360 имеет доступ ко всему физическому адресному пространству 310, но у LHS 350 отсутствует доступ к той части физического адресного пространства 310, которая составляет защищенную память 312. (Необходимо отметить, что хотя Фиг.3 показывает защищенную и незащищенную части адресного пространства как смежные, нет жесткого требования в отношении такой смежности. Более того, нет необходимости для RHS 360 иметь доступ ко всему физическому адресному пространству или для LHS 350 иметь доступ ко всем частям физического адресного пространства вне защищенной памяти 312.)

Как указанно выше, есть несколько систем, в которых почти все запросы на доступ к памяти делаются по физическим адресам. Одним из путей реализации защищенной памяти 312 в такой системе является управление содержимым карт преобразования адресов таким образом, чтобы для LHS 350 не было предоставлено ни одного виртуального адреса для защищенной памяти 312. (Когда есть потенциально несколько видов запросов на доступ, определяющих свою цель с помощью физических адресов, доступ к защищенной памяти может быть ограничен неким

вспомогательным совместно функционирующим средством, таким как вектор исключения, фильтрующий запросы на доступ к физической памяти от устройств с прямым доступом к памяти или от других источников.) Существует несколько алгоритмов, доступных для обеспечения того, чтобы LHS 350 не могла использовать карту преобразования адресов, которая бы вела к защищенной памяти, но главная идея всех этих алгоритмов состоит в том, что (1) когда процессор работает в LHS 350, обеспечивается то, что любая карта, загруженная в CR3 (ячейка 201 запоминающего устройства, показанная на Фиг.2), не ведет к страницам, содержащимся в защищенной памяти 312; и (2) для любой попытки внести изменения в активную карту в LHS 350 оценивается предполагаемое изменение так, чтобы в результате изменение не привело к ссылке на страницу защищенной памяти 312.

Далее представлен пример алгоритма для реализации защищенной памяти с помощью ATC:

Пусть D1 - набор страниц, которые могут использоваться как каталоги страниц. Пусть D2 - набор страниц, которые могут использоваться как таблицы страниц. Пусть D - объединение D1 и D2 (т.е. D=D1 U D2). Каждая запись в каталоге страниц или таблице страниц, помеченная как «имеется в наличии» (т.е. бит присутствия каждой установлен), называется «ссылкой». Страница в D2 является «используемой для записи», если имеется малая ссылка записи-чтения из некоторой страницы в D1 на страницу в D2. («Малая» ссылка - это ссылка из каталога на таблицу, т.е. ссылка в каталоге, которая, в конечном счете, приведет к малой странице. «Большая» ссылка - это ссылка в каталоге, которая указывает на большую страницу.) Исходим из того, что существует политика, определяющая страницы, в которые определенным объектам разрешено выполнять доступ для чтения и/или записи.

Поддерживаются следующие инварианты:

- CR3 находится в D1;
- все страницы в D1 и D2 доступны для чтения согласно соответствующей политике;
- каждая малая ссылка из страницы в D1 указывает на страницу в D2;
- ссылки из страниц в D2 указывают на страницу, доступную для чтения согласно соответствующей политике;
- каждая ссылка записи-чтения из используемой для записи страницы в D2 указывает на страницу, доступную для записи согласно соответствующей политике и не находящуюся в D;
- каждая малая страница, содержащаяся в большой странице, являющейся целью большой ссылки из страницы в D1, доступна для чтения согласно соответствующей политике; если ссылка является ссылкой записи-чтения, тогда малая страница также доступна для записи согласно соответствующей политике и не находится в D.

Если, для примера, политика определяет защищенную память 312 как недоступную, тогда поддержка описанных выше инвариантов для всех таблиц преобразования адресов, используемых LHS 350, будет обеспечивать тот факт, что ни один из запросов на доступ, который идентифицирует свою цель посредством виртуального адреса и возникает в LHS 350, не сможет когда-либо достичь защищенной памяти 312.

ИСПОЛЬЗОВАНИЕ БУФЕРОВ БЫСТРОГО ПРЕОБРАЗОВАНИЯ АДРЕСОВ С ATC

Буфер быстрого преобразования адресов (TLB) является, по существу, кэш-буфером, в котором хранятся данные отображений. Основной идеей, лежащей в основе TLB, является то, что к страницам, к которым недавно осуществлялся доступ, по всей вероятности снова осуществят доступ и поэтому недавно использованные

отображения из заданной виртуальной страницы на соответствующую физическую страницу сохраняются в TLB. Вычисление отображения на основе таблицы преобразования адресов (по меньшей мере, в наиболее общей двухуровневой схеме виртуальной адресации, используемой в процессоре INTEL x86) требует двух доступов к памяти: один доступ к каталогу страниц для нахождения местоположения требуемой таблицы страниц, и второй доступ к таблице страниц для нахождения требуемой страницы данных. Доступ к фактическому целевому местоположению, к которому требуется получить доступ, может быть осуществлен лишь после двух (других) доступов к памяти, необходимых для определения физического местоположения искомых данных. TLB уменьшает необходимость в таких дополнительных доступах к памяти посредством обеспечения быстрой памяти, в которой поддерживается информация о недавно использованных отображениях. Таким образом, для каждого преобразования адреса, которое требуется осуществить, процессор вначале обращается к своему TLB для определения того, кэшированы ли данные отображений для заданной виртуальной страницы в TLB. К таблице преобразования адресов обращаются, если данные отображений отсутствуют в TLB.

Фиг.4 показывает, как процессор использует TLB 402. Процессор 120 исполняет часть кода и во время этого исполнения появляется команда на чтение или запись в целевую ячейку 406 памяти 400. Команда не идентифицирует физический адрес целевой ячейки 406, точнее определяет виртуальный адрес целевой ячейки 406. Таким образом, процессор 120 (или, во многих случаях, модуль управления памятью, связанный с процессором 120) должен преобразовать виртуальный адрес в физический адрес. Процессор 120 вначале просматривает TLB для определения того, кэшировано ли отображение, которое можно использовать для преобразования виртуального адреса (окружность 1). Если такое отображение существует, тогда это отображение используется для преобразования адреса. Если отображения не существует, то соответствующие части карты преобразования адресов 404 считываются из памяти 400 (окружность 2), и эти части карты используются для преобразования виртуального адреса. Независимо от того, как адреса преобразуются, преобразованный адрес используется для доступа к физической целевой ячейке 406 (окружность 3). Основанием того, что TLB работает эффективнее, является тот факт, что быстрее отыскать информацию в TLB 402, чем считать карту преобразования адресов 404 из памяти 400. Дополнительно в некоторых реализациях данные, хранящиеся в TLB 402, могут свести двухшаговый процесс преобразования в один шаг, т.е. использование карты преобразования адресов требует чтения каталога страниц (PD) и таблицы страниц (PT) на разных шагах; однако одна комбинация сдвига PD/сдвига PT может быть кэширована в TLB 402, позволяя, таким образом, идентифицировать виртуальную страницу посредством комбинации сдвига PD/сдвига PT за один шаг.

Когда буфера TLB используются в системе ATC, инварианты ATC модифицируются так, чтобы принимать во внимание преобразования, доступные через TLB. Для примера, в ранее описанном примере алгоритма ATC определение ссылки может быть изменено на следующее: ссылка с одной страницы на другую существует, если она либо существует в физической памяти, либо кэширована в некотором TLB. Хотя содержимое буферов TLB не доступно напрямую для просмотра в большинстве архитектур (например, в процессоре x86), их содержимое может быть ограниченно, потому что преобразования входят в буфера TLB только через память. Используя инварианты в иллюстративном алгоритме, буфера TLB необходимо очищать при

удалении страницы из D1 или D2, отказе от доступа на чтение или чтение/запись к странице, и записи или добавлении в D2 данных, изменяющих статус страницы с используемой для записи на неиспользуемую для записи или наоборот.

Однако реальных стимулов уменьшить число очисток TLB еще больше, поскольку очистка TLB является затратной операцией, снижающей производительность. Во-первых, TLB повышает эффективность только в том смысле, что в нем находятся полезные данные отображений, и очистка TLB удаляет эти данные отображений из TLB. Во-вторых, реальная очистка сама по себе может быть затратной процедурой по следующим соображениям: в архитектуре, описанной на Фиг. 3, поддержка защищенной памяти 312 зависит от отсутствия отображений, нарушающих политику. Поскольку отсутствие таких отображений, арушающих политику, в предпочтительном варианте осуществления изобретения является элементом обеспечения среды с высокой степенью надежности, управление отображениями должно само по себе осуществляться в среде с высокой степенью надежности. Таким образом, не только само управление картой преобразования адресов должно осуществляться в среде с высокой степенью надежности, но также и очистка TLB должна осуществляться (или, по меньшей мере, проверяться) в среде с высокой степенью надежности. Следовательно, каждая очистка TLB, на которую полагаются как на часть системы ATC, требует, чтобы текущая среда была изменена с LHS 350 на RHS 360. Процесс изменения контекста из одной среды в другую является, сам по себе, затратным, что увеличивает затраты на стоимость очистки. Следовательно, желательно избегать очистки TLB чаще, чем это необходимо.

Данное изобретение предлагает механизм для того, чтобы отложить и, по возможности, избежать некоторых из этих очисток. Изобретение в одном варианте осуществления основывается на двух субмеханизмах. Первый субмеханизм предусматривает, что операции, не изменяющие фактическую карту памяти (например, добавление страницы в D2 или изменение статуса страницы на недоступную для записи), в концептуальном плане могут быть отложены до тех пор, пока их завершение не потребуется для разрешения некой модификации в памяти. Для примера, действие по отказу от доступа на чтение к странице (в политике безопасности) может быть отложено до тех пор, пока не появится возможность модификации этой страницы (когда другой вычислительный объект создает отображение чтения/записи к этой странице). Второй субмеханизм использует временные метки для более точного отслеживания того, должно ли существовать конкретное отображение к странице в конкретном TLB.

В примерном алгоритме, представленном выше, отложенные операции, требующие очистки, могут быть разделены на два класса: те, что требуют удаления из буферов TLB любых преобразований, позволяющих странице быть прочитанной (отказ от доступа на чтение к странице или удаление страницы либо из D1, либо D2), и те, что требуют удаления из буферов TLB любых преобразований, допускающих изменение страницы (удаление доступа на запись для страницы). Назовем первые из них «операции очистки по чтению» и вторые «операции очистки по записи» и определим, что эти операции могут быть отложены в следующих случаях:

- операции очистки по чтению могут быть отложены до тех пор, пока некий вычислительный объект не попробует создать отображение для записи/чтения к транице, или до тех пор, пока некое устройство не сможет модифицировать страницу с помощью непосредственного доступа к памяти, или пока страница не будет модифицирована самим алгоритмом ATC;

- операции очистки по записи могут быть отложены до тех пор, пока некий вычислительный объект не попробует создать отображение для чтения или записи/чтения к странице, или до тех пор, пока некое устройство не сможет модифицировать страницу с помощью непосредственного доступа к памяти, или пока страница не будет модифицирована самим алгоритмом ATC.

Когда происходит операция очистки по чтению, необходимо очистить TLB, только если он еще потенциально содержит отображение чтения к рассматриваемой странице. Это может быть только в том случае, когда TLB не очищался со времени последнего отображения чтения к этой странице. Аналогично, когда происходит операция очистки по записи, необходимо очистить TLB, только если он не очищался с момента последнего отображения чтения-записи к странице.

Для определения того, был ли TLB очищен с момента последнего отображения чтения/чтения-записи к странице, мы поддерживаем две временные метки для каждой страницы (ее временная метка чтения и временная метка записи) и одну временную метку для каждого TLB (его временная метка очистки). Всякий раз, когда последнее отображение чтения/чтения-записи к странице удаляется, текущее время записывается в ее временную метку записи/записи-чтения. Всякий раз, когда TLB очищается, текущее время записывается в его временную метку очистки.

Время может измеряться разными способами. В первом примерном подходе часы используются только для временной метки очистки TLB. Когда TLB очищается, временная метка очистки изменяется (не важно каким образом). Если временная метка чтения/чтения-записи на странице отличается от временной метки очистки, отображение чтения/чтения-записи гарантированно не должно находиться в TLB. Если, по совпадению, временная метка очистки изменилась (возможно, несколько раз), но случайно совпадает с временной меткой чтения/чтения-записи (например, если счетчик циклически вернулся к началу), ущерба не будет, будет только лишняя очистка.

Во втором примерном подходе время измеряется с помощью часов реального времени или виртуальных часов. Если часы, используемые для записи временных меток чтения/чтения-записи и временных меток очистки, синхронизированы в пределах некоторого временного отклонения, отображение чтения/чтения-записи гарантированно не должно находиться в TLB, если временная метка чтения/чтения-записи превосходит метку очистки, по меньшей мере, на величину временного отклонения. Одним из путей полностью избежать отклонения часов является использование глобальных часов, совместно используемого счетчика, который увеличивается, когда любой TLB очищается, однако такие часы могут стать наиболее часто используемыми в системе.

Первый подход имеет то преимущество, что он не требует синхронизированных часов. Однако если необходимо рассматривать большое число TLB, то первый подход требует отдельных временных меток чтения/чтения-записи для каждого TLB, тогда как второй подход требует только одного считывания часов. Более того, это исключает конфликты памяти из-за часов. Два подхода можно комбинировать, и также можно использовать другие механизмы (например, векторные часы).

Данное изобретение предлагает механизм, допускающий отложенную очистку TLB, при которой явная очистка TLB откладывается до тех пор, пока не появится потенциальная возможность того, LHS 350 фактически осуществит доступ к странице, используя старую запись в TLB, которая больше не согласуется с картой преобразования адресов. По существу, когда последнее отображение на страницу

удалено, страница является «помеченной временной меткой». (Как обсуждалось выше, «временные метки» могут фактически содержать не время, а значение логических счетчиков.) Когда к странице позднее осуществляется доступ, определяется, следует ли очищать TLB в зависимости от того, был ли очищен TLB с момента, когда страница была помечена временной меткой.

В многопроцессорных компьютерных системах, таких как на Фиг.5, каждый из процессоров 120(1)-120(n) поддерживают свой собственный TLB 402(1)-402(n). Другими словами, каждый из процессоров 120(1)-120(n) может осуществлять доступ к одним и тем же базовым таблицам преобразования адресов, но они могут раздельно сохранять в кэше отображения из этих таблиц. В дополнение, в одном из вариантов осуществления каждый процессор может работать в LHS 350 или RHS 360. В соответствии с изобретением счетчики 502(1)-502(n) ассоциированы с соответствующим процессором 120(1)-120(n). Каждый раз, когда среда процессора меняется от RHS 360 на LHS 350, счетчик, соответствующий процессору, увеличивается.

Всякий раз, когда страница становится недоступной для LHS 350 (т.е. когда страница становится недоступной для LHS 350 в соответствии с надлежащей политикой доступа и отсоединяется от всех отображений, доступных LHS 350), страница эффективным образом помечается временной меткой, соответствующей значениям счетчиков. (Счетчики не отсчитывают «время» в физическом смысле, а могут быть предствлены как некий вид хронометража, поскольку счетчики увеличиваются в прямом направлении в ответ на возникновение некоторых событий.) Таким образом, посредством сравнения сохраненных значений счетчиков для страницы с текущим значением счетчика для процессора, который пытается осуществить доступ к странице, можно определить, вошел ли процессор в RHS 360 со времени, когда страница стала недоступной для LHS 350. (Как объяснялось выше, вхождение в RHS 360 служит причиной надежной очистки TLB.) Если процессор не входил в RHS 360 с момента, когда ссылки на страницу были удалены, тогда процессор входит в RHS 360 и очищает свой TLB. Если попытка доступа к странице произошла в LHS 350, тогда процессор возвращается в LHS 350 и пытается снова осуществить доступ к странице. Если процессор вошел в RHS 360 с момента удаления ссылок на страницу (т.е. если процессор в настоящий момент находится в RHS 360 или вошел в RHS 360 в некоторый момент после того, как ссылки на страницу были удалены), тогда становится известно, что TLB была надежно очищена с последнего момента, когда страница была доступна для LHS 350, и, таким образом, TLB не может содержать каких либо отображений на эту страницу, так что нет необходимости очищать TLB в этот раз.

Фиг.6 представляет блок-схему последовательности операций процесса, с помощью которого TLB очищается после того, как страница стала недоступной для LHS 350. На начальном этапе процесса, показанного на Фиг.6, предполагается, что заданная страница (страница X) доступна для осуществления доступа из LHS 350 в соответствии с примененной политикой доступа. В некий момент страница X становится недоступной для LHS 350 в соответствии с политикой доступа, и карты преобразования адресов, доступные для LHS 350, изменяют так, чтобы гарантировать, что ссылки на страницу X удалены из этих карт(602). В то время, когда ссылки на страницу X удалены, значения счетчиков для всех процессоров на вычислительном устройстве записываются, и эту запись связывают со страницей X для последующего извлечения (604). Запись предпочтительно содержит текущее значение счетчика для каждого процессора на вычислительном устройстве.

После того, как ссылки на страницу X удаляются из всех карт, доступных для LHS 350, некоторый промежуток времени может пройти перед тем, как к странице X фактически будет осуществлен доступ. В некоторый момент, однако, запрос на доступ к странице X будет порожден некоторым процессором (процессором Y) (606). Записанные значения счетчиков, связанные со страницей, извлекают, и сохраненное значение для процессора Y сравнивают с текущим значением счтчика процессора Y. (Эта запись должна также включать в себя сохраненные значения счетчиков для других процессоров, но эти значения игнорируются при определении того, требуется ли процессору Y очищать свой TLB.) Если текущее значение счетчика процессора Y больше, чем сохраненное значение счетчика, тогда доступ к странице разрешается (614) без дальнейших запросов, поскольку это тот случай, когда процессор Y входил в RHS 360 и очищал свой TLB с момента, когда страница X была сделана недоступной для LHS 350. С другой стороны, если сохраненное значение счетчика для процессора Y такое же, как текущее значение счетчика процессора Y, тогда в RHS 360 не было вхождения с момента, когда страница X была удалена, что означает, что TLB процессора Y может все еще включать в себя отображение на страницу X.

Если процессор Y работает в RHS 360 (610), тогда TLB не может содержать отображения на страницу X без нарушения политики доступа, поэтому доступ к странице X разрешен (614). Если, однако, процессор Y работает в LHS 350 (и известно из 608, что процессор Y пока не очищал свой TLB с момента удаления ссылок на страницу X), возможно, что содержимое TLB процессора Y предоставляет для LHS 350 отображение на страницу X. Таким образом, процессор Y входит в RHS 360 и очищает свой TLB (612). После того, как TLB будет очищен, процессор Y возвращается к LHS 350, и запрос на доступ к странице выполняется снова (614) (тем самым, требуя повторное преобразование запрашиваемого адреса для теперь пустого TLB).

Необходимо отметить, что установка страницы в недоступное для LHS 350 состояние не является единственным событием, которое может вызвать необходимость очистки TLB, и механизм данного изобретения может быть также использован для инициирования очистки TLB по ряду других причин. Для примера, удаление страницы из любого из множеств D1 или D2 (как определено выше) или отказ от доступа на чтение (или запись) для страницы может также дать толчок к необходимости очистить TLB, поскольку эти события могут повлиять на законность отображений в соответствии с имеющейся политикой. В таком случае механизм данного изобретения может быть использован способом, подобным описанному выше, т.е. значения счетчиков могу быть записаны, когда страница удаляется из D1 или D2 или когда имеет место отказ от доступа на чтение (или запись), и любая последующая попытка процессора осуществить доступ к странице может инициировать сравнение между значением счетчика процессора и сохраненным значением счетчиков для этой страницы.

Необходимо далее отметить, что механизмы данного изобретения не ограничены случаем, когда очистка требует перехода к RHS 360, или даже случаем, когда как LHS 350, так и RHS 360 доступны. В более общем случае механизмы данного изобретения могут быть использованы в любом случае, когда есть возможность, что в TLB существуют утратившие силу записи, более не соответствующие состоянию карты, например, в случае, когда обычный процесс изоляции произведен операционной системой.

Надо отметить, что упомянутые выше примеры были предоставлены единственно

для намерения разъяснить и не могут быть истолкованы как ограничение данного изобретения. Поскольку изобретение описывалось со ссылками на различные варианты его осуществления, понятно, что слова, использованные в данном описании, являются словами описаний и иллюстраций, а не словами ограничения. Далее, несмотря на то что изобретение описано здесь со ссылками на конкретные способы, материалы и варианты осуществления, не подразумевается, что изобретение ограниченно деталями, раскрытыми здесь; напротив, изобретение охватывает все функционально эквивалентные структуры, способы и использования, попадающие под объем, определяемый прилагаемой формулой изобретения. Специалисты в данной области техники при изучении идей этого описания, смогут делать различные изменения к данному изобретению без отклонения от объема и сущности изобретения во всех его аспектах.

Формула изобретения

1. Способ очистки устаревших записей из первого из множества кэш-буферов отображений, причем каждый из упомянутого множества кэш-буферов отображений ассоциирован с соответствующим ему одним из множества устройств обработки данных из состава вычислительного устройства, при этом каждый из кэш-буферов используется для преобразования виртуальных адресов в физические адреса и сохранения отображений, основанных на карте преобразования адресов, причем способ включает в себя этапы, на которых

поддерживают счетчик,

обновляют упомянутый счетчик каждый раз, когда первый из упомянутого множества кэш-буферов отображений очищают,

записывают значение упомянутого счетчика в качестве реакции на изменение в карте преобразования адресов, при этом записанное значение счетчика сохраняют,

определяют на основе результата сравнения упомянутого значения счетчика с записанным значением счетчика, что первый из упомянутого множества кэш-буферов отображений не был доподлинно очищен с тех пор, когда произошло упомянутое изменение карты преобразования адресов, и

очищают первый из упомянутого множества кэш-буферов отображений,

при этом политика определяет дозволенный доступ к памяти, причем способ дополнительно включает в себя этап, на котором

осуществляют управление содержимым карты преобразования адресов так, чтобы карта преобразования адресов не предоставляла объекту отображения виртуальных адресов, которые позволили бы упомянутому объекту осуществить доступ к упомянутой памяти в нарушение упомянутой политики,

при этом упомянутое изменение содержит либо модификацию карты, которая переводит или поддерживает карту в состоянии, согласующемся с упомянутой политикой, либо модификацию карты, которая ограничивает упомянутому объекту доступ на запись к карте.

2. Способ по п.1, в котором карта преобразования адресов содержит ссылку на часть упомянутой памяти, при этом управление содержимым карты преобразования адресов включает в себя этап, на котором делают упомянутую часть упомянутой памяти недоступной для упомянутого объекта, и при этом упомянутое изменение включает в себя удаление всех ссылок на упомянутую часть упомянутой памяти из карты преобразования адресов.

3. Система для управления использованием кэш-буферов отображений адресов,

включающая в себя

множество процессоров, каждый из которых имеет кэш-буфер отображений и счетчик, ассоциированный с ним,

память, хранящую карту преобразований адресов, причем каждый из кэш-буферов отображений хранит отображения, основывающиеся на упомянутой выше карте преобразования адресов, при этом имеется политика, которая регламентирует доступ к упомянутой памяти, причем осуществляется управление содержимым карты преобразования адресов для предотвращения предоставления отображений, которые могут обеспечить доступ к упомянутой памяти в нарушение упомянутой политики,

первое логическое средство, которое очищает первый из кэш-буферов отображений и которое увеличивает первый из счетчиков, когда первый из кэш-буферов отображений очищается,

второе логическое средство, которое записывает значение упомянутого первого счетчика в качестве реакции на изменение в упомянутой карте преобразования адресов или в свойстве, относящемся к упомянутой карте преобразования адресов, причем записанное значение счетчика сохраняется в соответствии с упомянутым изменением,

третье логическое средство, которое сравнивает записанное значение счетчика с текущим значением первого счетчика и которое вызывает очистку первого из кэш-буферов отображений, если результат сравнения показывает, что первый из кэш-буферов отображений не очищался с момента упомянутого изменения.

4. Система по п.3, в которой каждый из процессоров ассоциирован с первым из счетчиков, при этом первое логическое средство увеличивает первый из счетчиков, когда любой из кэш-буферов отображений очищается.

5. Система о п.3, в которой каждый из кэш-буферов отображений ассоциирован с отличным от остальных одним из упомянутого множества счетчиков, при этом первое логическое средство увеличивает счетчик, соответствующий заданному кэш-буферу отображений, когда этот кэш-буфер отображений очищается.

6. Система по п.3, в которой упомянутое изменение содержит перевод упомянутой карты преобразования адресов в состояние, в котором все ссылки на первую страницу упомянутой памяти удалены из упомянутой карты преобразования адресов.

7. Система по п.3, в которой упомянутое изменение содержит перевод упомянутой карты преобразования адресов в состояние, при котором отсутствуют ссылки, позволяющие осуществлять запись на первую страницу упомянутой памяти.

8. Система по п.3, в которой третье логическое средство вызывается в качестве реакции на обнаружение того, что результат упомянутого изменения должен быть использован.

9. Система по п.8, в которой упомянутое изменение содержит перевод упомянутой карты преобразования адресов в состояние, при котором все ссылки на первую страницу упомянутой памяти удалены из упомянутой карты преобразования адресов, при этом упомянутое обнаружение основывается на виртуальном адресе, преобразованном в местоположение на упомянутой первой странице.

10. Машиночитаемый носитель информации, имеющий закодированные на нем машиноисполняемые команды, выполняющие способ управления очисткой буфера быстрого преобразования адресов, который кэширует отображения адресов, при этом способ включает в себя этапы, на которых

принимают запрос на доступ, указывающий целевое местоположение с помощью виртуального адреса,

преобразуют упомянутый виртуальный адрес для получения физического адреса целевого местоположения,

сравнивают (1) сохраненное значение счетчика, связанное со страницей, которая содержит упомянутый целевой адрес, с (2) текущим значением счетчика,

определяют на основе результата сравнения сохраненного значения счетчика и упомянутого текущего значения счетчика, что буфер быстрого преобразования адресов не очищался с момента возникновения события, повлиявшего на то, что отображение для упомянутой страницы изменилось, и

очищают буфер быстрого преобразования адресов,

при этом политика определяет возможность доступа к памяти для программного объекта, причем упомянутое выше событие включает в себя удаление ссылок на страницу упомянутой памяти из карты преобразования адресов, на которой основываются упомянутые отображения адресов, причем упомянутая страница становится недоступной для упомянутого программного объекта в соответствии с упомянутой политикой.

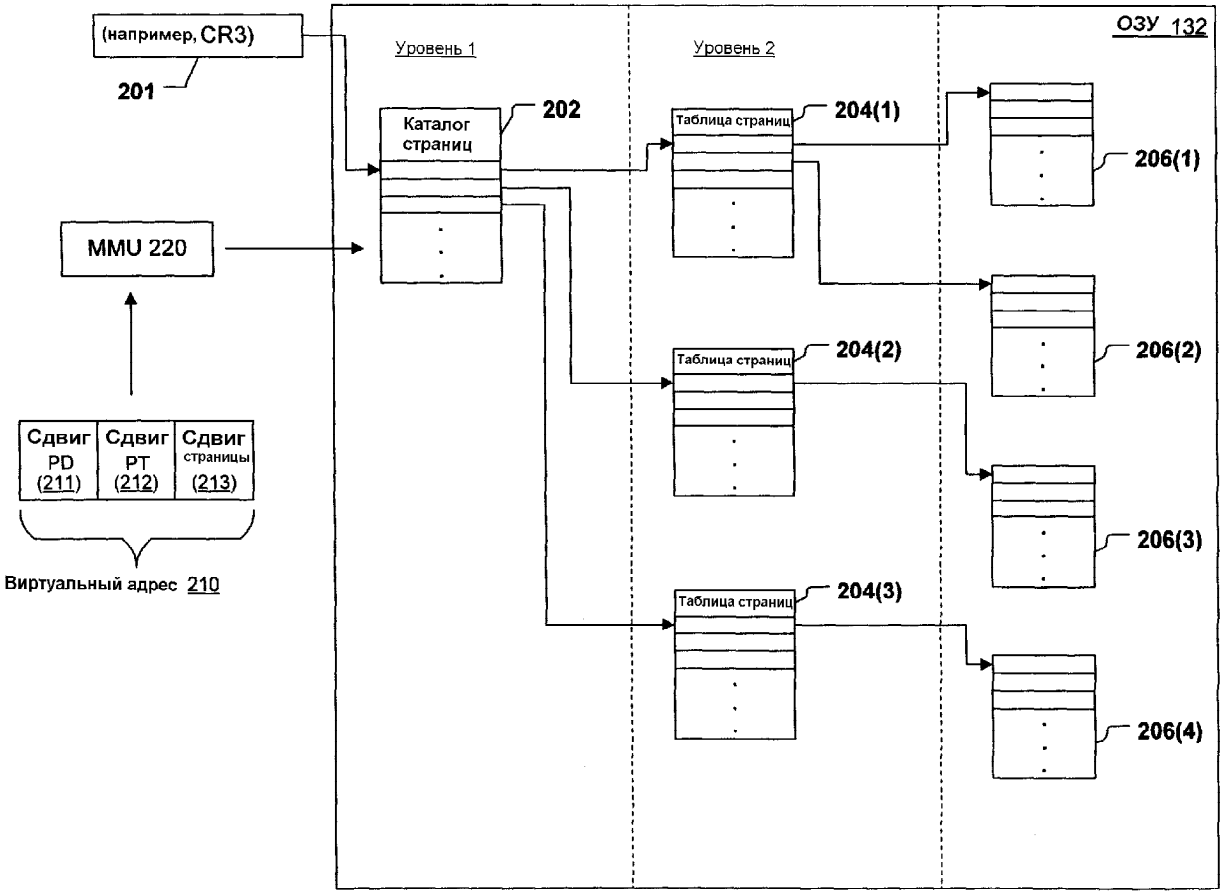
11. Машиночитаемый носитель информации, имеющий закодированные на нем машиноисполняемые команды, выполняющие способ управления очисткой буфера быстрого преобразования адресов, который кэширует отображения адресов, при этом способ включает в себя этапы, на которых

принимают запрос на доступ, указывающий целевое местоположение с помощью виртуального адреса,

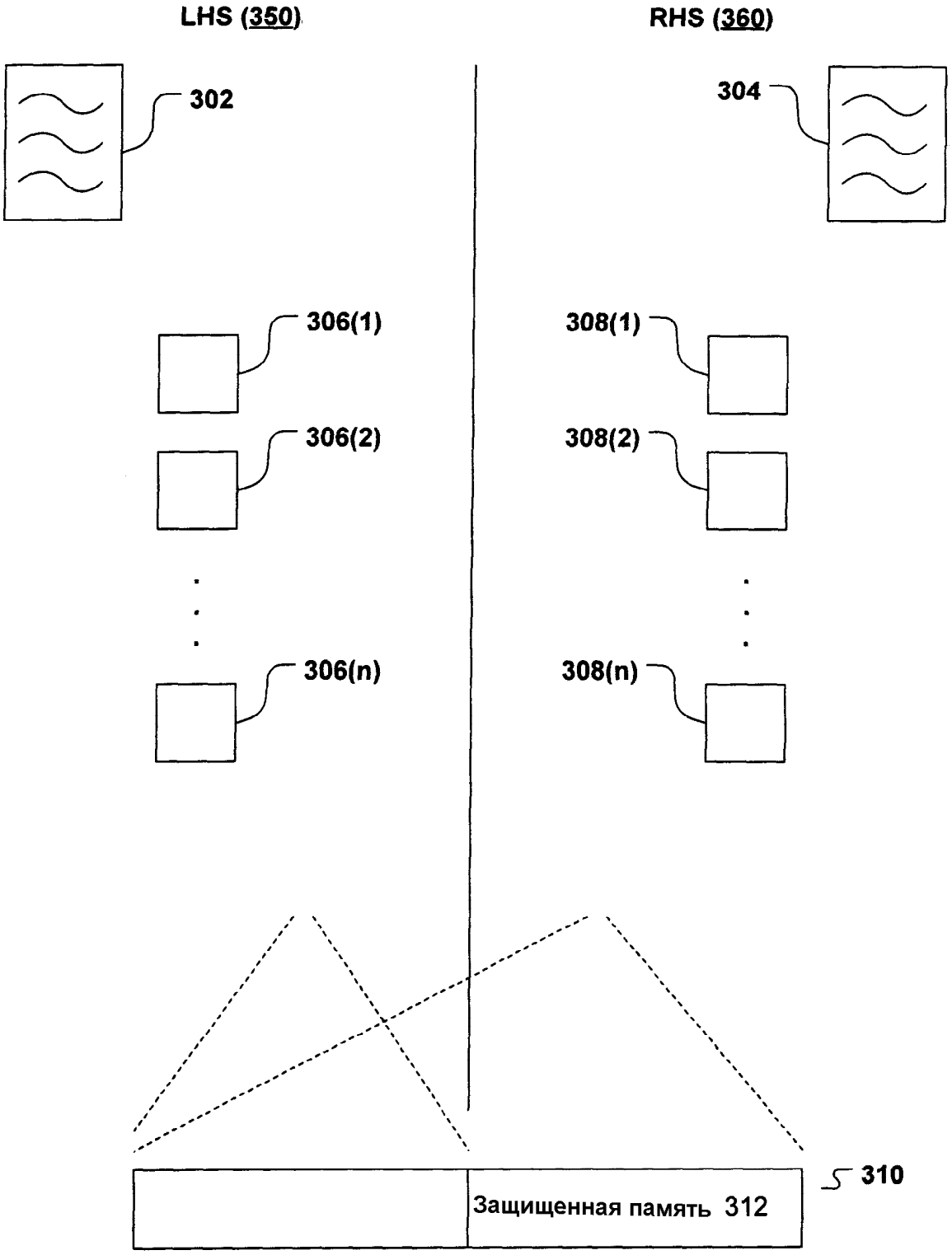
преобразуют упомянутый виртуальный адрес для получения физического адреса целевого местоположения,

сравнивают (1) сохраненное значение счетчика, связанное со страницей, которая содержит упомянутый целевой адрес, с (2) текущим значением счетчика,

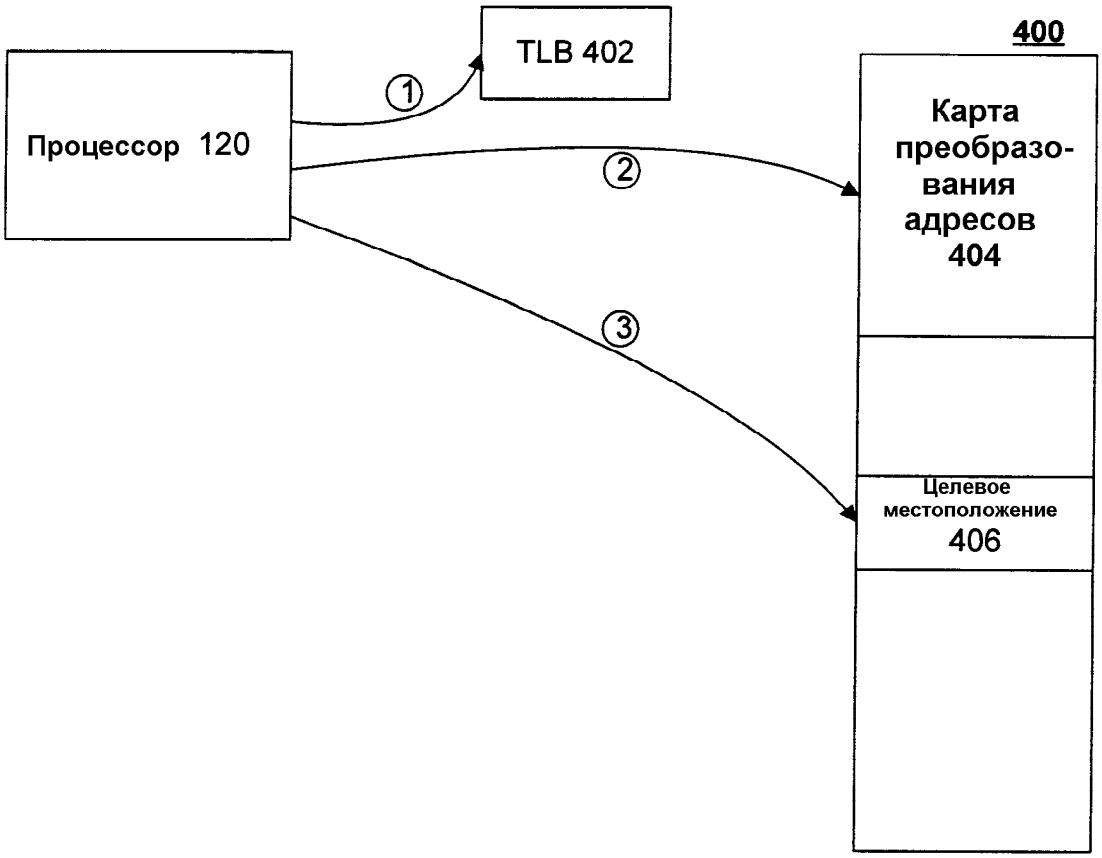
определяют на основе результата сравнения сохраненного значения счетчика и упомянутого текущего значения счетчика, что буфер быстрого преобразования адресов не очищался с момента возникновения события, повлиявшего на то, что отображение для упомянутой страницы изменилось, и очищают буфер быстрого преобразования адресов, при этом политика определяет возможность доступа к памяти для программного объекта, причем упомянутое выше событие включает в себя корректировку карты преобразования адресов для того, чтобы сделать отображения на страницу недоступными для записи, при этом либо (1) упомянутая политика определяет упомянутую страницу как недоступную для записи для упомянутого программного объекта, либо (2) упомянутая страница сохраняет часть упомянутой карты преобразования адресов.


(например, CR3)
201
MMU 220
Сдвиг PD (211)
Сдвиг PT (212)
Сдвиг страницы (213)
Виртуальный адрес 210
Уровень 1
Уровень 2
ОЗУ 132
Каталог страниц
202
Таблица страниц
204(1)
204(2)
204(3)
206(1)
206(2)
206(3)
206(4)

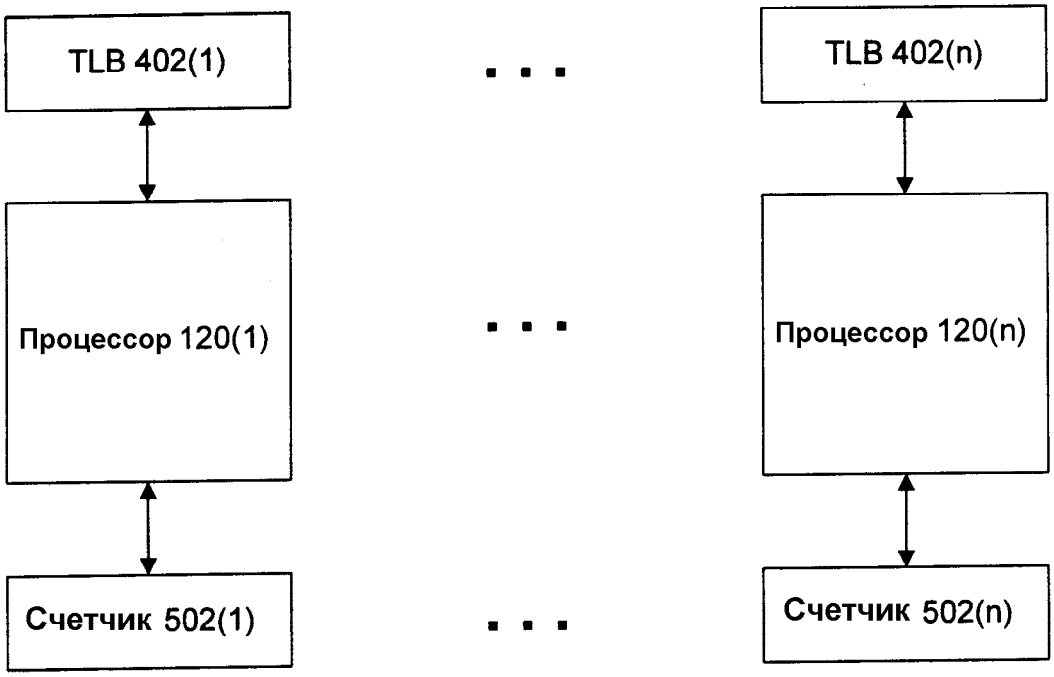
Фиг.2


LHS (350)
RHS (360)
302
304
306(1)
306(2)
306(n)
308(1)
308(2)
308(n)
310
Защищенная память 312

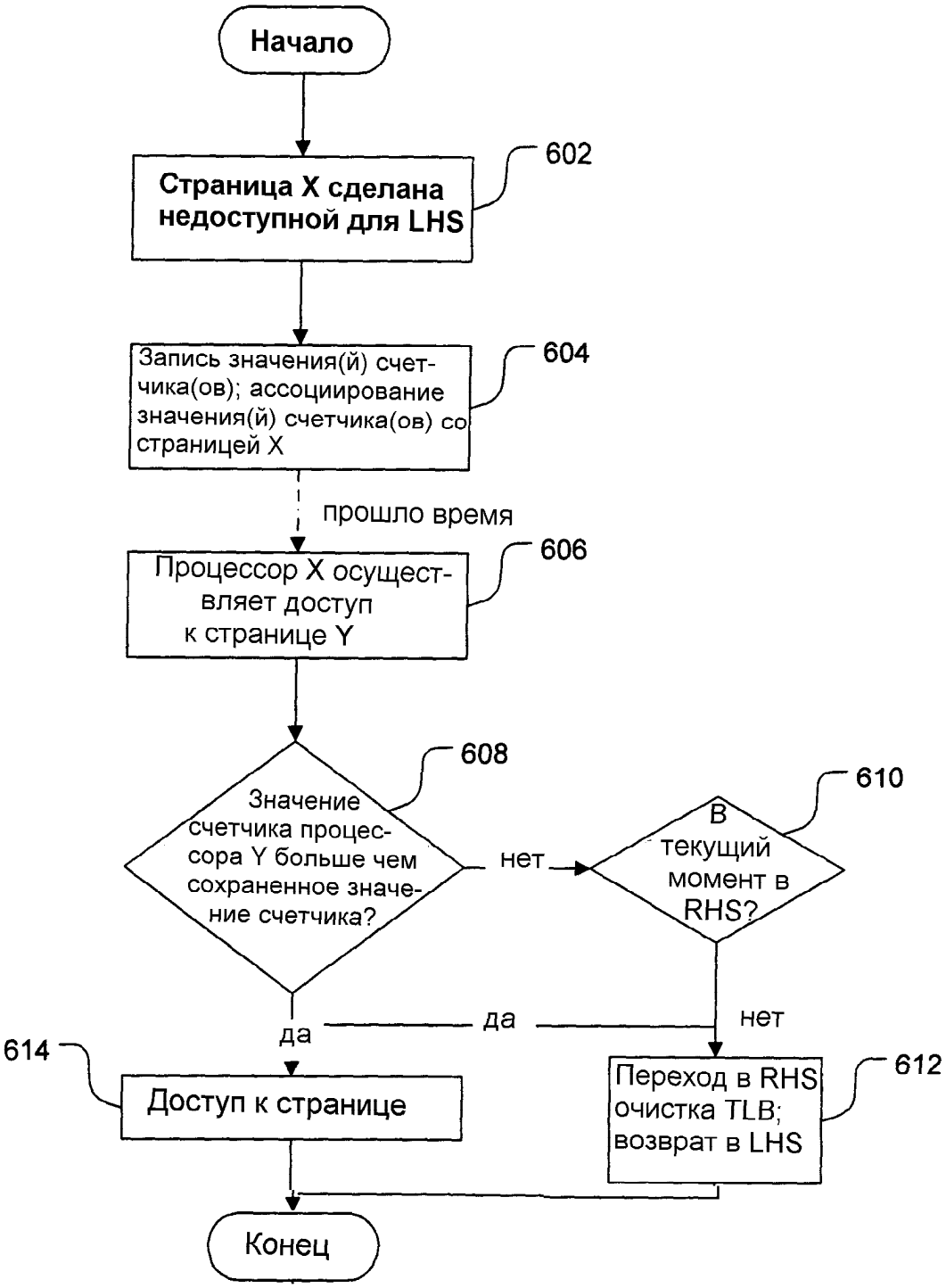
Фиг.3


400
TLB 402
Процессор 120
1
2
3
Карта преобразо-вания адресов 404
Целевое местоположение 406

Фиг.4


TLB 402(1)
TLB 402(n)
Процессор 120(1)
Процессор 120(n)
Счетчик 502(1)
Счетчик 502(n)

Фиг.5


Начало
Страница X сделана недоступной для LHS
602
Запись значения(й) счетчика(ов); ассоциирование значения(й) счетчика(ов) со страницей X
604
прошло время
Процессор X осуществляет доступ к странице Y
606
Значение счетчика процессора Y больше чем сохраненное значение счетчика?
608
нет
В текущий момент в RHS?
610
да
да
нет
614
Доступ к странице
Переход в RHS очистка TLB; возврат в LHS
612
Конец

Фиг.6