Операционные системы

Управление виртуальной памятью в Win32.

Общие принципы.

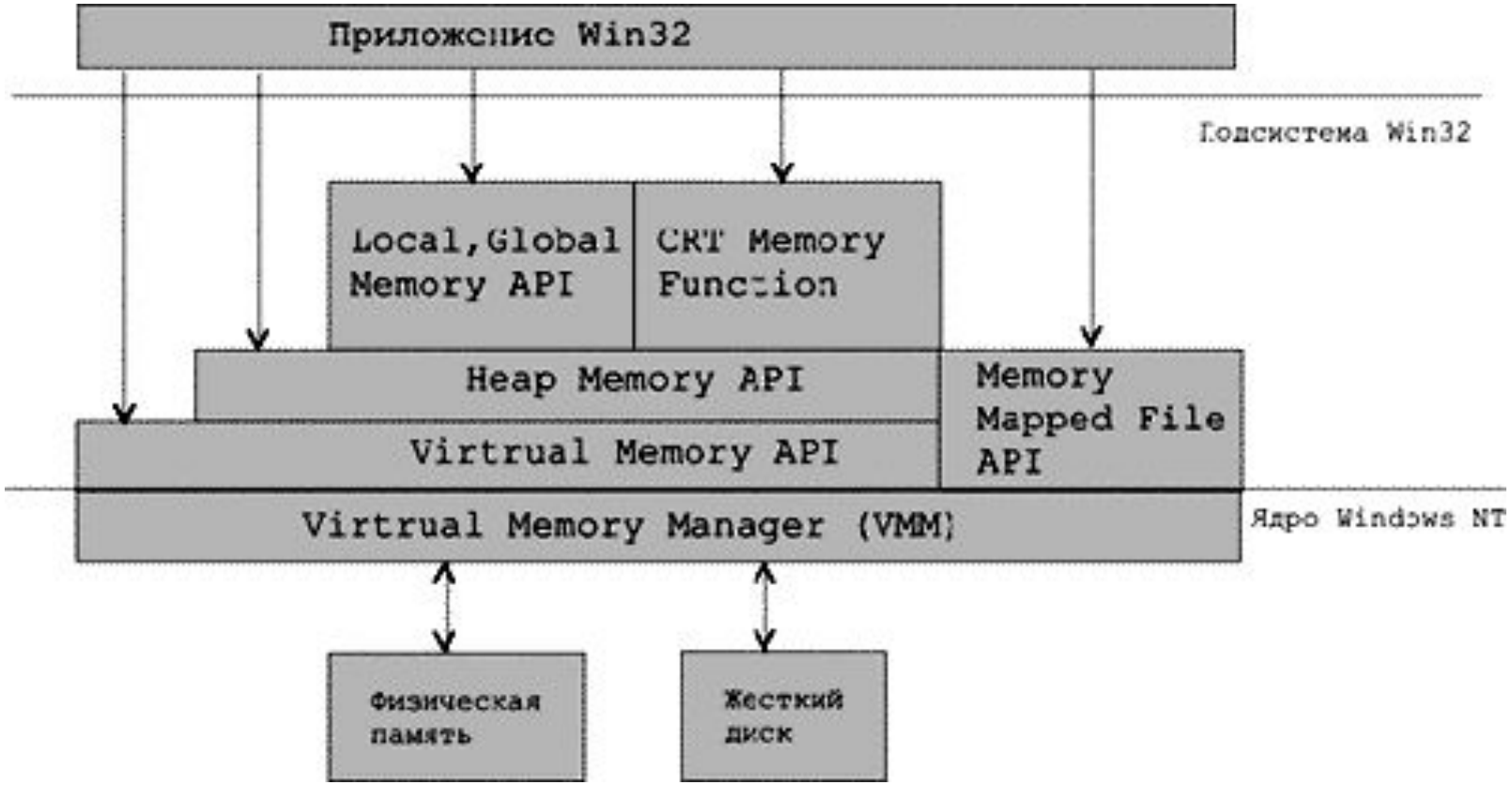
Менеджер виртуальной памяти

- управление виртуальными адресными пространствами процессов;
- разделение памяти между процессами;
- защита виртуальной памяти одного процесса от других процессов.

Менеджер виртуальной памяти

- Менеджер виртуальной памяти (VMM) является составной частью ядра ОС. Приложения не могут получить к нему прямой доступ.
- Для управления памятью прикладным программам предоставляются различные интерфейсы (API).
  - Virtual Memory API – набор функций, позволяющих приложению работать с виртуальным адресным пространством: назначать физические страницы блоку адресов и освобождать их, устанавливать атрибуты защиты.
  - Memory Mapped File API – набор функций, позволяющий работать с файлами, отображаемыми в память. Новый механизм, предоставляемый Win32 API для работы с файлами и взаимодействия процессов.
  - Heap Memory API – набор функций, позволяющих работать с динамически распределяемыми областями памяти (кучами).
  - Local, Global Memory API – набор функций работы с памятью, совместимых с 16-битной Windows. Следует избегать их использования.
  - CRT Memory API – функции стандартной библиотеки языка “C” периода исполнения (runtime).

Архитектура API управления памятью



Адресное пространство процесса

- Чтобы включить в Windows Server 2003 и Windows 2000 расширенное пользовательское пространство, необходимо указать в файле Boot.ini ключ */3GB*.
- Программа для адресации 3 Гб должна быть собрана с ключом */LARGEADDRESSAWARE:YES*.

Средства защиты памяти

- **Объектно-ориентированная защита памяти.** Каждый раз, когда процесс открывает указатель на блок адресов, монитор ссылок безопасности проверяет, разрешен ли доступ процесса к данному объекту.
- **Отдельное адресное пространство для каждого процесса.** Аппаратура запрещает процессу доступ к физическим адресам другого процесса.
- **Два режима работы**: режим ядра, в котором процессам разрешен доступ к системным данным, и пользовательский режим, в котором это запрещен.
- **Страничный механизм защиты.** Каждая виртуальная страница имеет набор признаков, который определяет разрешенные типы доступа в пользовательском режиме и в режиме ядра.
- **Принудительная очистка страниц**, освобождаемых процессами.

Расширение физических адресов (PAE)

- Расширение физических адресов (**Physical Address Extension** – **PAE**) – это функция процессоров с архитектурой IA32, которая делает возможной адресацию физической памяти объемом свыше 4 ГБ.
- PAE – режим работы встроенного блока управления памятью x86-совместимых процессоров, в котором используются 64-битные элементы таблиц страниц (из которых для адресации используются только 36 бит), с помощью которых процессор может адресовать 64 ГБ физической памяти, хотя каждый процесс всё равно может адресовать максимум до 4 ГБ ВП.

Поддержка PAE в различных операционных системах

- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows Server 2003 Enterprise Edition
- Microsoft Windows Server 2003 Datacenter Edition
- Linux начиная с версии 2.3.23
- FreeBSD поддерживает PAE: в линейке 4.x версий — начиная с 4.9, в линейке 5.x версий – начиная с 5.1, все 6.x и более поздние.
- Solaris поддерживает PAE, начиная с версии 7
- В Mac OS X режим PAE включён по умолчанию при использовании 32-разрядного ядра.

Реализация PAE в Windows

Operating system	Maximum memory support with PAE
Windows 2000 Advanced Server	8 GB of physical RAM
Windows 2000 Datacenter Server	32 GB of physical RAM
Windows Server 2003, Enterprise Edition	32 GB of physical RAM
Windows Server 2003, Datacenter Edition	64 GB of physical RAM
Windows Server 2003 SP1, Enterprise Edition	64 GB of physical RAM
Windows Server 2003 SP1, Datacenter Edition	128 GB of physical RAM

Чтобы включить PAE, необходимо указать в файле Boot.ini ключ **/PAE**.

Операционные системы

Управление виртуальной памятью в Win32.

Страничное преобразование.

Страничное преобразование

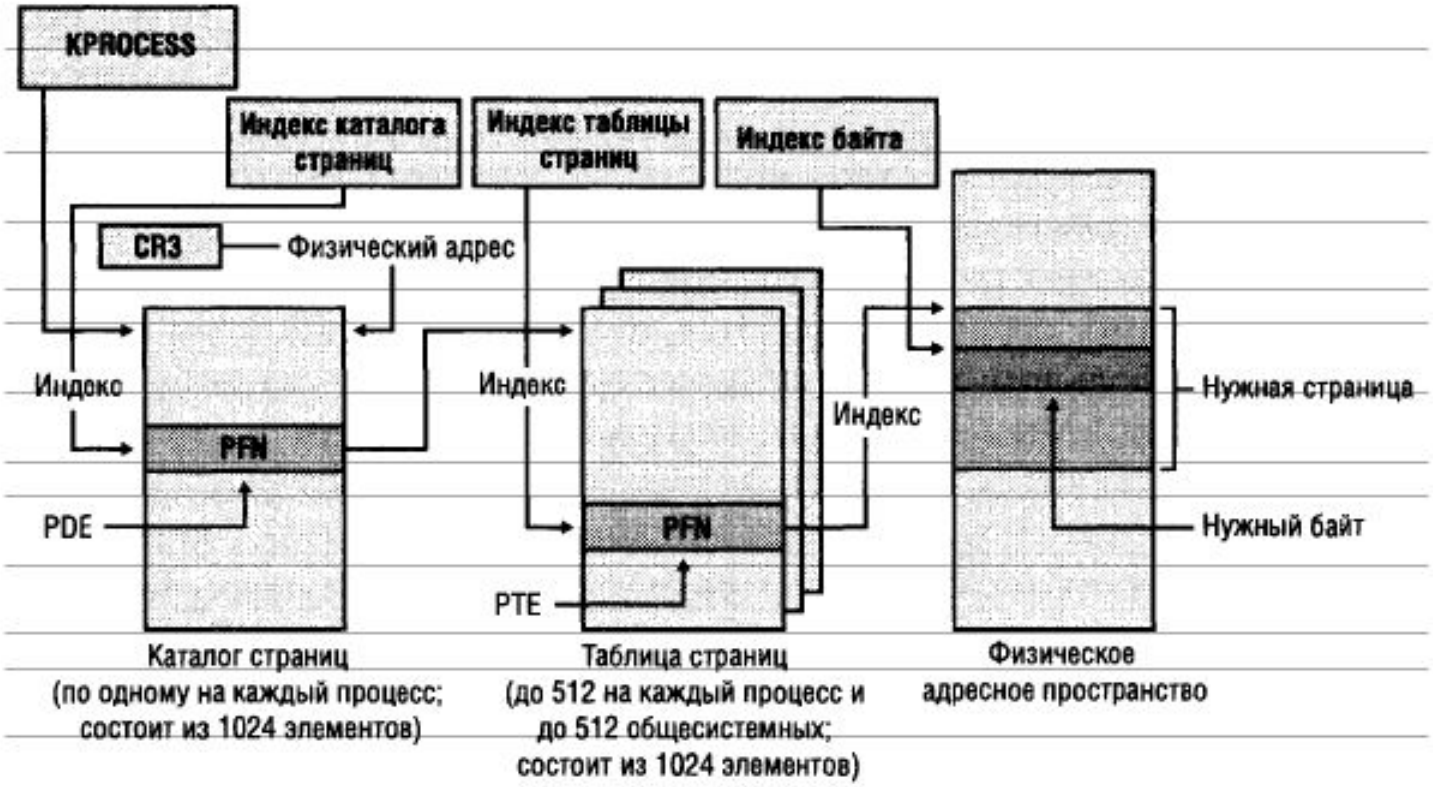
- Виртуальная память в Windows имеет страничную организацию, принятую во многих современных ОС. Процессоры Intel начиная с Pentium Pro позволяют ОС применять одно-, двух- и трехступенчатые схемы. И даже разрешается одновременное использование страниц различного размера.
- Каждому процессу назначается свой каталог страниц. Именно поэтому адресное пространство каждого процесса изолировано, что очень хорошо с точки зрения защиты процессов друг от друга.
- В прошлом Windows поддерживала только двухступенчатую схему преобразования с фиксированным размером страниц.
- Последние версии Windows поддерживают страницы двух размеров, а также реализацию трехступенчатой схемы страничного преобразования для реализации механизма PAE.

Страничное преобразование

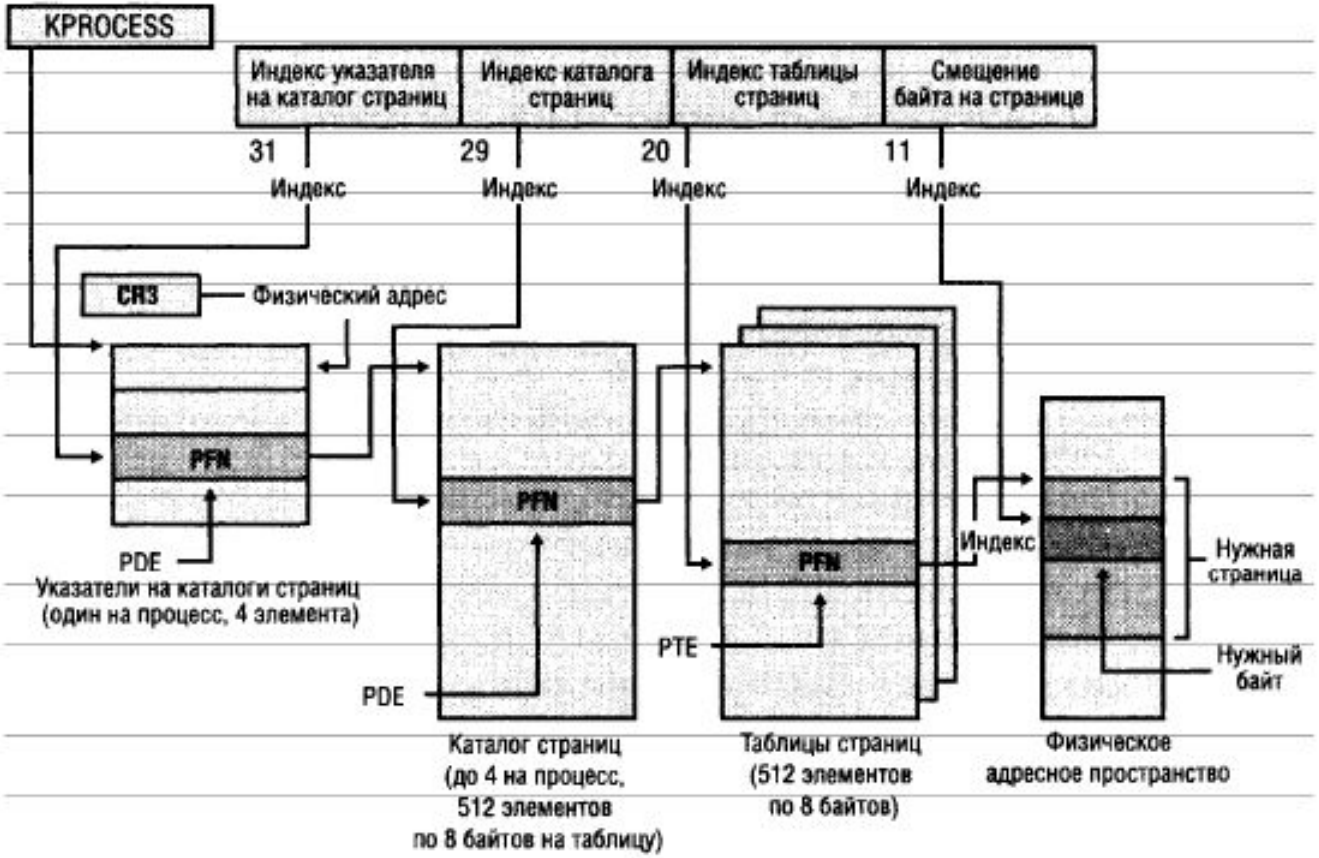
- 32-разрядный виртуальный адрес в ОС Windows разбивается на три части:

  - Старшие 10 разрядов адреса определяют номер одного из 1024 элементов в каталоге страниц, адрес которого находится в регистре процессора CR3. Этот элемент содержит физический адрес таблицы страниц.
  - Следующие 10 разрядов линейного адреса определяют номер элемента таблицы. Элемент, в свою очередь, содержит физический адрес страницы виртуальной памяти.
  - Размер страницы – 4 Кбайт, и младших 12 разрядов линейного адреса как раз хватает ($2^{12} = 4096$), чтобы определить точный физический номер адресуемой ячейки памяти внутри этой страницы.

Трансляция виртуального адреса в x86 системах



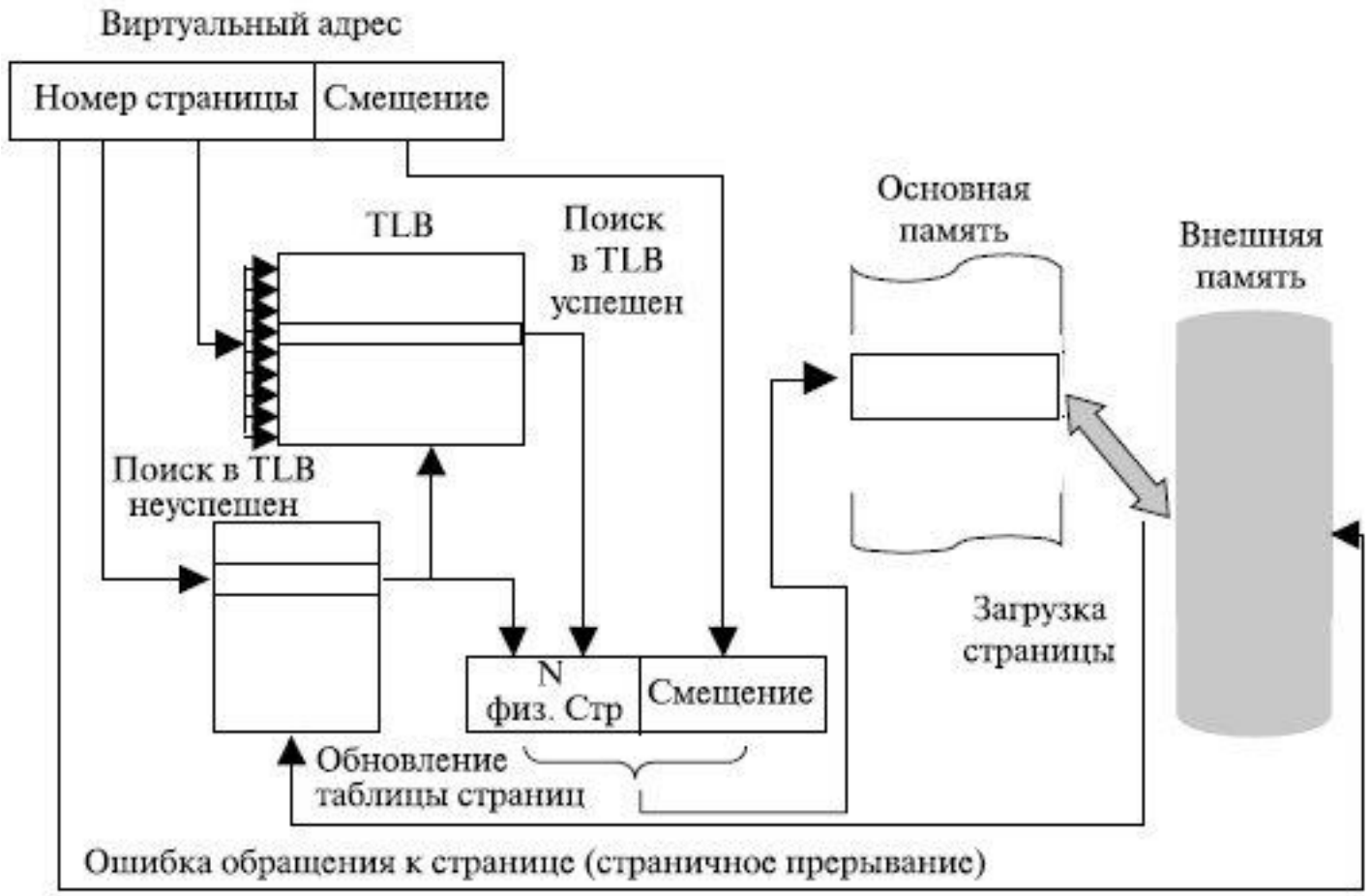
Проецирование страниц по механизму PAE



Ассоциативный буфер трансляции

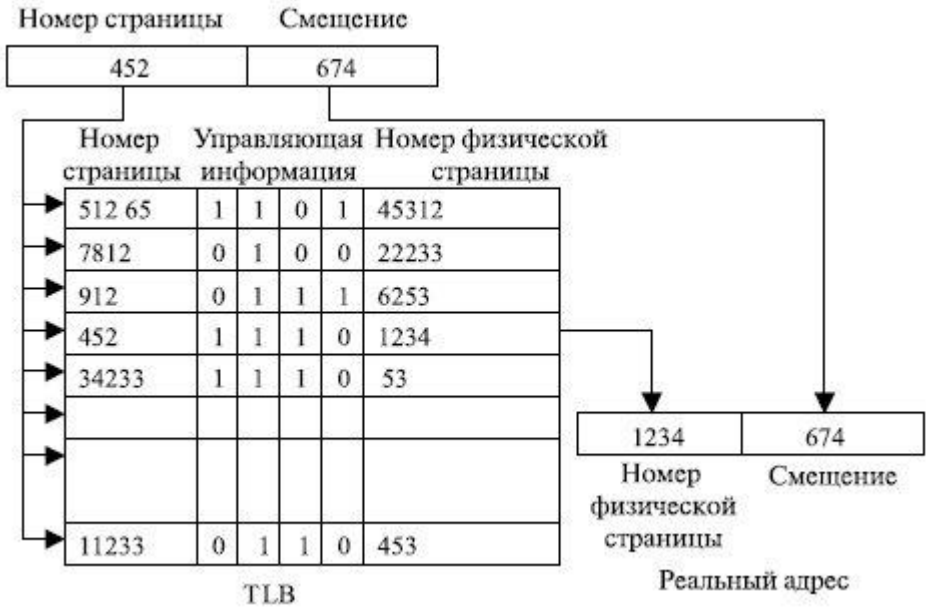
- Чтобы ускорить этот процесс современные процессоры используют ассоциативный буфер трансляции (translation look-aside buffer, TLB), для кэширования наиболее частых связей между физической и виртуальной памятью.
- TLB кэш имеет большое значение, так как страниц памяти в стандартных 32-разрядных Linux, Unix или Windows серверах очень много.
- Кэш TLB представляет собой вектор, ячейки которого можно считывать и сразу сравнивать с целевым значением. В случае TLB вектор содержит сопоставления физических и виртуальных адресов для недавно использовавшихся страниц, а также атрибуты защиты каждой страницы
- Если запрашиваемый адрес не находится в кэше, то процессор, перед ответом на запрос, переходит к таблице страниц для поиска соответствия виртуального адреса физическому.
- TLB должен также предусматривать способы организации записей в кэш и принятия решения о том, какая из старых записей должна быть удалена при внесении в кэш новой записи.

Реализация TLB



Пример использования TLB

- Процессор аппаратно способен одновременно опрашивать все записи TLB для определения того, какая из них соответствует заданному номеру страницы.
- Такой подход известен как ассоциативное отображение (associative mapping).



Поддержка больших страниц

- Начиная с версии Windows Server 2008 включена поддержка сверхбольших страниц (обычно 2Мбайт). Реальный размер зависит от аппаратной платформы.
- Недостатком маленьких страниц является неэффективное использование TLB (Translation Lookside Buffer), в случае использования больших страниц TLB используется более эффективно, что приводит к значительному увеличению производительности.

Архитектура	Малая страница	Большая страница
X86	4 КБ	4 МБ (2 МБ в PAE-режиме)
x64	4 КБ	2 МБ
IA64	8 КБ	16 МБ

Эффективность использования больших страниц

- Для страниц размером 4 КБ, механизм TLB содержит 32 записи в L1 кэше и 512 записей в L2 кэше. Так как каждая запись ссылается на 4 КБ, вы можете посчитать что вместе они покрывают чуть более 2 МБ виртуальной памяти.
- Для больших страниц TLB содержит восемь записей. Так как каждая страница отображает 2 МБ, TLB может показывать 16 МБ виртуальной памяти.

Элемент таблицы страниц

Элемент таблицы страниц

- Защита – Win32 API поддерживает ряд значений, в том числе: *PAGE_NOACCESS, PAGE_READONLY, PAGE_READWRITE, PAGE_EXECUTE*.
- Базовый физический адрес страницы в памяти.
- Pagefile – индекс используемого файла подкачки (один из 16 возможных в системе файлов).
- State – состояние страницы в системе:
  - *T (Transition)* – отмечает страницу как переходную;
  - *D (Dirty)* – страница, в которую была произведена запись;
  - *P (Present)* – страница присутствует в ОП или находится в файле подкачки.

Отдельные состояния страниц

- Valid – страница используется процессом. Она реально существует в ОП и помечена в PTE как присутствующая в рабочем множестве процесса (P=1, D=0,1).
- Modified – содержимое страницы было изменено (D=1). В PTE страница помечена как отсутствующая (P=0) и переходная (T=1).
- Standby – содержимое страницы не изменялось (D=0). В PTE страница помечена как отсутствующая (P=0) и переходная (T=1).
- Free – страница, на которую не ссылается ни один PTE. Страница свободна, но подлежит обнулению, прежде чем будет использована.
- Zeroed – свободная и обнуленная страница, пригодная к непосредственному использованию любым процессом.
- Bad – страница, которая вызывает аппаратные ошибки и не может быть использована ни одним процессом.

Отдельные состояния страниц

T	D	P	Page state
0	-	0	Invalid page (Free)
-	0	1	Valid page
-	1	1	Valid dirty page
1	0	0	Invalid page in transition (Standby)
1	1	0	Invalid dirty page in transition (Modified)

Операционные системы

Управление виртуальной памятью в Win32.

Свопинг.

Свопинг

- Для того, чтобы обеспечить все линейное адресное пространство процесса физическими ячейками памяти, Windows применяет свопинг.
- Организацией свопинга занимается VMM. При генерации системы на диске образуется специальный файл свопинга, куда записываются те страницы, которым не находится места в физической памяти. Процессы могут захватывать память в своем ВАП и, затем, использовать ее. Страница может иметь различные состояния.
- VMM использует локальный алгоритм **LRU** (Least Recently Used) – замещение дольше всех неиспользовавшихся страниц.

Стратегия управления виртуальной памятью

- *Стратегия выборки (fetch policy)*
- *Стратегия размещения (placement policy)*
- *Стратегия замещения (replacement policy)*

Стратегия выборки

- *Стратегия выборки (fetch policy):*
  - Выборка определяет, в какой момент необходимо переписать страницу с диска в ОП.
  - В Windows используется классическая схема выборки с упреждением: система переписывает в память не только выбранную страницу, но и несколько следующих по принципу пространственной локальности, гласящему: наиболее вероятным является обращение к тем ячейкам памяти, которые находятся в непосредственной близости от ячейки, к которой производится обращение в настоящий момент. Поэтому вероятность того, что будут востребованы последовательные страницы, достаточна высока. Их упреждающая подкачка позволяет снизить накладные расходы, связанные с обработкой прерываний.
- *Стратегия размещения (placement policy)*
- *Стратегия замещения (replacement policy)*

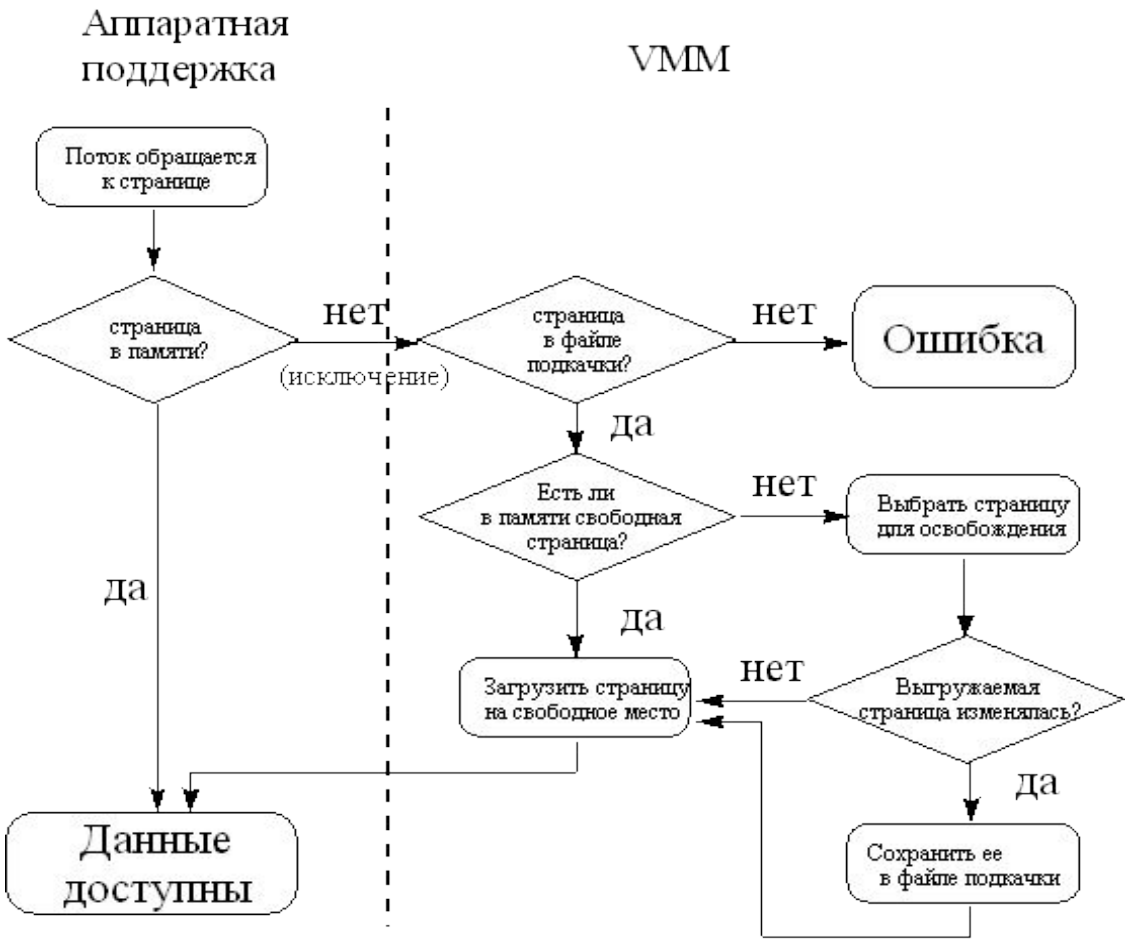
Стратегия размещения

- *Стратегия выборки (fetch policy)*
- *Стратегия размещения (placement policy):*
  - Размещение определяет, в какое место оперативной памяти необходимо поместить подгружаемую страницу.
  - Для систем со страничной организацией данная стратегия практически не имеет никакого значения, и поэтому Windows выбирает первую попавшуюся свободную страницу.
- *Стратегия замещения (replacement policy)*

Стратегия замещения

- *Стратегия выборки (fetch policy)*
- *Стратегия размещения (placement policy)*
- *Стратегия замещения (replacement policy):*
  - Замещение начинает действовать с того момента, когда в оперативной памяти компьютера не остается свободного места для размещения подгружаемой страницы. В этом случае необходимо решить, какую страницу вытеснить из физической памяти в файл подкачки (свопинг).

Реализация свопинга



Реализация алгоритма замещения LRU

- VMM периодически просматривает список страниц с атрибутом Valid и пытается похитить их у процесса **(1)**. Он помечает их как отсутствующие (P=0), но на самом деле оставляет их на месте, только переводит в разряд Modified или Standby в зависимости от значения бита D из PTE.
- Если содержимое страницы была изменено в ОП (D=1), то VMM выполнит запись страницы на диск **(4)**.
- Если похищенная страница принадлежит рабочему множеству, то к ней в ближайшее время произойдет обращение. Это, конечно, вызовет исключение - ведь страница-то помечена как отсутствующая. Но VMM очень быстро сделает эту страницу вновь доступной процессу, поскольку она реально находится в памяти **(2)**.
- Далее если к странице не будет обращений (страница вне рабочего множества), то она со временем перейдет в состояние Free **(5)** и станет доступна для замещения страниц в рамках данного процесса **(6)**.
- Затем системный поток обнуляет страницу – Zeroed **(7)**, и она станет доступна другим процессам системы **(8)**.

Переходы между состояниями страниц