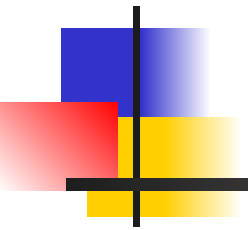
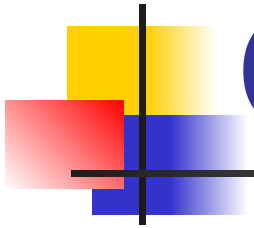


Виртуальная память

Операционные Системы

кафедра ИУС ФТК СпбГПУ

Коликова Т.В.



О чем

- Введение
- Управление памятью
  - Трансляция адресов
  - Ассоциативный буфер трансляции
  - Стратегия подкачки и рабочие наборы
  - База данных страничных фреймов
  - Дескрипторы виртуальных адресов

Виртуальная память

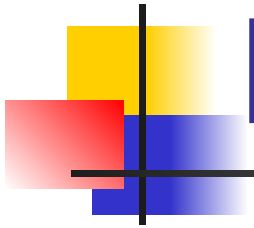
- Нельзя выполнить программу, размер которой > размера физической памяти
- **Виртуальная память** – это централизованная система выгрузки на диск содержимого памяти при переполнении последней
- Можно создавать и запускать “большие” программы

Виртуальная память

- физическая и логическая структура
- Способ, которым ОС транслирует одну структуру в другую
- **Физическая память** - последовательность однобайтовых ячеек
- **Логическая память** или **виртуальная память (virtual memory)** – это способ представления памяти для программы
- В современных ОС она не совпадает с физической структурой памяти
- Системы виртуальной памяти используют:
  - сегментное
  - линейное представление памяти

Виртуальная память

- **Виртуальное адресное пространство (virtual address space**)– это набор адресов памяти, которые могут использовать потоки процесса
- Каждый процесс имеет отдельное адресное пространство > физической памяти
- Диапазон физических адресов компьютера (у каждого байта уникальный адрес)
- Диапазон виртуальных адресов – количество битов в адресе (32-разрядный адрес - в 4 Гб)



Виртуальная память

- Система виртуальной памяти должна:
  - Транслировать, или отображать (map)
    - определяет по виртуальному адресу соответствующий физический адрес
  - Выгружать на диск часть содержимого памяти

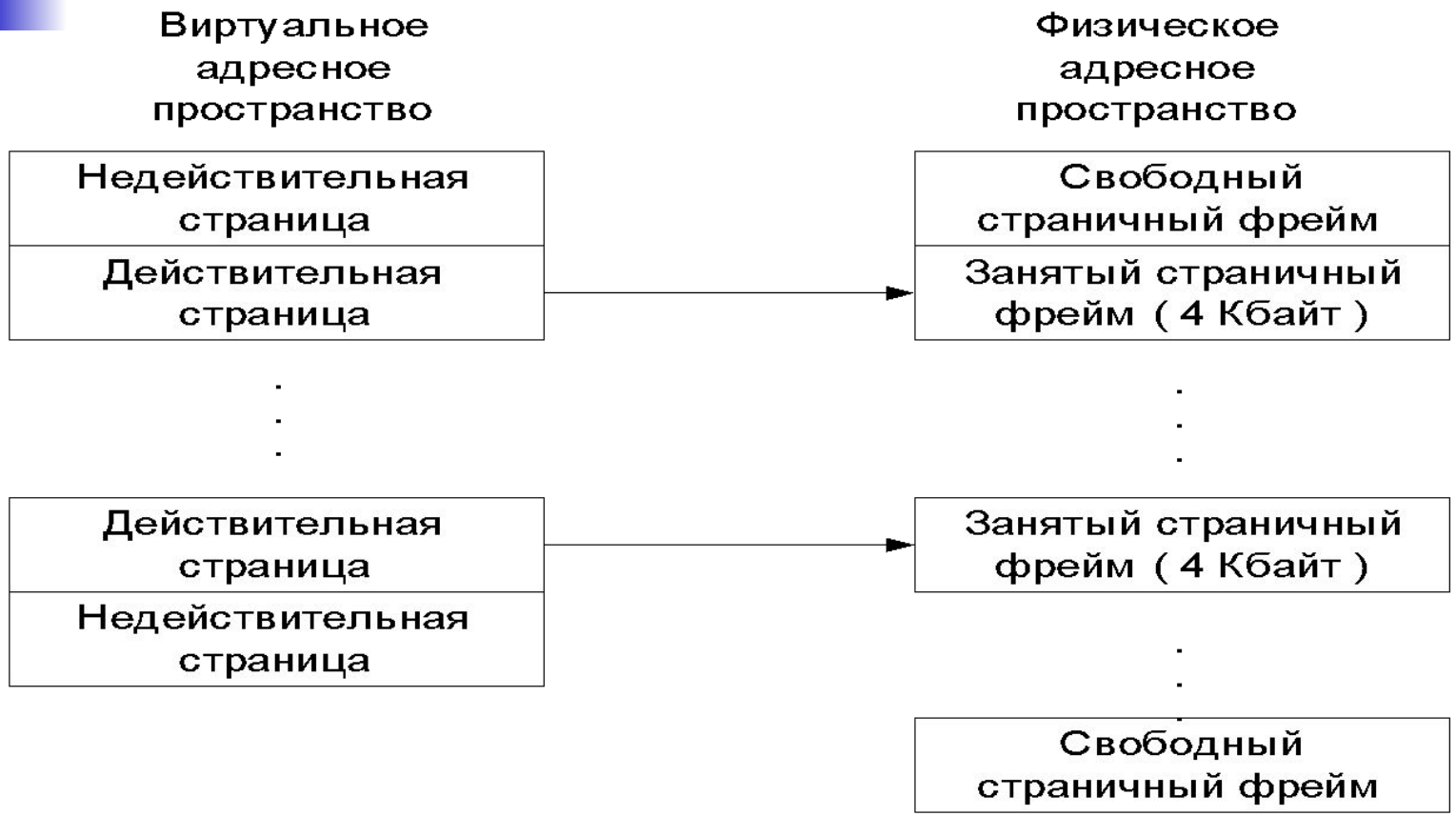
Виртуальная память

- Медленно перемещать по одному байту
- Виртуальное адресное пространство разделено на блоки равного размера - **страницы (pages)**
- Физическое адресное пространство разделяется на блоки - **страничные фреймы (page frames**)

Виртуальная память

- Страницы, находящиеся в физической памяти и доступные немедленно, называются **действительными страницами (valid pages**)
- Страницы, находящиеся на диске (или находящиеся в памяти, но недоступные немедленно), называются **недействительными (invalid pages)**

Виртуальная память



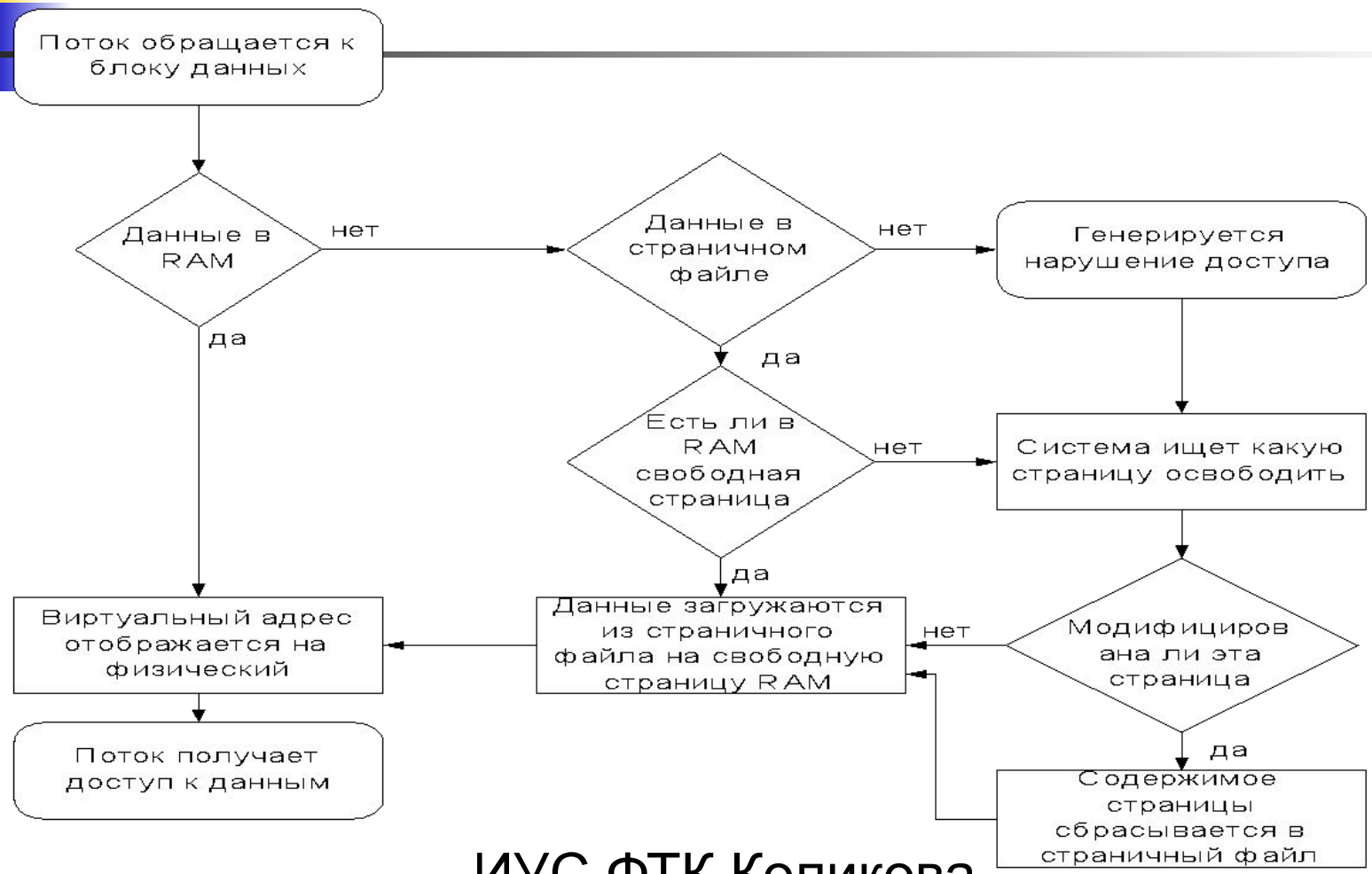
Виртуальная память

- При обращении потока по виртуальному адресу, который находится на недействительной странице процессор генерирует **страничную ошибку (page fault)**
- Система виртуальной памяти находит нужную страницу на диске и загружает ее в свободный страничный фрейм физической памяти
- Когда остается мало доступных страничных фреймов, система виртуальной памяти выбирает фреймы, которые можно освободить и копирует их содержимое на диск
- Этот процесс называется **подкачкой страниц (paging)**

Виртуальная память

- Обработка страничной ошибки – это дорогая операция, которая требует много тактов процессора
- Можно снизить затраты за счет выбора оптимального размера страницы
  - При большем размере страницы мы загружаем в память большее количество данных и, поэтому страничные ошибки будут возникать реже
  - Но при слишком большом размере, мы можем загрузить лишние данные
- Оптимальный размер – 4Кб

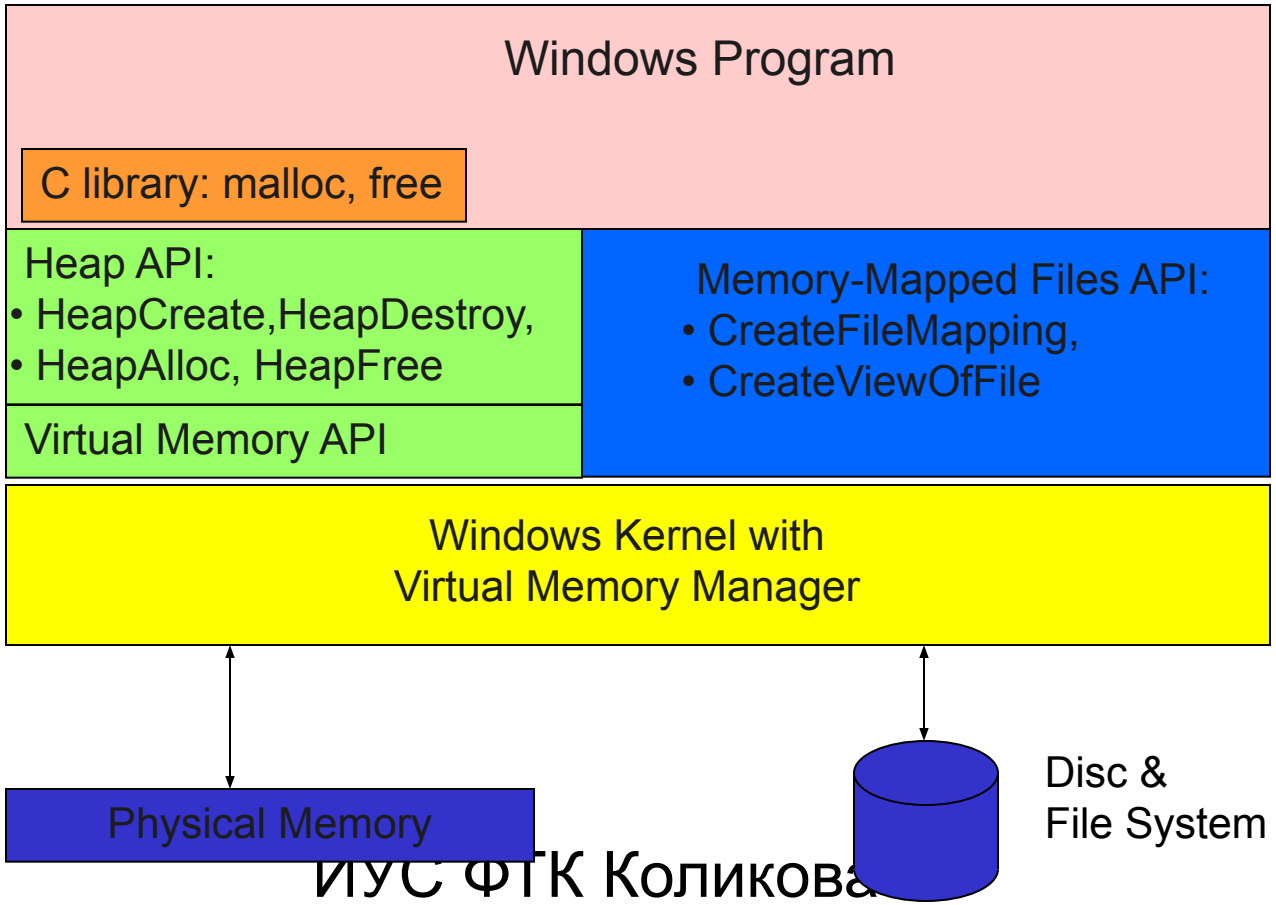
Виртуальная память



Управление памятью

- **Диспетчер виртуальной памяти** предоставляет набор базовых сервисов. Эти сервисы позволяют процессу:
  - выделять память в два этапа
  - выполнять чтение/запись из/в виртуальной памяти
  - фиксировать виртуальные страницы в физической памяти
  - получать информацию о виртуальных страницах
  - защищать виртуальные страницы
  - сбрасывать содержимое виртуальных страниц на диск

Windows API Memory Management Architecture



Windows Memory Management

- Windows maintains pools of memory in heaps
- A process can contain several heaps
  - C library functions manage default heap: malloc, free, calloc
- Heaps are Windows objects – have handle
  - Each process has own default heap
  - Return value of NULL indicates failure (instead of INVALID_HANDLE_VALUE)

```
HANDLE GetProcessHeap( VOID );
HANDLE HeapCreate (DWORD floptions,
      DWORD dwInitialSize,
      DWORD dwMaximumSize);
BOOL HeapDestroy( HANDLE hHeap );
```

Managing Heap Memory

```
LPVOID HeapAlloc( HANDLE hHeap,
      DWORD dwFlags,
      DWORD dwBytes );
```

- dwFlags:
  - HEAP_GENERATE_EXCEPTION,
    - raise SEH on memory allocation failure
    - STATUS_NO_MEMORY, STATUS_ACCESS_VIOLATION
  - HEAP_NO_SERIALIZE:
    - no serialization of concurrent (multithreaded) requests
  - HEAP_ZEROC_MEMORY: initialize allocated memory to zero
- dwSize:
  - Block of memory to allocate
  - For non-growable heaps: 0x7FFF8 (0.5 MB)
- HeapFree(), HeapReAlloc(),
- HeapCompact(), HeapValidate()

HeapLock(), HeapUnlock(): Manage concurrent accesses to heap

Управление памятью

- Диспетчер виртуальной памяти выделяет память в два этапа:
  - **резервирование. Зарезервированная память (reserved memory)** – это набор виртуальных адресов, которые диспетчер виртуальной памяти зарезервировал для использования процессом. Это быстрая и дешевая операция.
  - **передача. Переданной памятью (committed memory)** называется память, для которой диспетчер виртуальной памяти выделил место в **файле подкачки (paging file)** – файле на диске, в который он записывает виртуальные страницы, когда их надо удалить из памяти
- Поток может сразу зарезервировать и передать виртуальную память, либо вначале зарезервировать ее, а передавать по мере необходимости

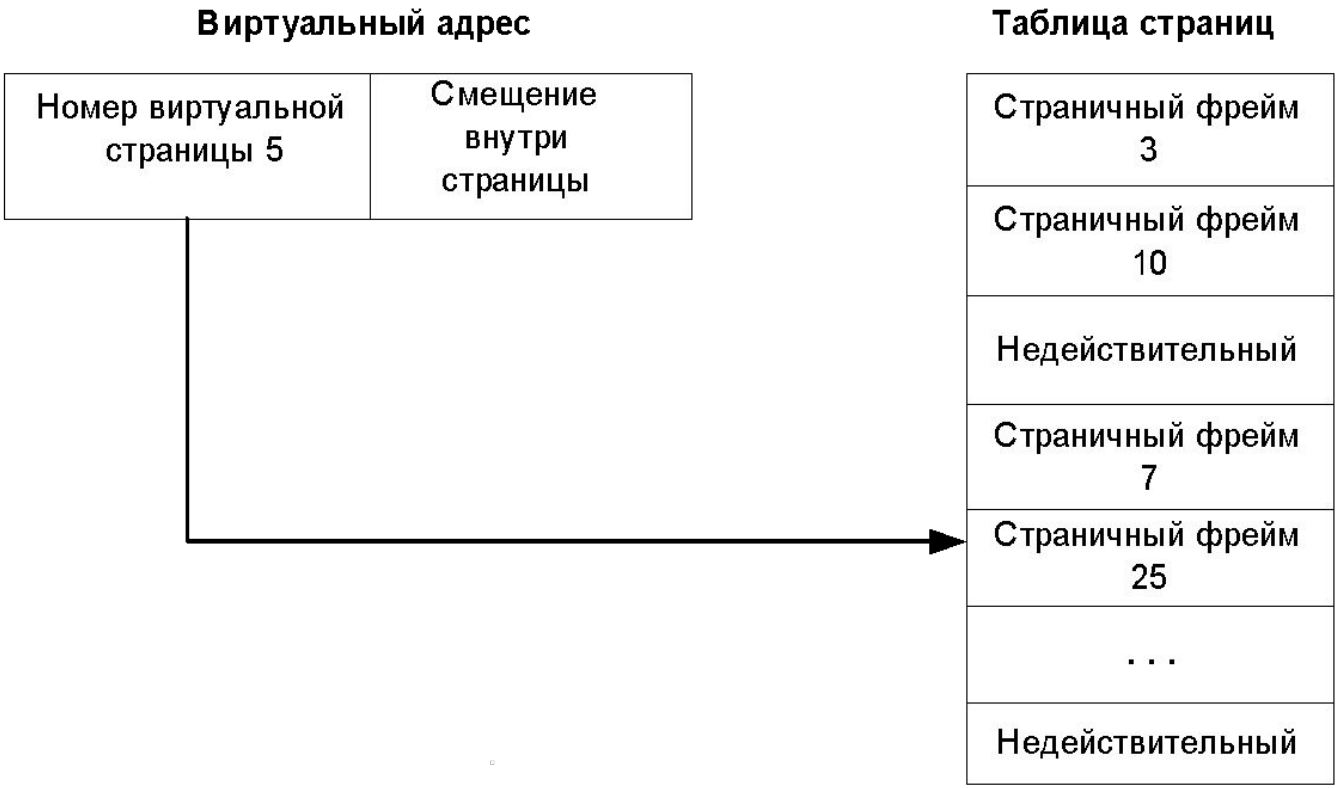
Управление памятью

- Резервирование памяти полезно при создании динамических структур данных
- Поток резервирует последовательные виртуальные адреса, а передает их, когда по ним необходимо поместить данные
- Диспетчер виртуальной памяти списывает с процесса часть его квоты в файле подкачки при передаче, но не при резервировании
- Так поток может зарезервировать большой регион виртуальной памяти, но не расходовать квоту до тех пор, пока эта память действительно не понадобится

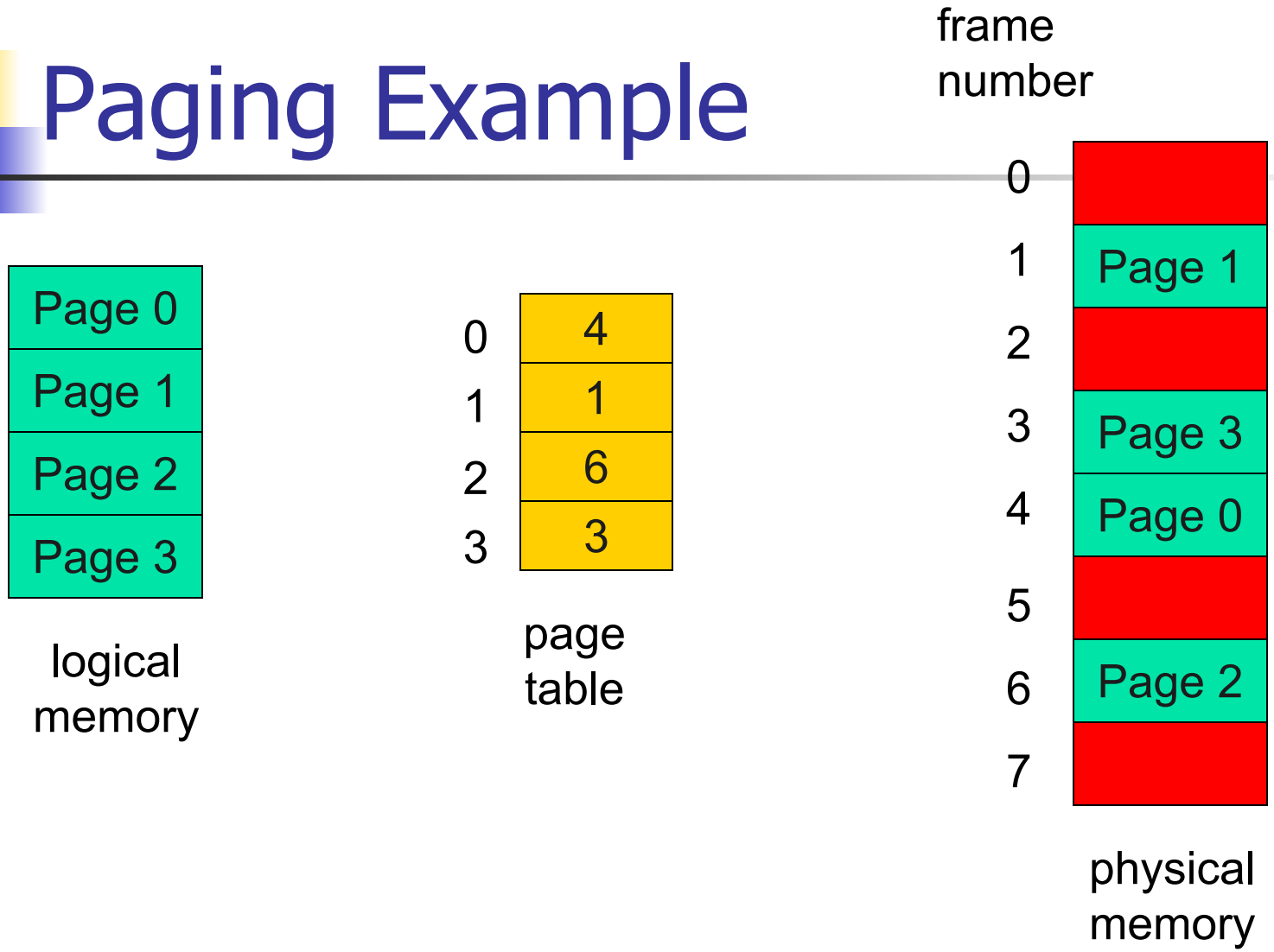
Трансляция адресов

- Приложения используют 32-разрядные виртуальные адреса
- С помощью структур данных создаваемых и поддерживаемых диспетчером памяти, происходит трансляция этих виртуальных адресов в физические
- Каждый виртуальный адрес сопоставляется со структурой системного пространства, которая называется **элементом таблицы страниц (page table entry, PTE)**
- Она и содержит физический адрес, соответствующий виртуальному

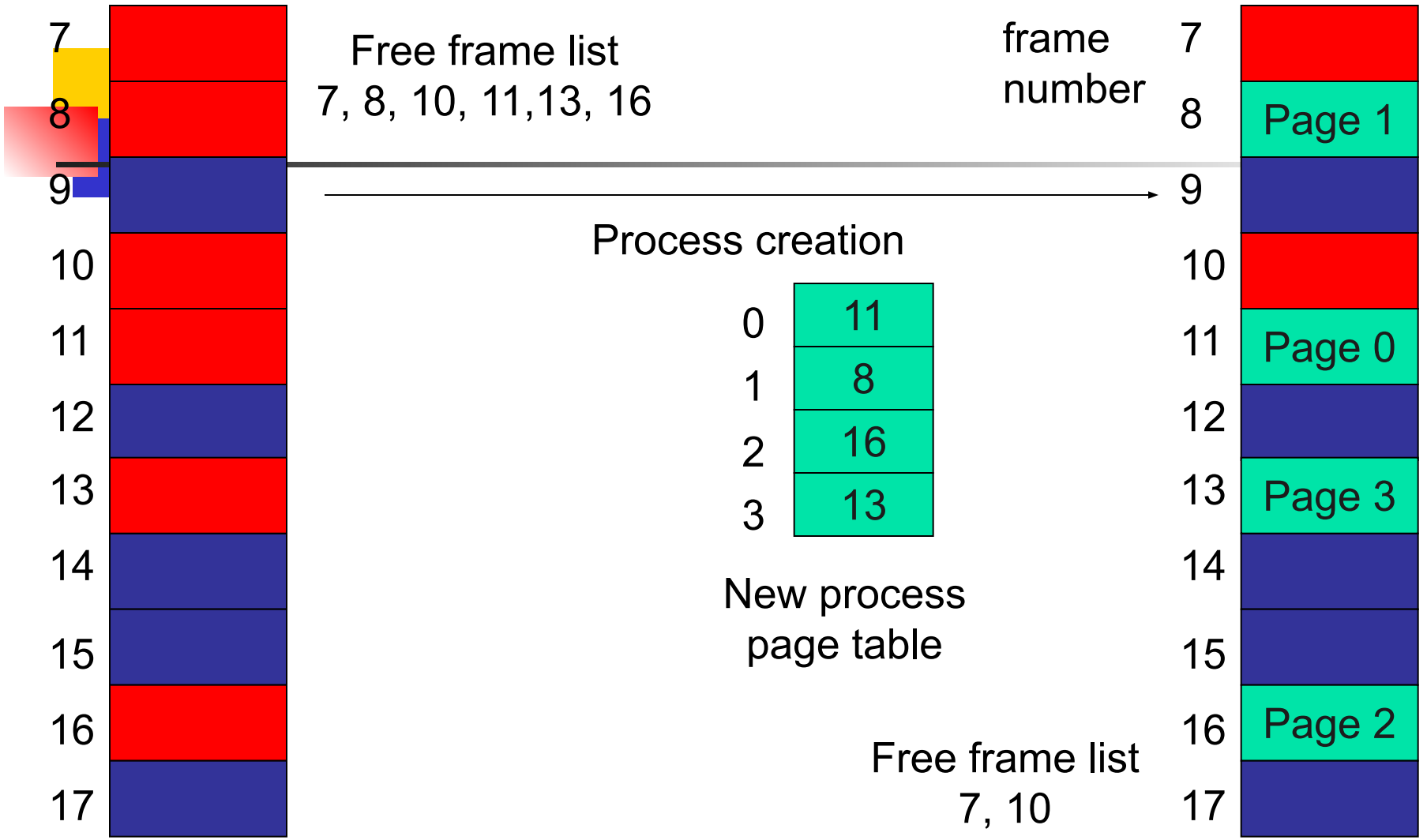
Трансляция адресов



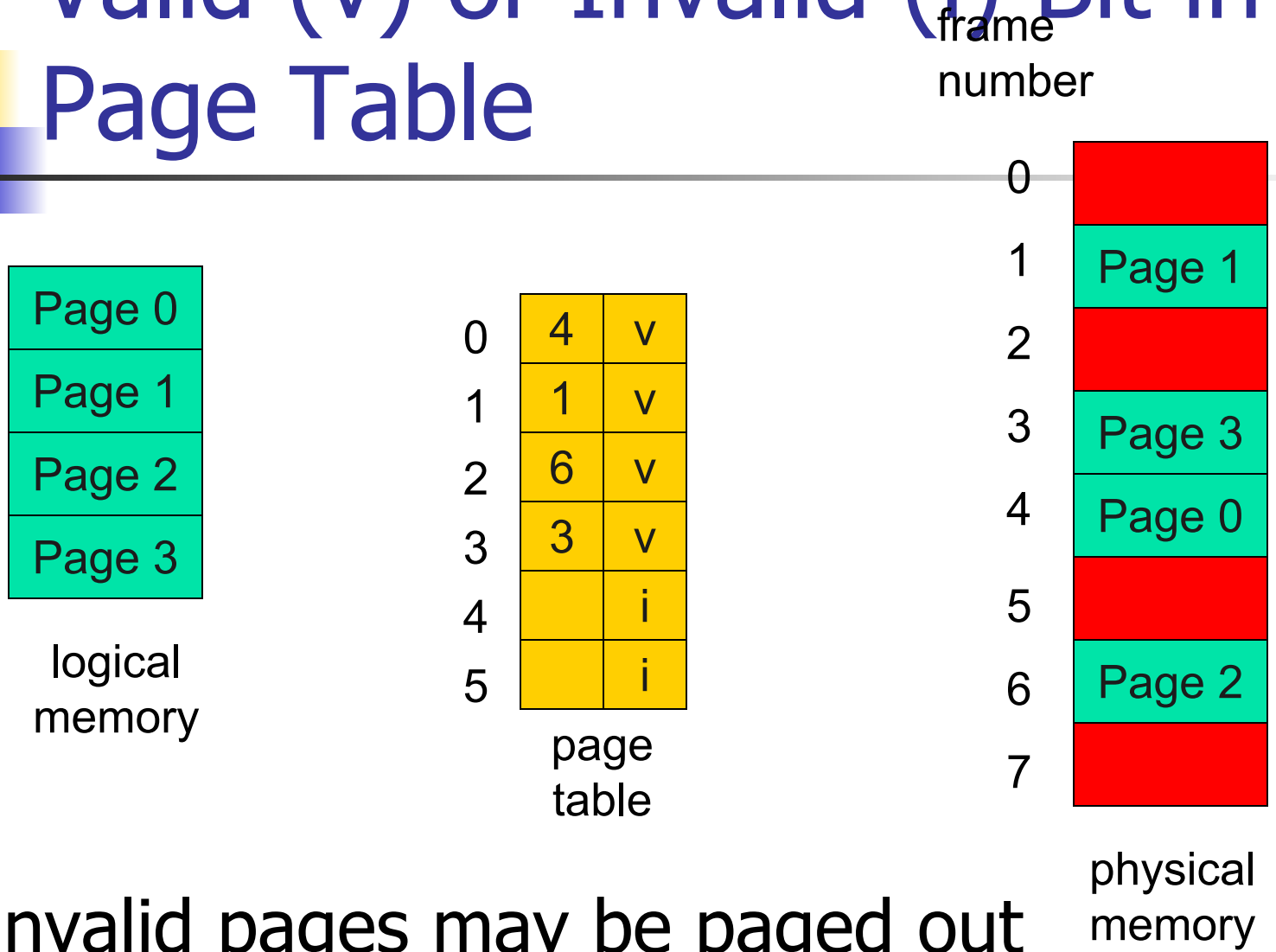
Paging Example



Free Frames


7
8
9
10
11
12
13
14
15
16
17
Free frame list
7, 8, 10, 11,13, 16
frame number
7
8
9
10
11
12
13
14
15
16
17
Page 1
Page 0
Page 3
Page 2
Process creation
0
1
2
3
11
8
16
13
New process page table
Free frame list
7, 10
Before allocation
After allocation
physical memory

Valid (v) or Invalid (i) Bit in a Page Table

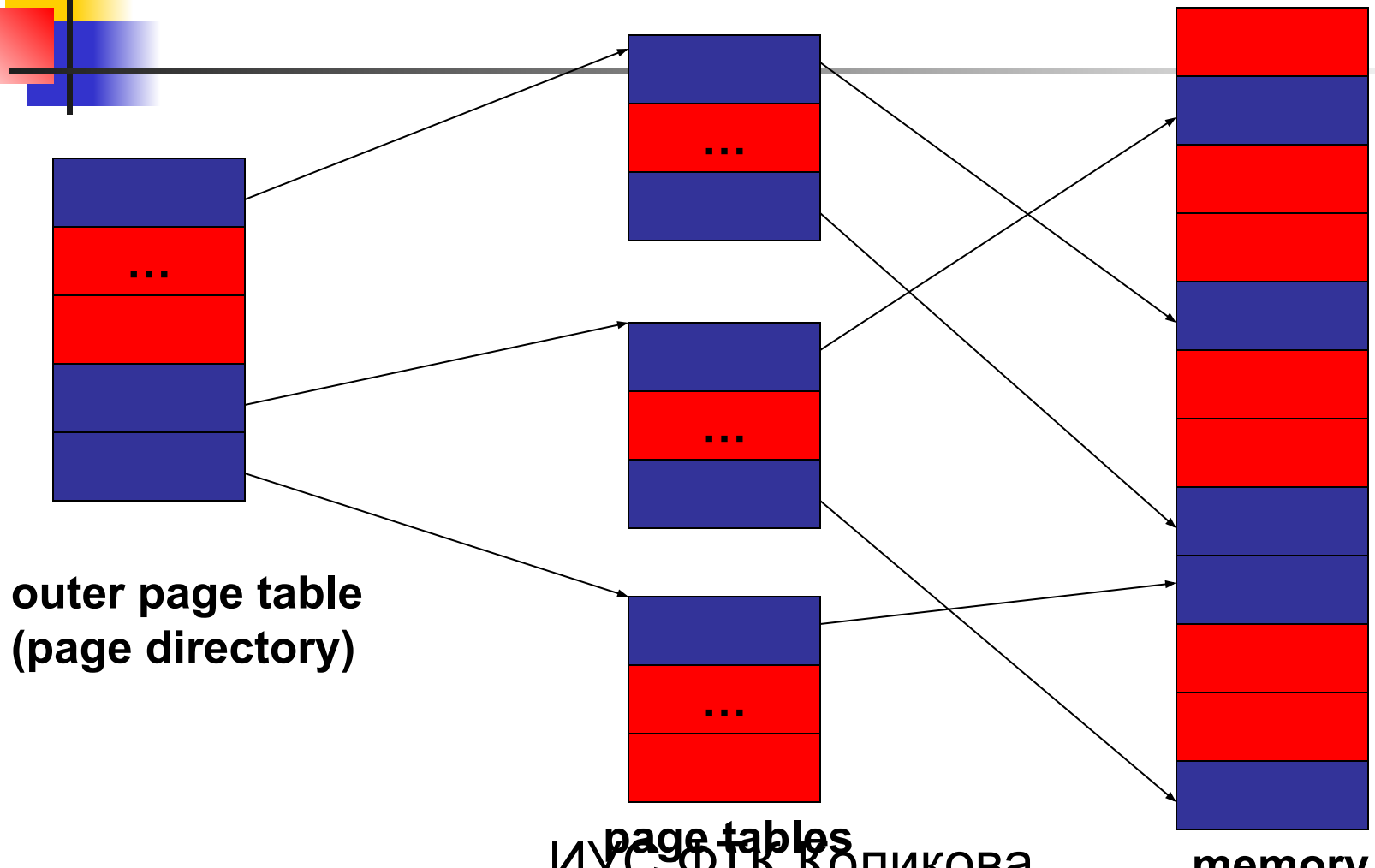


- Invalid pages may be paged out

Трансляция адресов

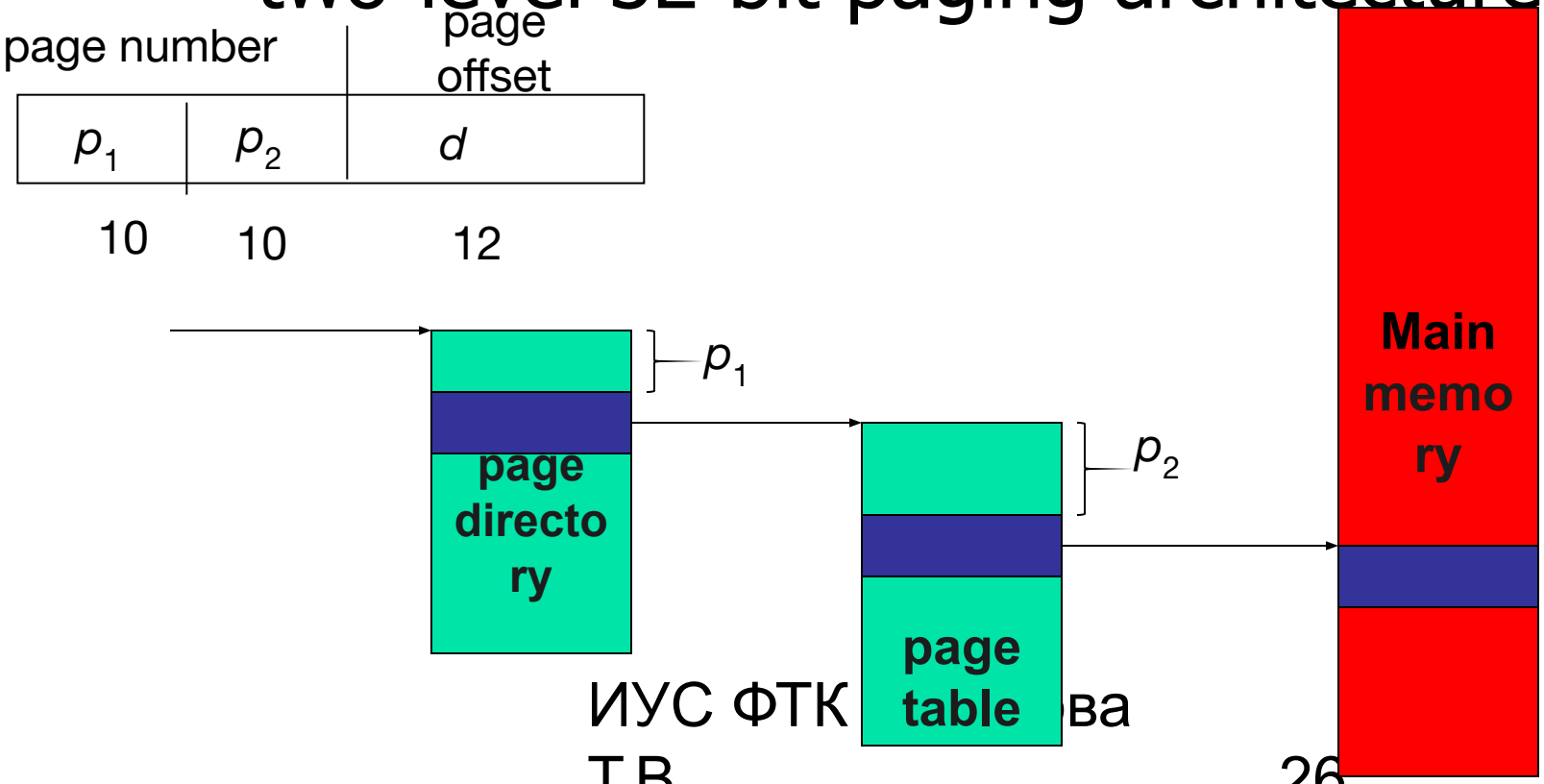
- Для трансляции виртуальных адресов в физические используется двухуровневая таблица страниц
- Виртуальный 32-разрядный адрес состоит из трех элементов:
  - ***индекса каталога страниц***
  - ***индекса таблицы страниц***
  - ***индекса байта***

Two-Level Page-Table Scheme

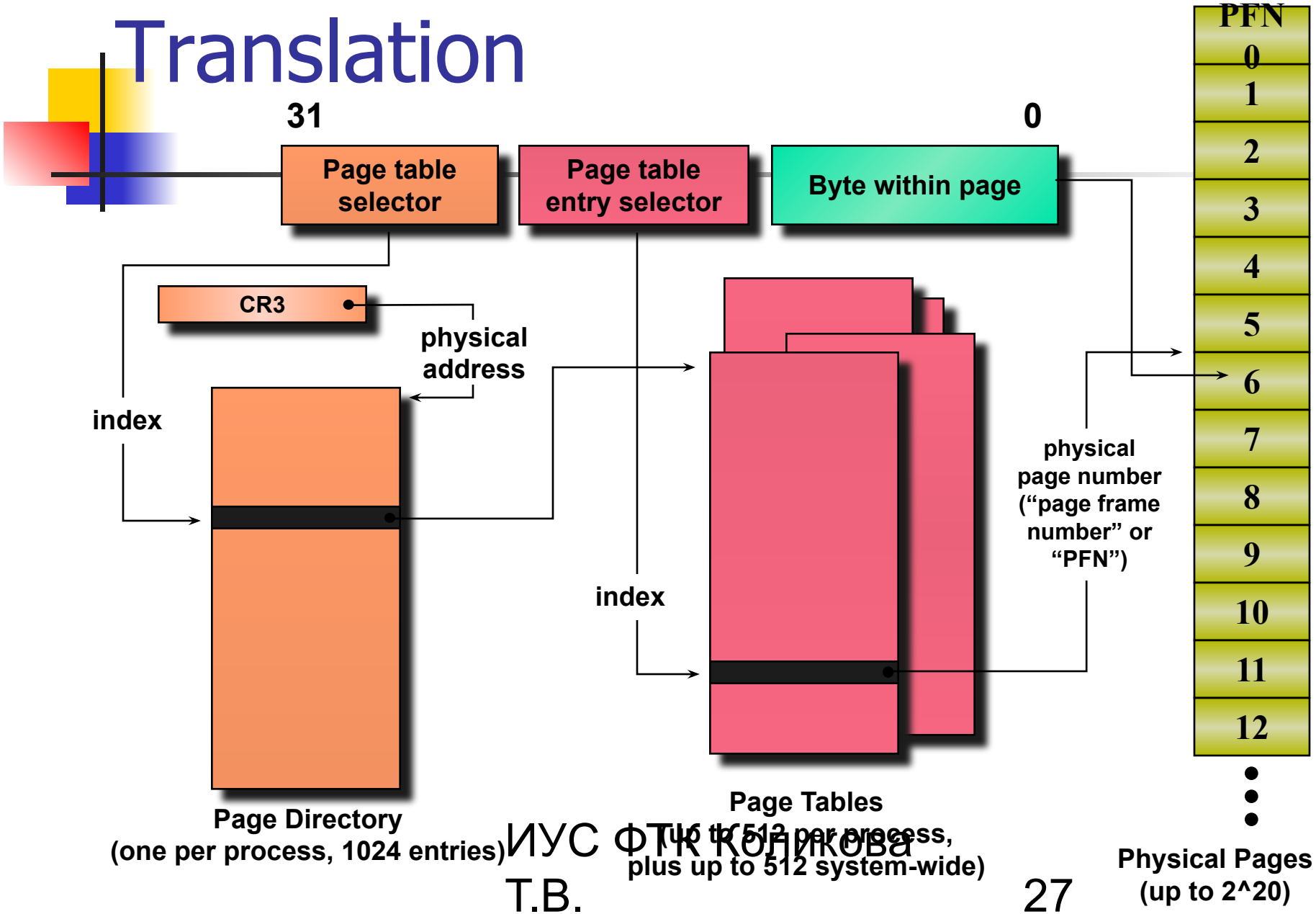


Address-Translation Scheme

- Address-translation scheme for a two-level 32-bit paging architecture



x86 Virtual Address Translation



Трансляция адресов

Трансляция адресов

- **Индекс каталога страниц (page directory index)** применяется для поиска таблицы страниц, содержащий PTE для данного виртуального адреса
- С помощью **индекса таблицы страниц (page table index)** осуществляется поиск PTE, который содержит физический адрес, по которому проецируется виртуальная страница
- **Индекс байта (byte index)** позволяет найти конкретный адрес на физической странице
- У каждого процесса есть один **каталог страниц (page directory**), который представляет собой страницу с адресами всех таблиц страниц для данного процесса

Трансляция адресов

- При трансляции виртуального адреса:
  - Определяется адрес каталога страниц текущего процесса. При переключении контекста процесса ОС заносит этот адрес в специальный регистр процессора
  - Индекс каталога страниц используется для поиска элемента каталога страниц (page directory entry, PDE). Он указывает на ту таблицу страниц, которая нужна для трансляции виртуального адреса. PDE содержит номер фрейма страницы (page frame number, PFN) таблицы страниц (если она находится в памяти, таблица может быть выгружена в страничный файл)

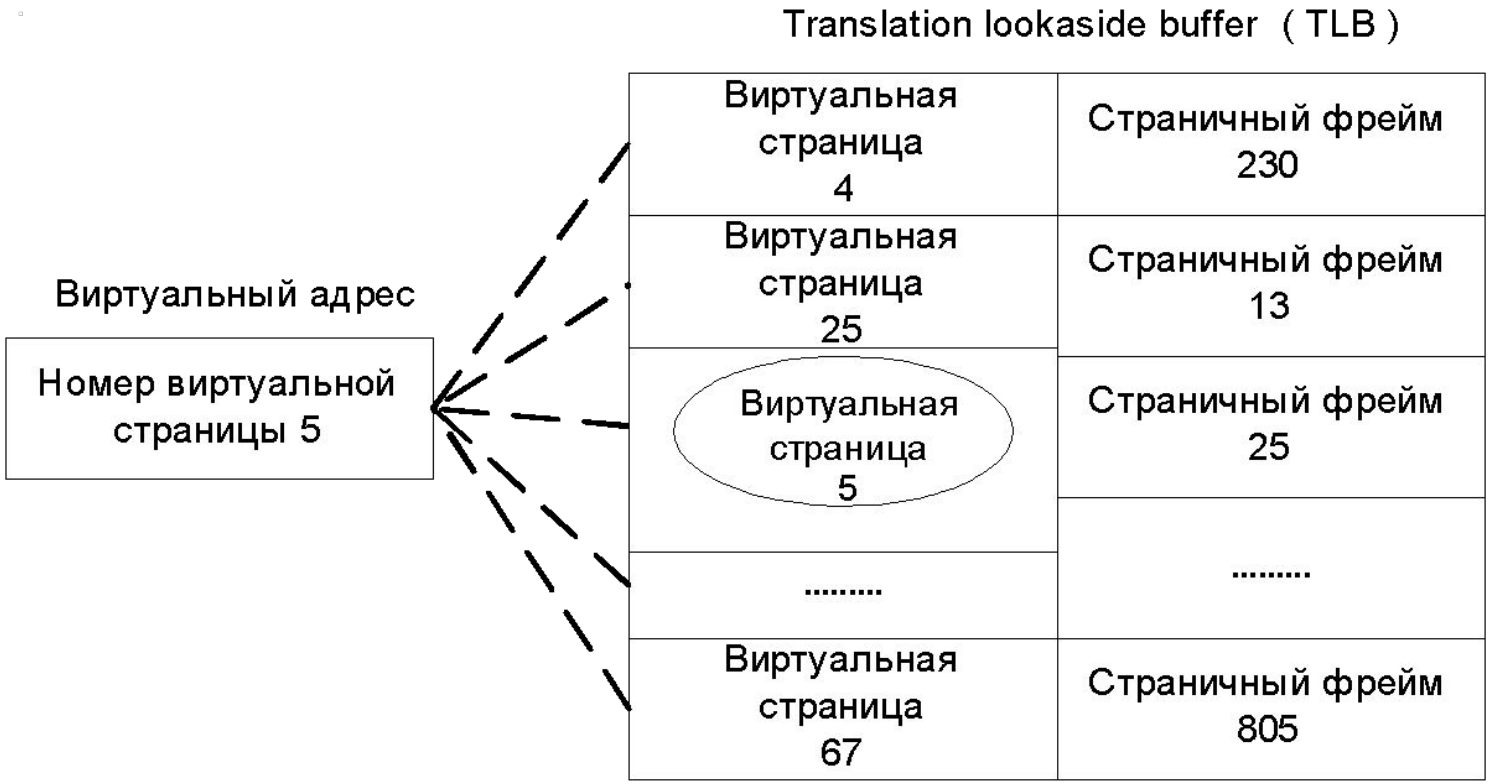
Трансляция адресов

- Индекс таблицы страниц используется как указатель для поиска PTE, который определяет где находится страница. Если она действительная, то PTE содержит PFN соответствующей страницы физической памяти. Если страница оказывается недействительной, то подсистема управления памятью пытается найти ее и сделать действительной
- Если PTE указывает на действительную страницу, для поиска нужных данных используется индекс байта

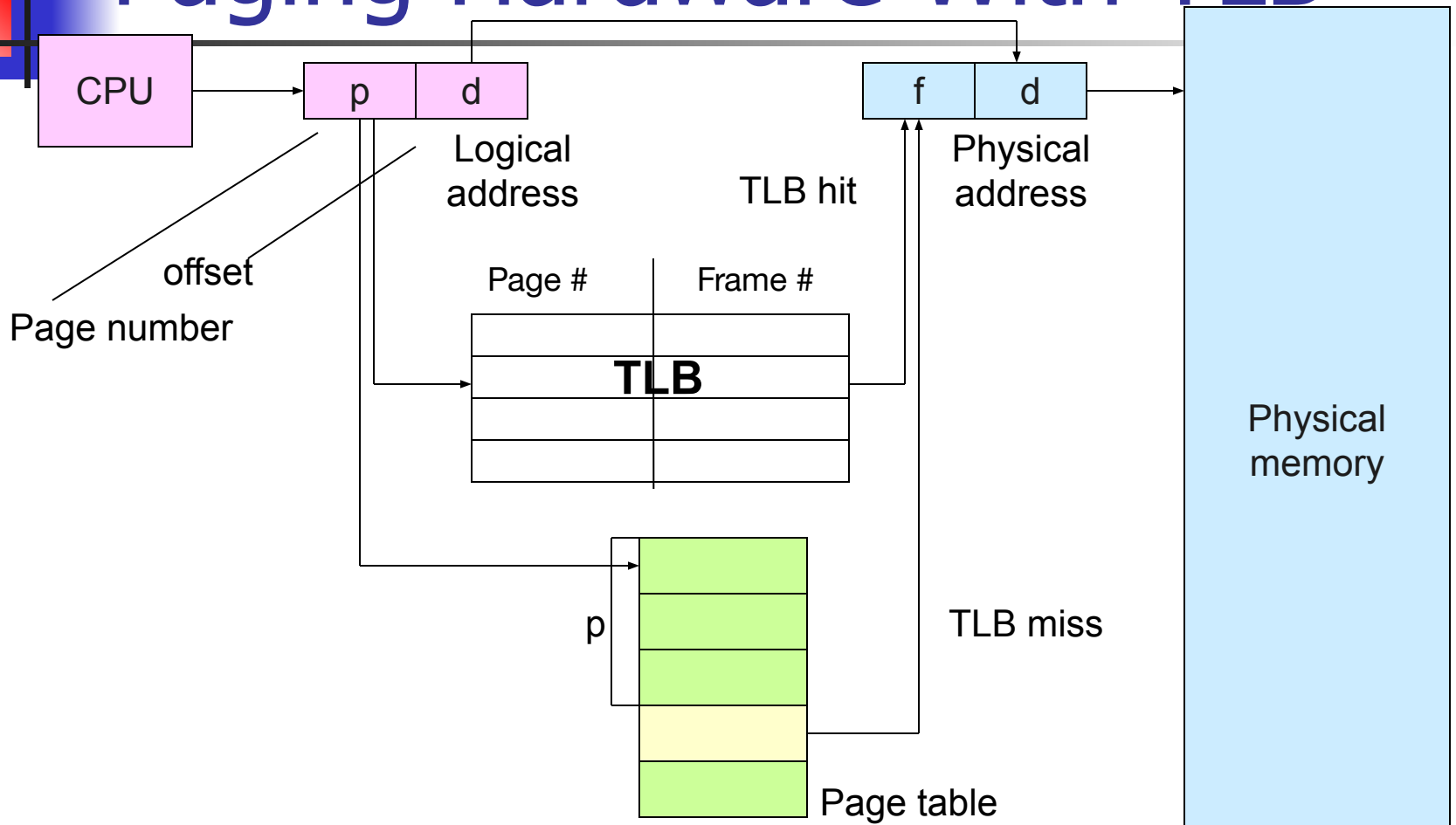
Ассоциативный буфер трансляции

- Трансляция каждого адреса требует двух операций поиска:
  - сначала нужно найти подходящую таблицу страниц в каталоге страниц
  - а затем элемент в этой таблице
- Выполнение этих операций при каждом обращении по виртуальному адресу снижает быстродействие системы
- Процессоры кэшируют транслируемые адреса
- Процессор типа x86 поддерживает такой кэш в виде массива ассоциативной памяти, называемого **ассоциативным буфером трансляции (translation lookaside buffer, TLB)**

Ассоциативный буфер трансляции

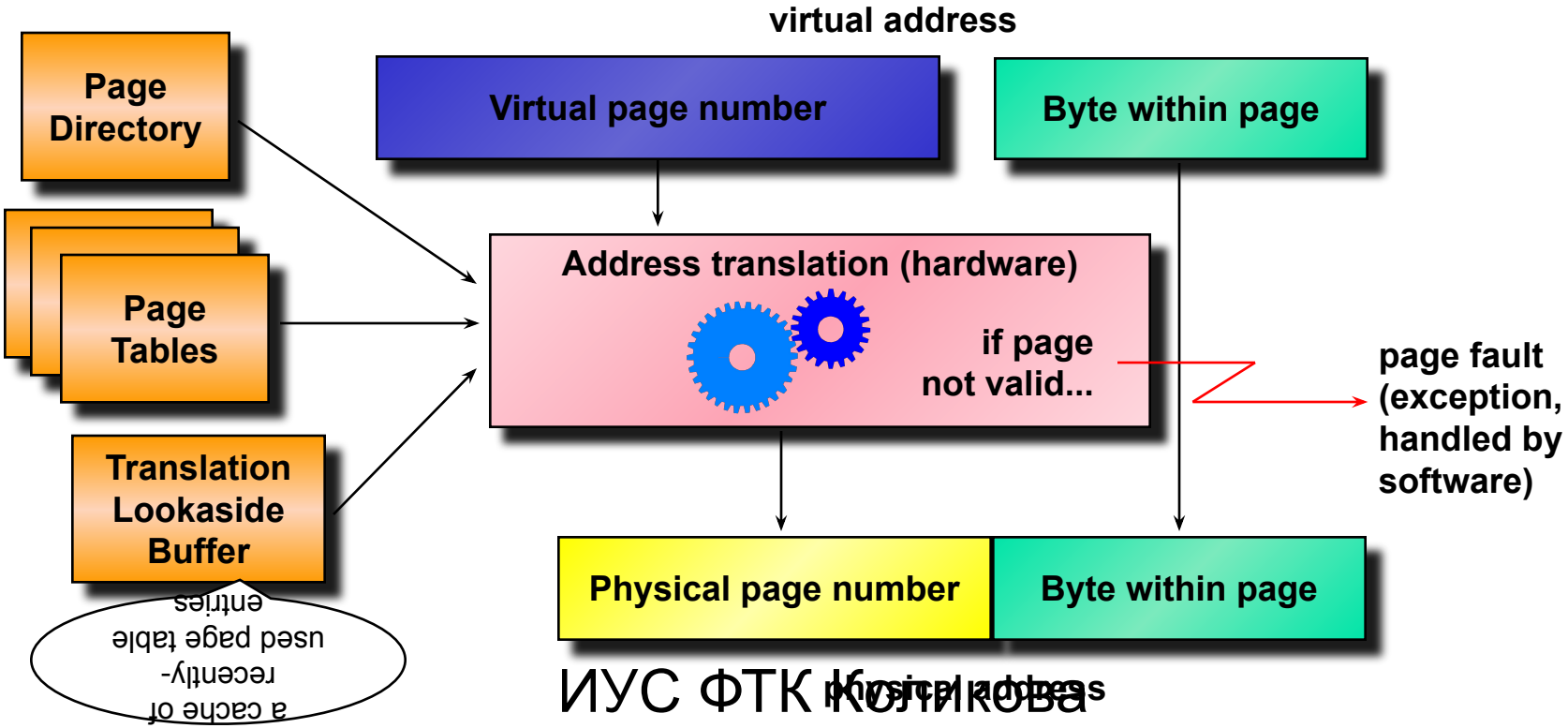


Paging Hardware With TLB



Virtual Address Translation

- The hardware converts each valid virtual address to a physical address



Ассоциативный буфер трансляции

- Виртуальные адреса объединяются в страницы по 2^{12}(4Кб), таким образом в каждом адресном пространстве получается 2^{20} страниц
- Если один вход таблицы страниц имеет размер 2^2 (4б), то для отображения всей виртуальной памяти процесса потребуется ($2^{20} * 2^2 : 2^{12}$) 1024 страничных фрейма
- И это только для адресного пространства одного процесса

Стратегия подкачки и рабочие наборы

- Обычно система виртуальной памяти определяет три типа стратегии:
  - ***стратегия считывания (fetch policy)***
  - ***стратегия размещения (placement policy)***
  - ***стратегия замещения (replacement policy)***

Стратегия подкачки и рабочие наборы

- ***Стратегия считывания (fetch policy)*** определяет, когда надо перемещать страницу с диска в память
  - Можно пытаться загрузить страницы, которые потребуются процессу, до того как он их запросит (заранее)
  - А можно использовать стратегию подкачки по запросе (demand paging policy), в этом случае страница загружается в память только тогда, происходит страничная ошибка

Стратегия подкачки и рабочие наборы

- Диспетчер виртуальной памяти Windows использует алгоритм подкачки по запросу с кластаризацией
- Когда возникает страничная ошибка, диспетчер виртуальной памяти загружает страницу, вызвавшую ошибку, вместе с небольшим количеством окружающих ее страниц
- Это позволяет минимизировать количество страничных ошибок

Стратегия подкачки и рабочие наборы

- При возникновении страничной ошибки система виртуальной памяти должна определить, в какое место физической памяти следует загрузить эту виртуальную страницу
- Здесь начинает действовать ***стратегия размещения (placement policy)***
- Диспетчер виртуальной памяти просто выбирает первый страничный фрейм из списка свободных страничных фреймов
- Если этот список пуст, то диспетчер просматривает другие списки в заданном порядке

Стратегия подкачки и рабочие наборы

- Если страничная ошибка происходит, когда вся физическая память заполнена, то применяется ***стратегия замещения (replacement policy)***
- Она определяет какую страницу нужно извлечь из памяти, чтобы освободить место для новой страницы
- Обычно используют следующие стратегии:
  - замещение используемого меньше всего (least recently used)
  - первым пришел, первым ушел (first in, first out – FIFO)

Стратегия подкачки и рабочие наборы

- В первом случае необходимо, чтобы система виртуальной памяти отслеживала, когда страница использовалась последний раз
- Если нужно загрузить новую страницу, то на диск выгружается та страница, которая не использовалась дольше всех остальных, а ее страничный фрейм освобождается
- В соответствии со вторым алгоритмом, на диск перемещается та страница, которая дольше всех находилась в памяти, независимо от того, как часто она использовалась

Стратегия подкачки и рабочие наборы

- Стратегии замещения делятся на:
  - ***локальные (local replacement policy)***
  - ***глобальные (global replacement policy)***
- При стратегии локального замещения каждому процессу выделяется фиксированное (динамически настраиваемое) число страничных фреймов
- Когда процесс использовал все выделенные ему фреймы, система виртуальной памяти удаляет из физической памяти одну из его страниц при каждой страничной ошибке в данном процессе

Стратегия подкачки и рабочие наборы

- В случае стратегии глобального замещения для обработки страничной ошибки используется любой страничный фрейм, независимо от того, принадлежит ли он процессу, в котором произошла эта страничная ошибка
- Так при использовании стратегии глобального замещения FIFO необходимо отыскать страницу, которая дольше остальных находилась в памяти
- А при использовании стратегии локального замещения FIFO необходимо отыскать самую старую страницу данного процесса

Стратегия подкачки и рабочие наборы

- Стратегия глобального замещения создает ряд проблем:
  - процессы подвержены влиянию других процессов
  - плохое приложение может подорвать работу всей системы
- По этим причинам диспетчер виртуальной памяти Windows использует стратегию локального замещения FIFO
- Для реализации этого подхода ему необходимо для каждого процесса отслеживать его страницы, находящиеся в памяти
- Это множество страниц называется ***рабочим набором (working set)*** процесса

Стратегия подкачки и рабочие наборы

- Минимальный размер рабочего набора - минимальное число страниц процесса, которые будут гарантированно находиться в памяти во время его исполнения
- Максимальный размер рабочего набора
- Когда физической памяти становится недостаточно, диспетчер виртуальной памяти использует ***автоматическое урезание рабочего набора*** (***automatic working set trimming***), для увеличения объема свободной памяти
- Диспетчер виртуальной памяти, используя стратегии локального замещения и автоматического урезания рабочего набора, пытается обеспечить максимально возможную производительность для каждого процесса
  - Процесс может изменить минимальный и максимальный размер своего рабочего набора, вызвав сервис объекта процесс

База данных страничных фреймов

- Таблицы страниц процесса содержат информацию о том, в каком месте физической памяти расположены виртуальные страницы
- Кроме того, диспетчеру виртуальной памяти нужна структура данных для отслеживания состояния физической памяти
- Чтобы знать свободен ли данный страничный фрейм или если нет, то кто его использует

База данных страничных фреймов

- Для этого используется **база данных страничных фреймов (page frame database)**
- Она представляет собой массив элементов (от 0 до число страничных фреймов в системе – 1)
- Каждый элемент содержит информацию о соответствующем страничном фрейме
- Действительные элементы таблицы страниц указывают на элементы базы данных страничных фреймов
- Диспетчер виртуальной памяти использует этот указатель, когда процесс обращается по действительному виртуальному адресу, чтобы найти физическую страницу, соответствующую виртуальному адресу

База данных страничных фреймов

- Некоторые недействительные элементы таблицы страниц также ссылаются на элементы базы данных страничных фреймов.
- Эти переходные элементы таблицы страниц указывают на страничные фреймы, которые могут быть, но еще не использованы повторно и их содержимое пока не изменилось
- Если процесс обращается к одной из таких страниц, прежде чем ее использовал другой процесс, диспетчер виртуальной памяти может ее быстро восстановить

База данных страничных фреймов

База данных страничных фреймов

- Недействительные элементы таблицы страниц содержат адреса на диске, по которым хранятся страницы
- При обращении процесса к ним происходит страничная ошибка, и диспетчер виртуальной памяти считывает содержимое страницы с диска
- Страничный фрейм может находится в одном из шести состояний:
  - **Действительный** – страничный фрейм используется процессом и на него указывает действительный элемент таблицы страниц
  - **Обнуленный** – страничный фрейм свободен и инициализирован нулями
  - **Свободный** - страничный фрейм свободен, но не инициализирован

База данных страничных фреймов

- **Резервный** – данный фрейм использовался процессом, но был удален из его рабочего набора. Соответствующий элемент таблицы страниц недействителен, но помечен как переходный
- **Измененный** – аналогично резервному, но процесс, использовавший данный страничный фрейм, осуществил запись в него и содержимое еще не записано на диск. Соответствующий элемент таблицы страниц недействителен, но помечен как переходный
- **Плохой** – страничный фрейм вызвал ошибку четности или другой аппаратный сбой, и его нельзя использовать

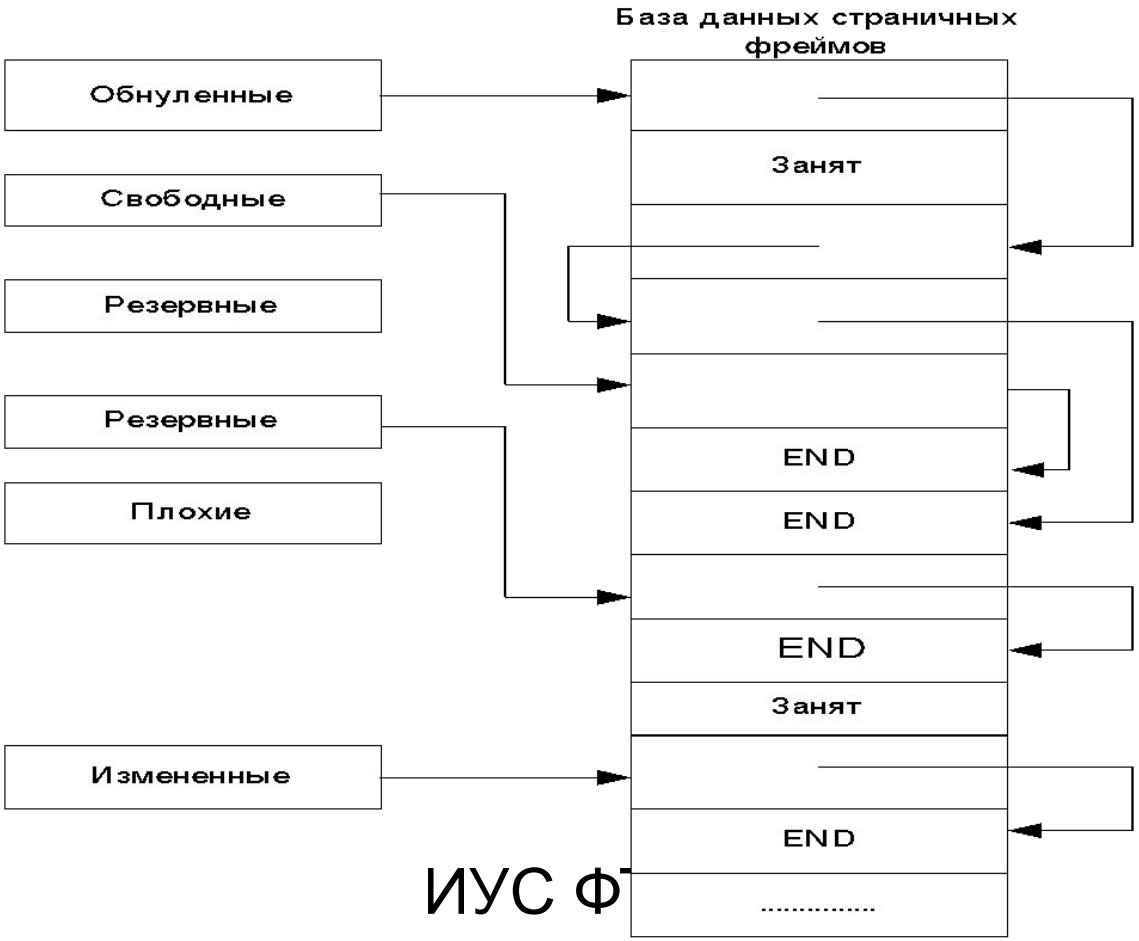
База данных страничных фреймов

- В базе данных страничные фреймы, находящиеся в одном и том же состоянии, группируются и получается пять отдельных списков:
  - **список обнуленных**
  - **список свободных**
  - **список резервных**
  - **список измененных**
  - **список плохих страниц**

База данных страничных фреймов

- В данные списки включены все страничные фреймы, которые не используются в данный момент
- Указатели на используемые фреймы содержатся в таблице страниц соответствующего процесса
- Если процесс освобождает страничный фрейм, то диспетчер виртуальной памяти помещает его обратно в один из списков

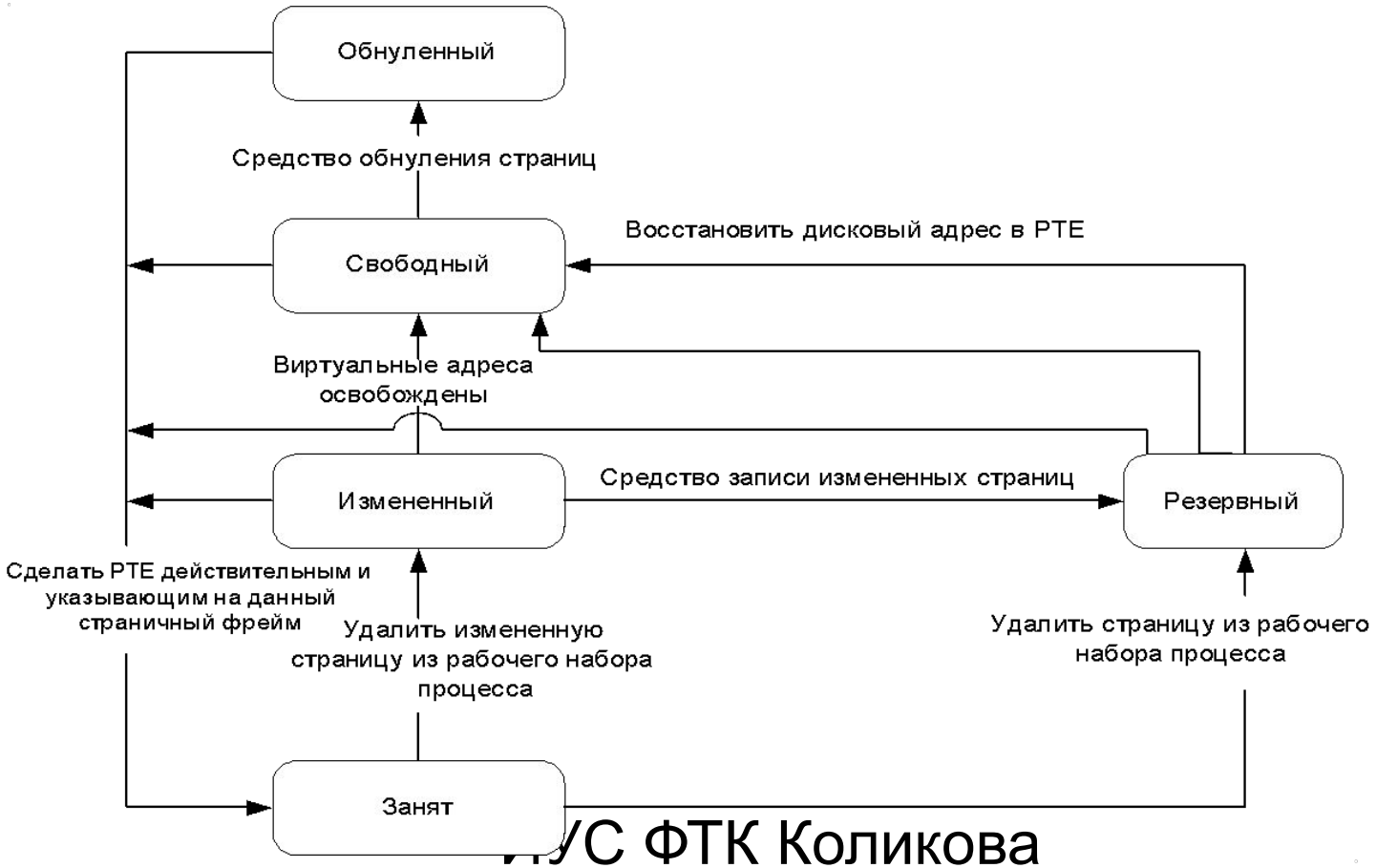
База данных страничных фреймов



База данных страничных фреймов

- Если диспетчеру виртуальной памяти требуется инициализированный нулями страничный фрейм для обработки страничной ошибки, то он пытается взять первый из списка обнуленных
- Если этот список пуст, то диспетчер выбирает фрейм из списка свободных и обнуляет его
- Если диспетчеру не нужна инициализированная страница, то он берет первую из списка свободных
- Если этот список пуст, то используется первая из списка обнуленных
- Если оба списка пусты, то используется список резервных

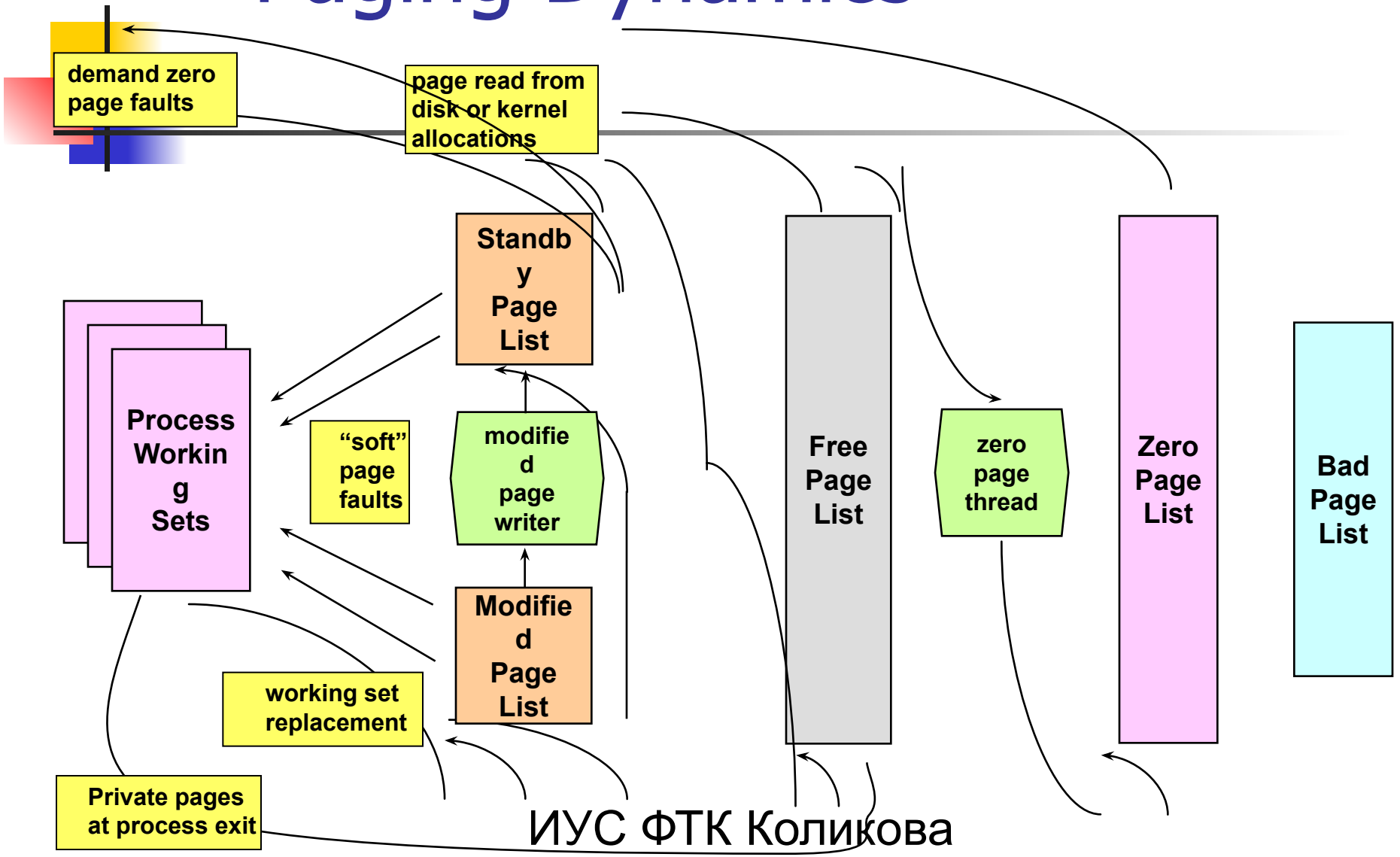
База данных страничных фреймов



База данных страничных фреймов

- Когда количество страниц в списках обнуленных, свободных и резервных снижается до порогового значения, поток под названием **средство записи измененных страниц (modified page writer)** пробуждается и записывает содержимое измененных страниц на диск, после чего помещает их в список резервных

Paging Dynamics



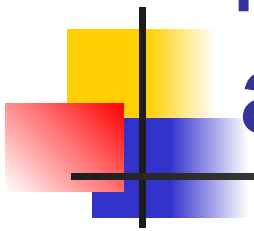
Дескрипторы виртуальных адресов

- Момент загрузки страниц в память диспетчер памяти определяет, используя алгоритм подкачки по требованию
- Страница загружается с диска, если поток, обращаясь к ней, вызвал ошибку страницы
- Подкачка по требованию является одной из форм **отложенных вычислений (lazy evalution**), операция выполняется только при ее абсолютной необходимости

Дескрипторы виртуальных адресов

- Диспетчер памяти использует этот же механизм и при формировании таблиц страниц
- Например, когда поток передает с помощью VirtualAlloc, диспетчер памяти мог бы немедленно создать таблицы страниц, необходимые для доступа ко всему объему выделенной памяти
- Вместо этого диспетчер виртуальной памяти откладывает формирование таблицы страниц до тех пор, пока поток не вызовет ошибку страницы

Дескрипторы виртуальных адресов



- Как же тогда диспетчер виртуальной памяти определяет какие адреса свободны?
- Чтобы решить эту проблему диспетчер поддерживает набор структур данных, которые позволяют вести учет зарезервированных и свободных виртуальных адресов в адресном пространстве процесса
- Эти структуры данных называются дескрипторами виртуальных адресов (virtual address descriptors, VAD)
- Для каждого процесса диспетчер памяти поддерживает свой набор VAD, описывающий состояние адресного пространства этого процесса

Дескрипторы виртуальных адресов

- При первом обращении потока по какому-либо адресу диспетчер памяти должен создать PTE страницы, содержащий данный адрес
- Для этого он находит VAD, чей диопазон включает нужный адрес, и использует его информацию для заполнения PTE
- Если адрес выпадает из диопазонов VAD или находится в зарезервированном, но не переданном диопазоне адресов, диспетчер памяти узнает, что поток не выделил память до попытки ее использования, и генерирует нарушение доступа