Виртуальная память – I

Основы информатики.

Компьютерные основы программирования

goo.gl/X7evF

На основе **CMU** 15-213/18-243:
Introduction to Computer Systems

goo.gl/TDDVV

Лекция **13**, **8** Мая, 2017

Лектор:

Дмитрий Северов, кафедра информатики 608 КПМ

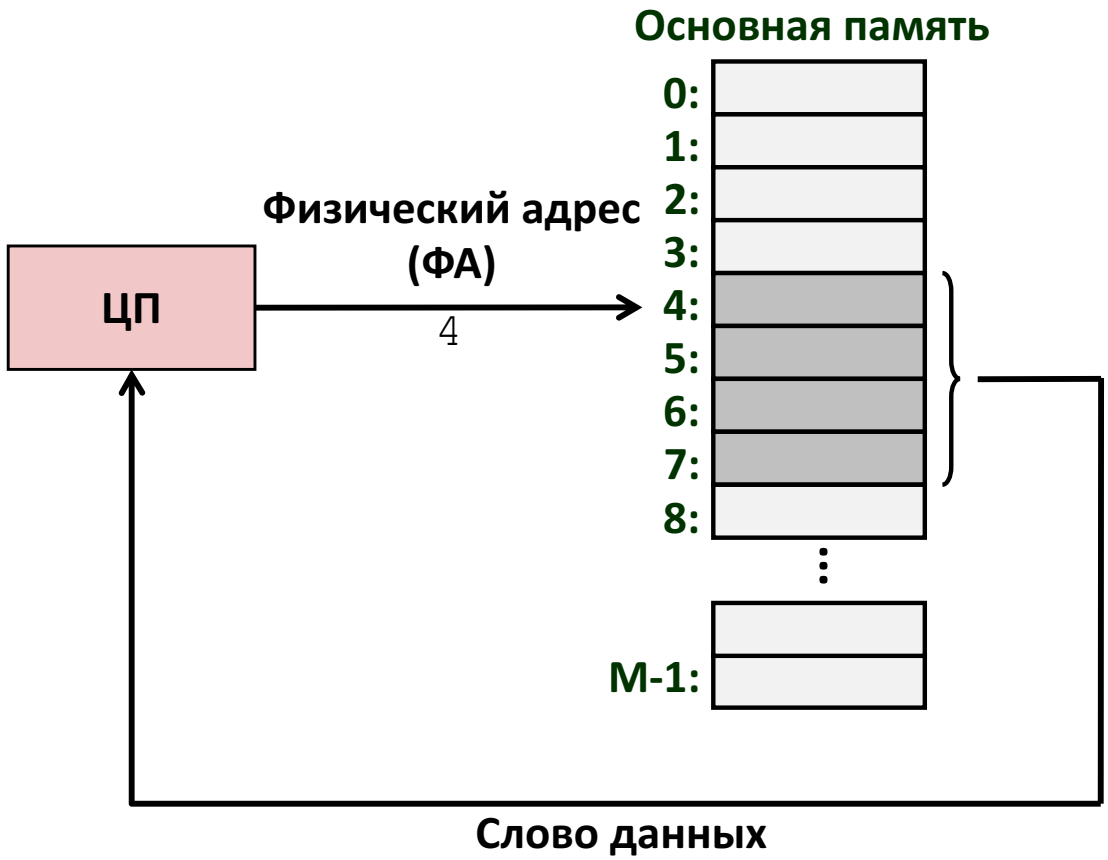
dseverov@mail.mipt.ru

cs.mipt.ru/wp/?page_id=346

Виртуальная память – 1: понятия

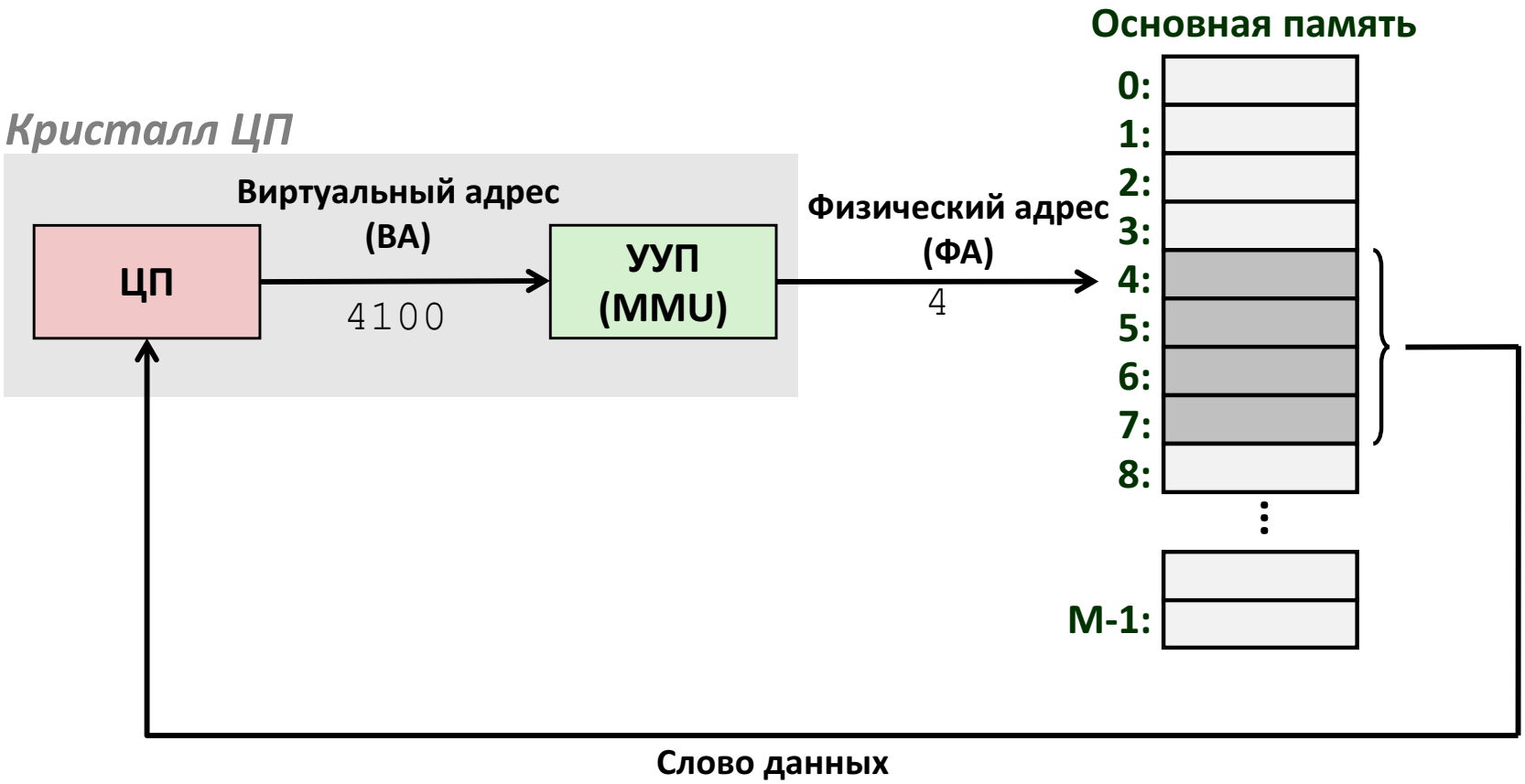
- **Пространства адресов**
- **ВП как средство кеширования**
- **ВП как средство управления памятью**
- **ВП как средство защиты памяти**
- **Трансляция адресов**

Система с физической адресацией



- Используется в “простых” системах – микроконтроллерах, встроенных в датчики, регуляторы света, часы

Системы с виртуальной адресацией



- Используется во всех современных серверах, ПК, смартфонах
- Одна из замечательных идей в информатике

Адресные пространства

- **Линейное адресное пространство:** Упорядоченное множество смежных неотрицательных целых адресов: {0, 1, 2, 3 … }

- **Виртуальное адресное пространство:** Множество N = 2^n виртуальных адресов: {0, 1, 2, 3, …, N-1}

- **Физическое адресное пространство:** Множество M = 2^m физических адресов {0, 1, 2, 3, …, M-1}

- **Четкое различием между данными (байтами) и их атрибутами (адресами)**
- **Каждый объект может иметь несколько адресов**
- **Каждый байт в основной памяти имеет:**
  один физический адрес, один (или более) виртуальных адресов

Виртуальная память нужна, чтобы…

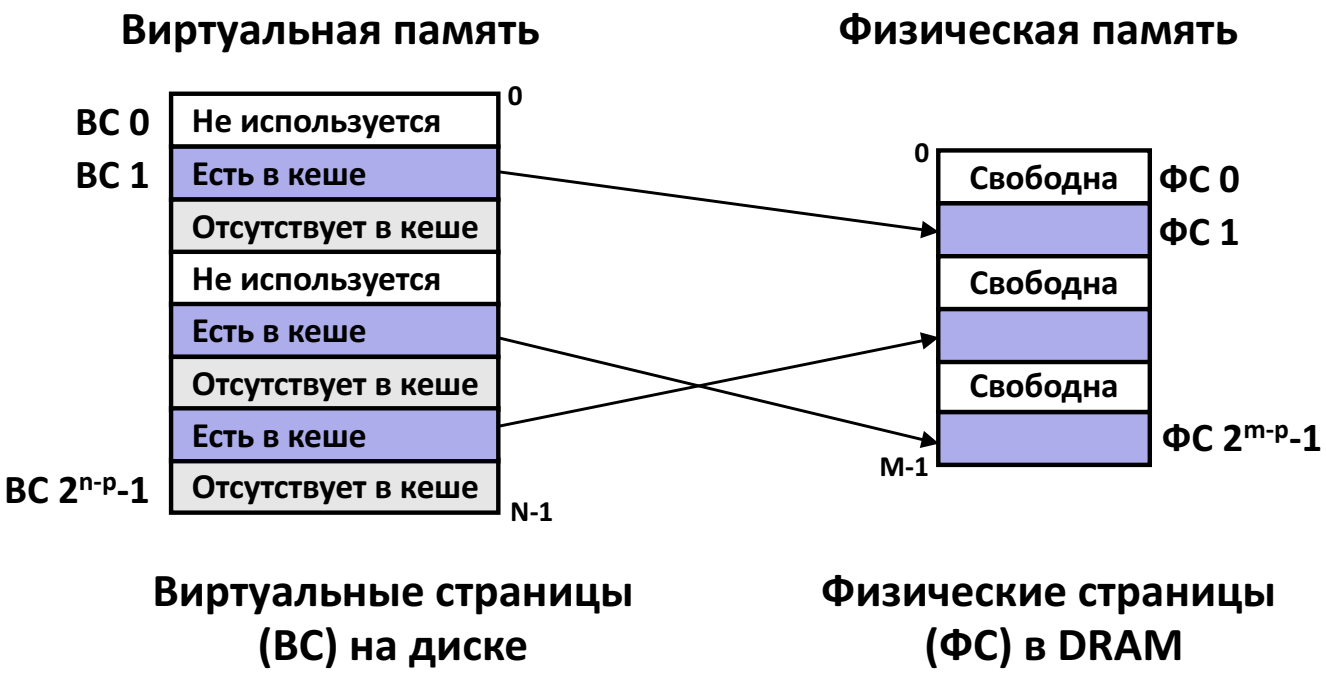
- **использовать физическую память эффективно**
  - DRAM – кеш к частям виртуального пространства
- **упростить управление памятью**
  - Каждый процесс получает типовой экземпляр линейного адресного пространства
- **изолировать адресные пространства**
  - Ни один процесс не может обращаться к памяти другого
  - Программы пользователей не имеют доступа привилегированной информации ядра ОС

Виртуальная память – 1: понятия

- Пространства адресов
- ВП как средство кеширования
- ВП как средство управления памятью
- ВП как средство защиты памяти
- Трансляция адресов

ВП как средство кеширования

- ***Виртуальная память -*** **массив N смежных байт на диске**
- **Содержимое массива на диске кешируется в *физической памяти* (*DRAM cache*)**
  - Блоки этого кеша – *страницы* размером P = 2^p байт



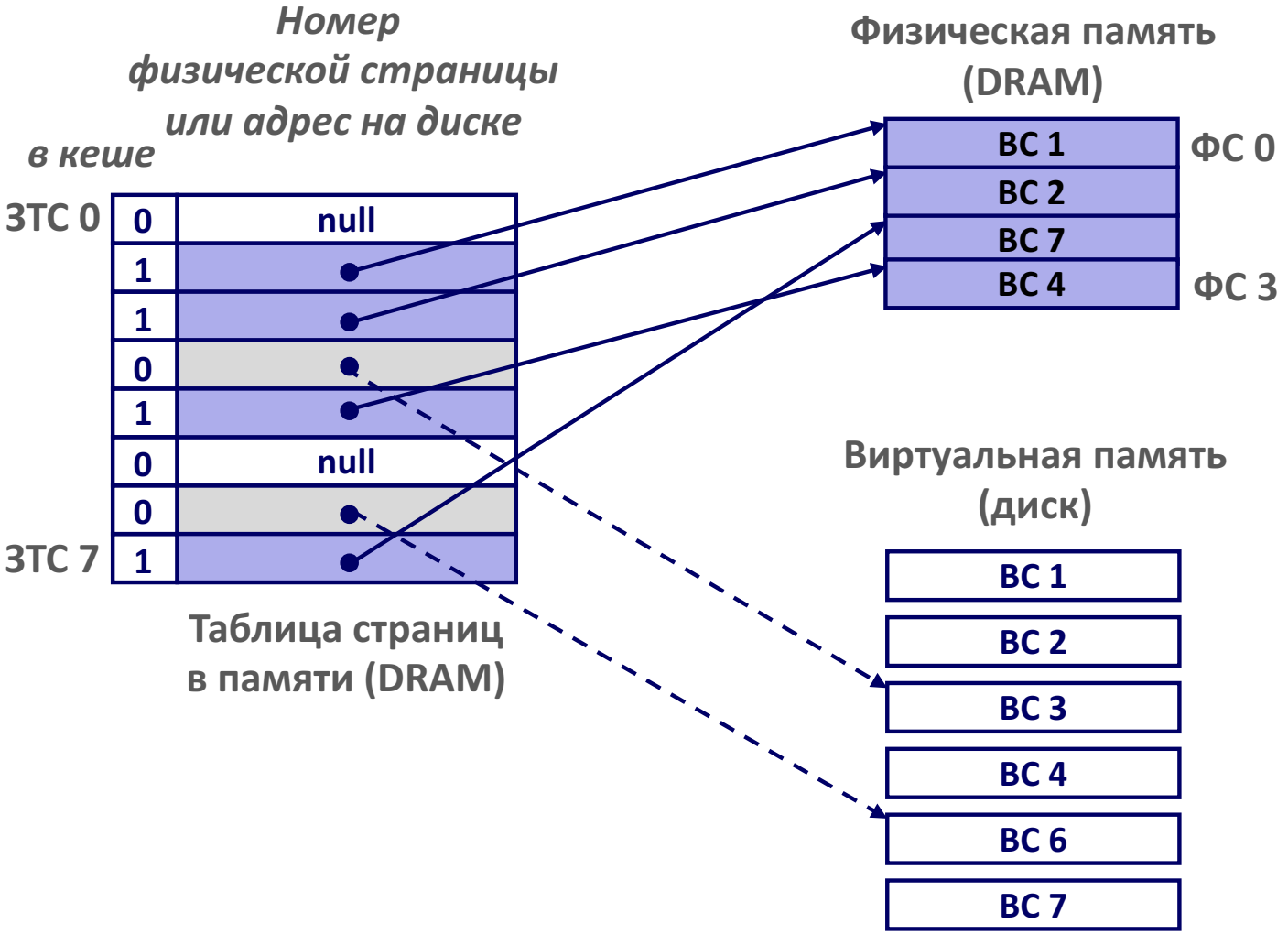
Организация DRAM-кеша

- **Определяется огромной ценой промаха**
  - DRAM на 2 порядка (***100 раз***) медленнее SRAM
  - Диск на 4 порядка (***10 тысяч раз***) медленнее DRAM

- **Следствия**
  - Большой размер страницы (блока): обычно 4-8 KB, иногда 4 MB
  - Полностью ассоциативен
    - любая ВС может быть размещена в любой ФС
    - требует “большую” функцию отображения, в отличие от кешей ЦП
  - Высокоизощрённые, дорогостоящие алгоритмы замены
    - Слишком сложные и неустоявшиеся для аппаратной реализации
  - Write-back, но не write-through

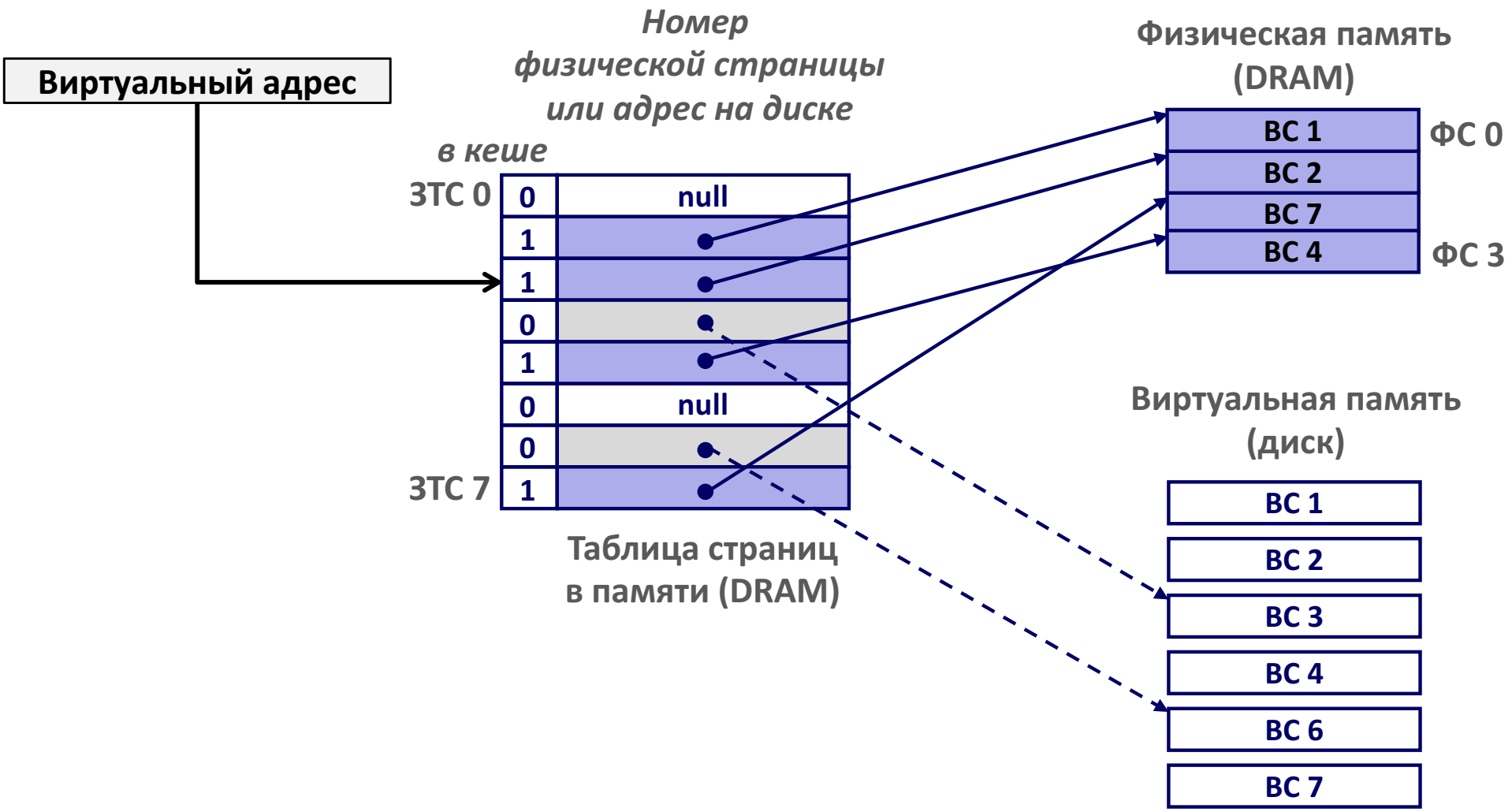
Таблица страниц – ключевая структура

- **Массив записей [в таблицу страниц] (ЗТС) отображений виртуальных страниц на физические.**
  - Отдельная для каждого процесса структура данных ядра ОС в DRAM



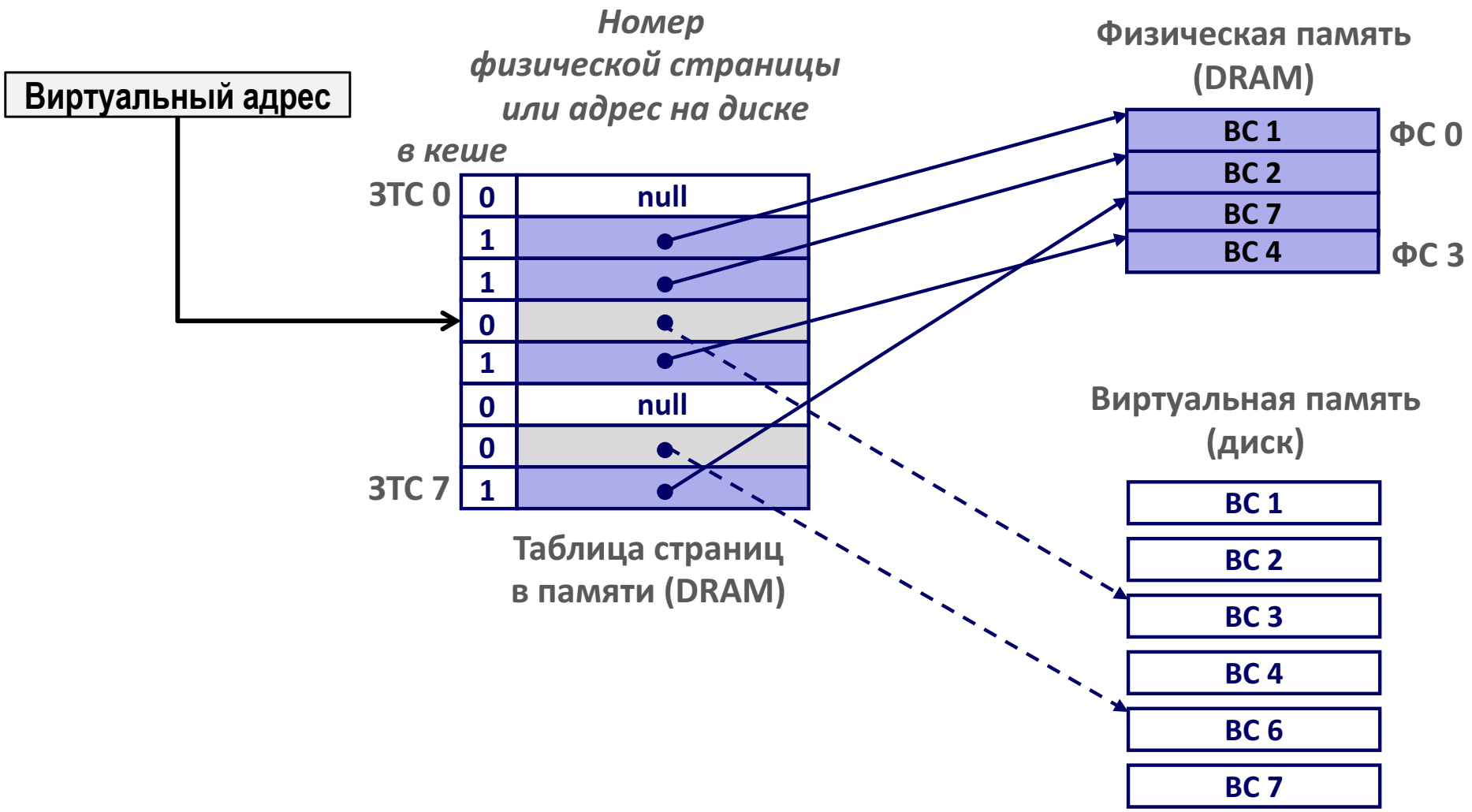
Страничное попадание

- ***Страничное попадание:*** **ссылка на страницу ВП говорит, что страница находится в DRAM (попадание DRAM-кеша)**



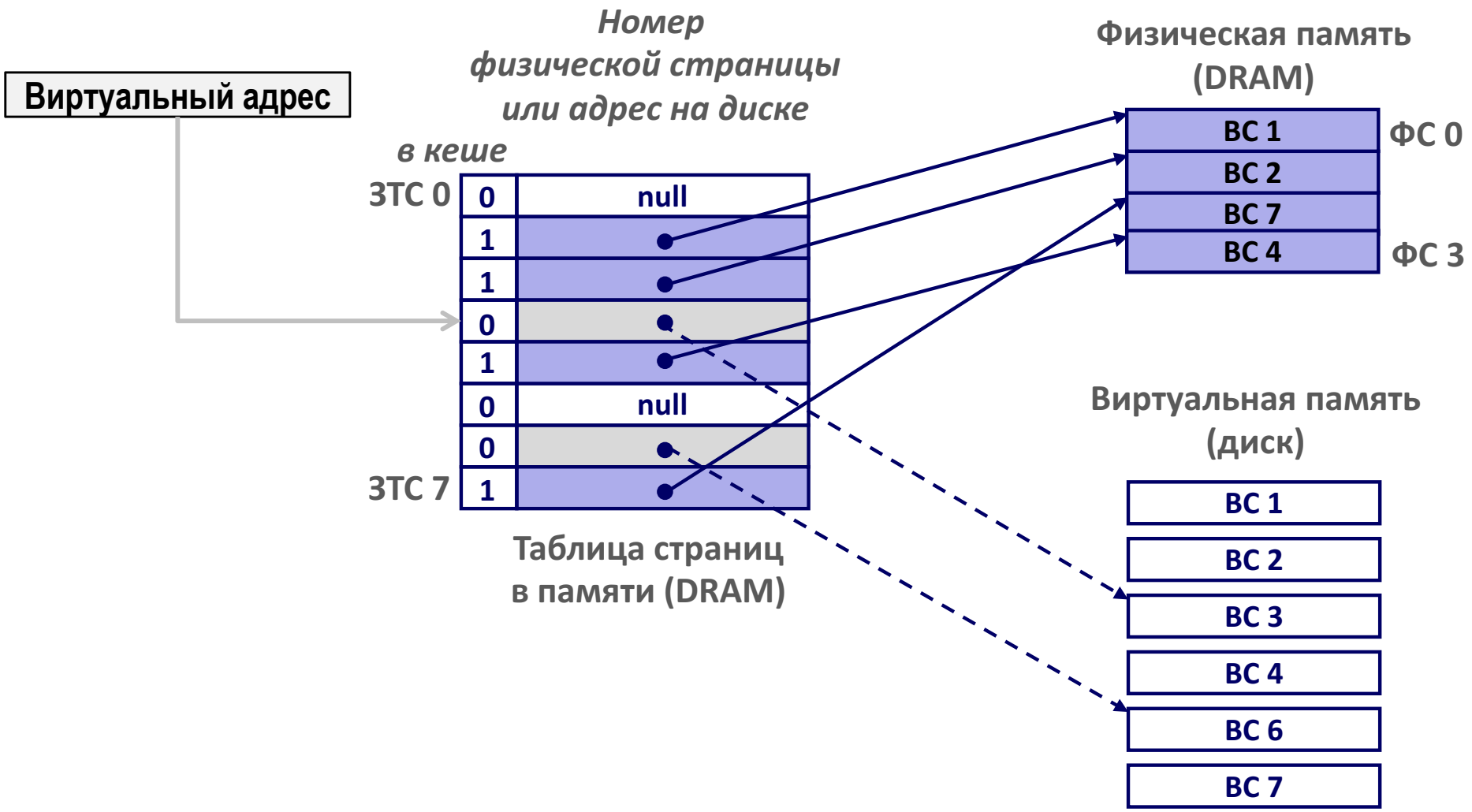
Страничный сбой

- ***Страничный сбой:*** **ссылка на страницу ВП говорит, что страница отсутствует в DRAM (промах DRAM-кеша)**



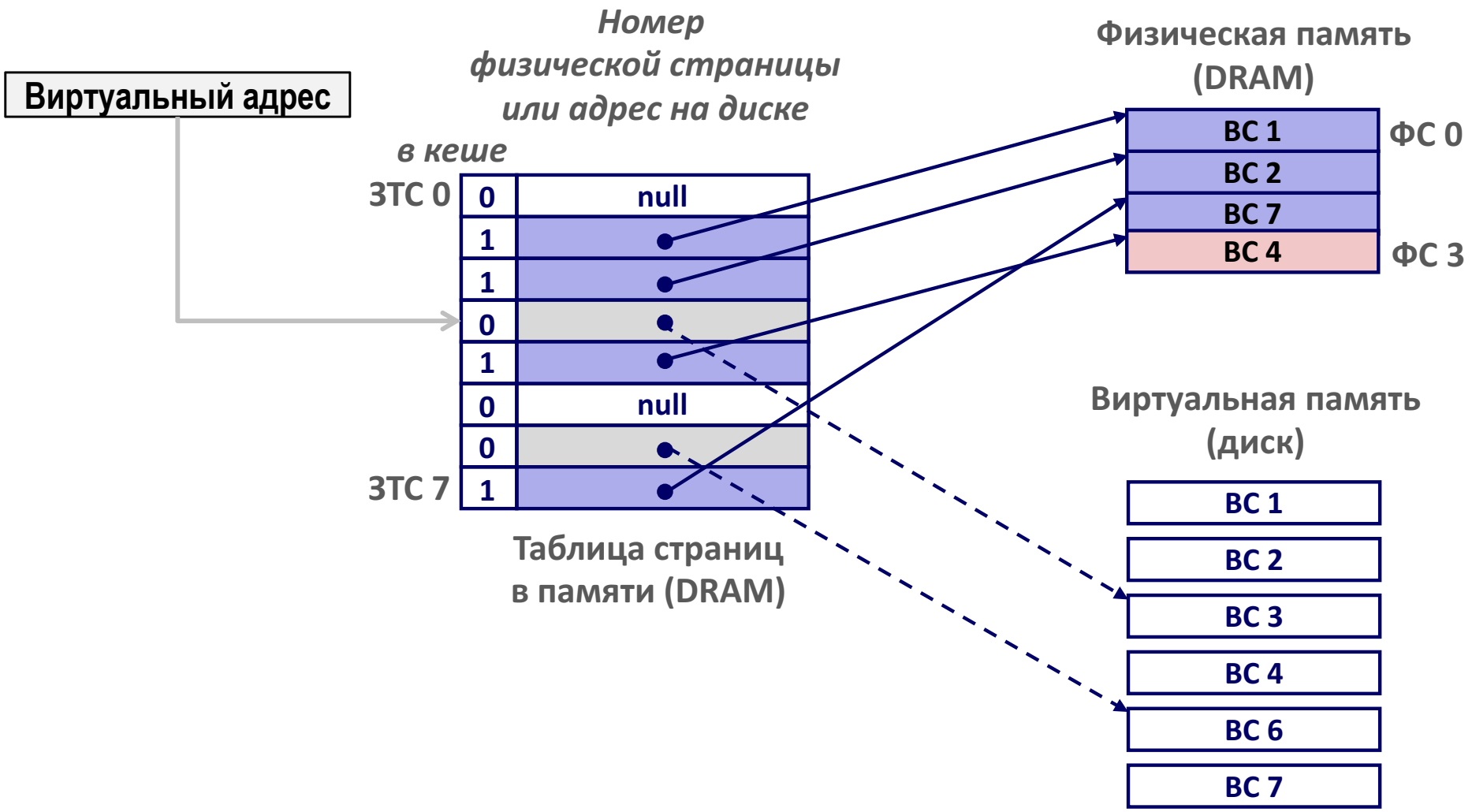
Обработка страничного сбоя – 1

- Страничный промах вызывает страничный сбой (исключение)



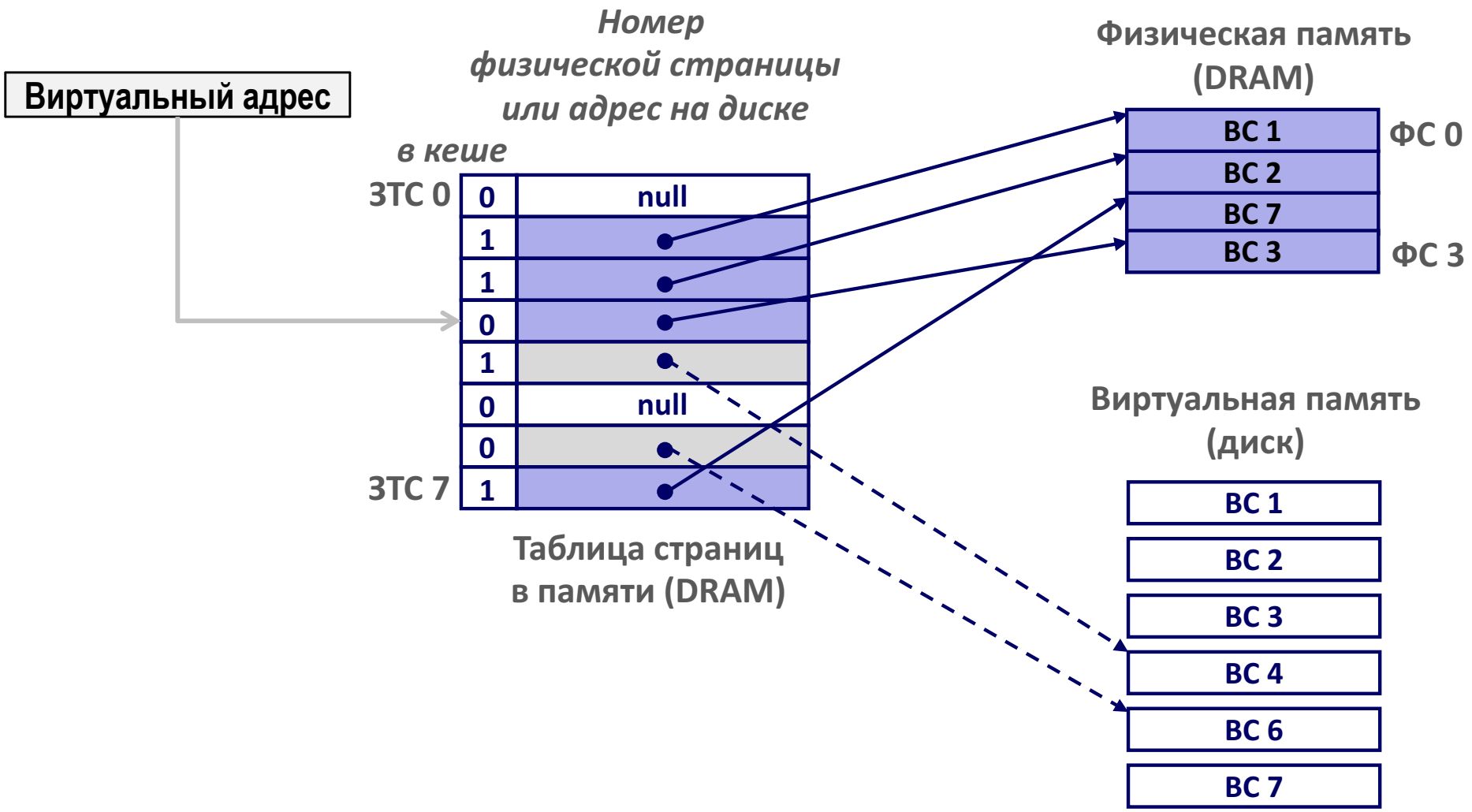
Обработка страничного сбоя – 2

- Страничный промах вызывает страничный сбой (исключение)
- Обработчик страничного сбоя выбирает жертву откачки (здесь ВС 4)



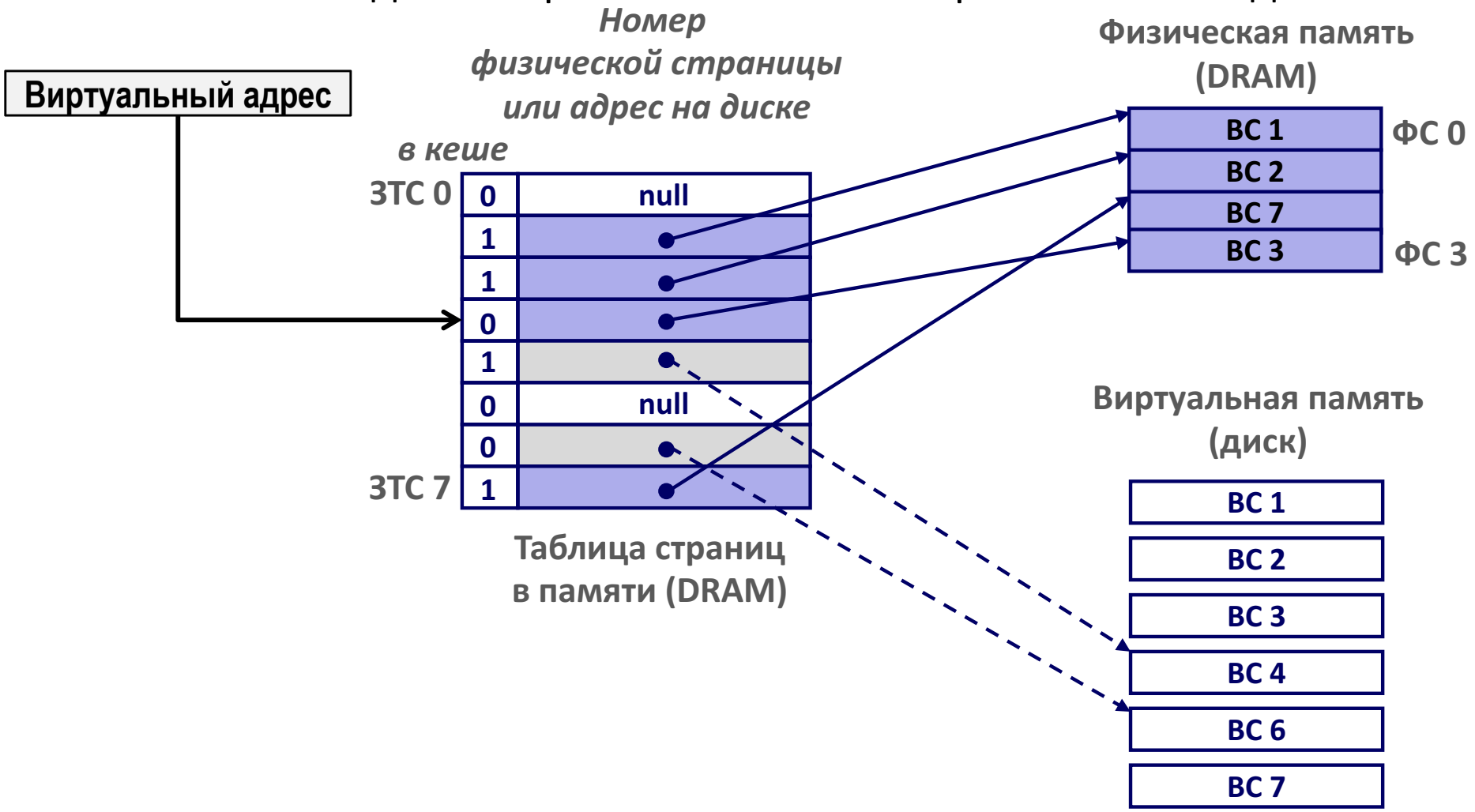
Обработка страничного сбоя - 3

- Страничный промах вызывает страничный сбой (исключение)
- Обработчик страничного сбоя выбирает жертву откачки (здесь ВС 4)
- Обработчик подкачивает с диска в память нужную страницу (здесь ВС 3)



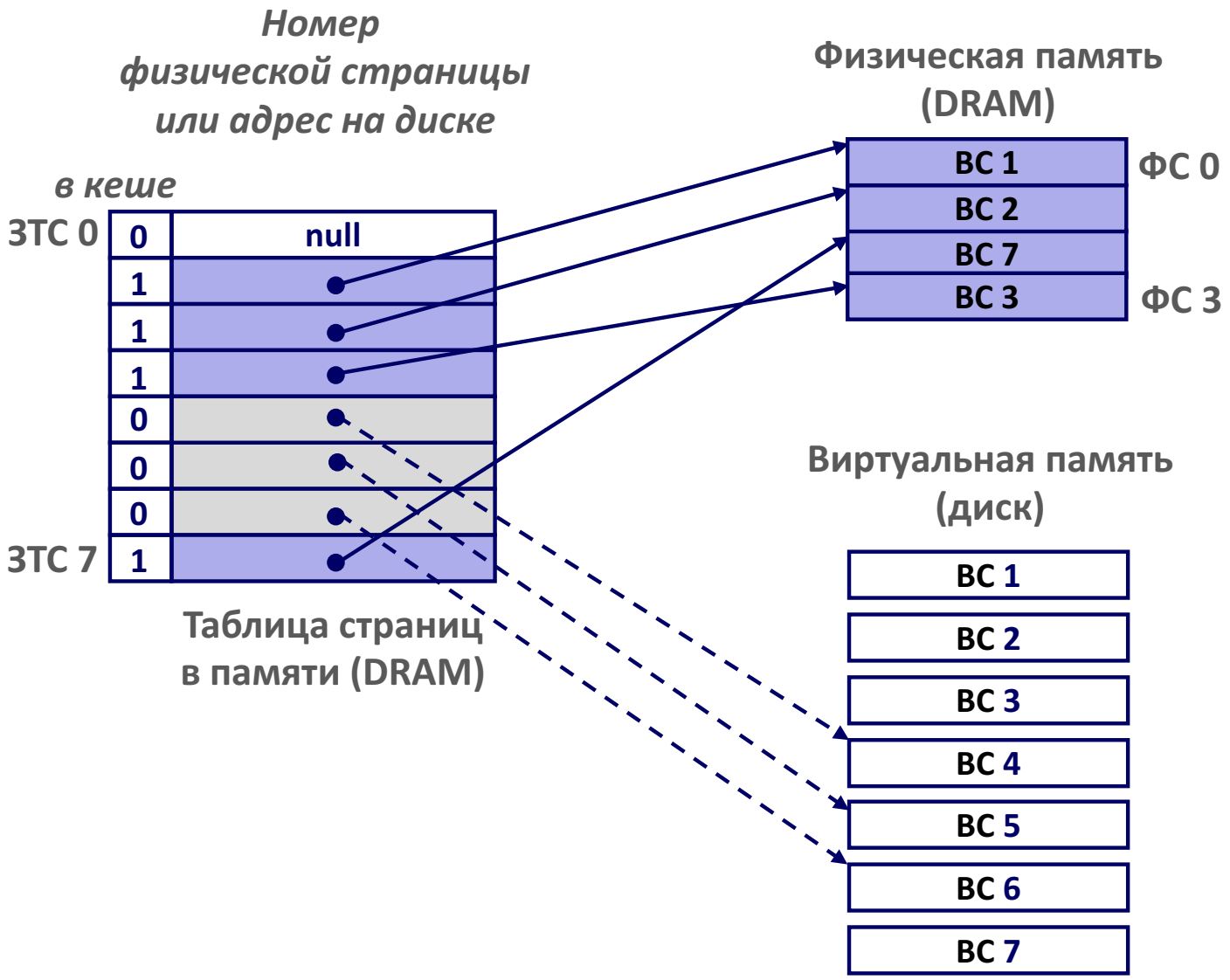
Обработка страничного сбоя - 4

- Страничный промах вызывает страничный сбой (исключение)
- Обработчик страничного сбоя выбирает жертву откачки (здесь ВС 4)
- Обработчик подкачивает с диска в память нужную страницу (здесь ВС 3)
- Сбойная команда повторно выполняется : страничное попадание!



Занятие страниц

- **Занятие новой страницы (ВС 5) виртуальной памяти.**



Локальность снова выручает!

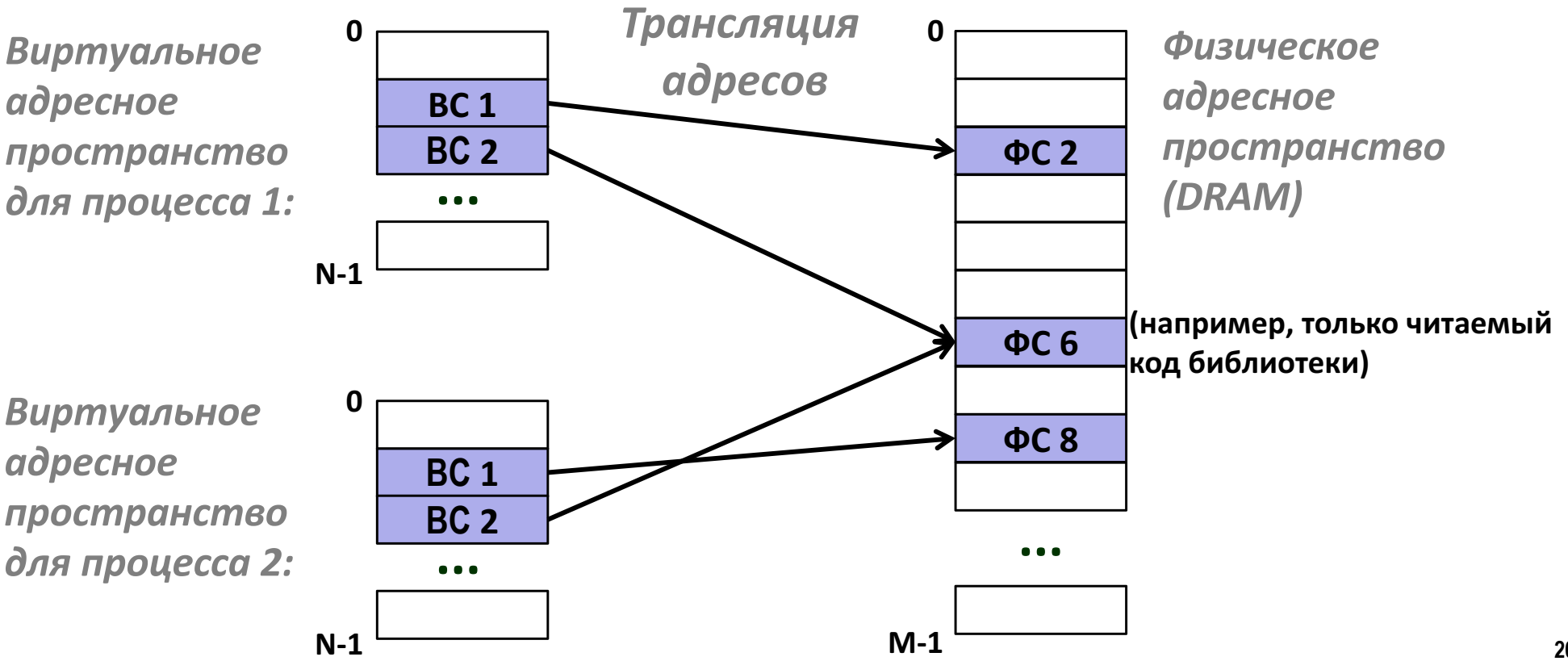
- **Виртуальная память работает благодаря локальности**
- **В каждый момент времени, программы стремятся доступаться к набору активных виртуальных страниц – *рабочему набору***
  - Программы с лучшей временной локальностью будут иметь рабочий набор меньшего размера
- **Если размер рабочего набора < размера основной памяти**
  - Хорошая производительность процесса после неизбежных промахов
- **Если сумма размеров рабочих наборов > размера основной памяти**
  - *Пробуксовка (Thrashing):* деградация производительности при непрерывной откачке/подкачке страниц

Виртуальная память – 1: понятия

- Пространства адресов
- ВП как средство кеширования
- ВП как средство управления памятью
- ВП как средство защиты памяти
- Трансляция адресов

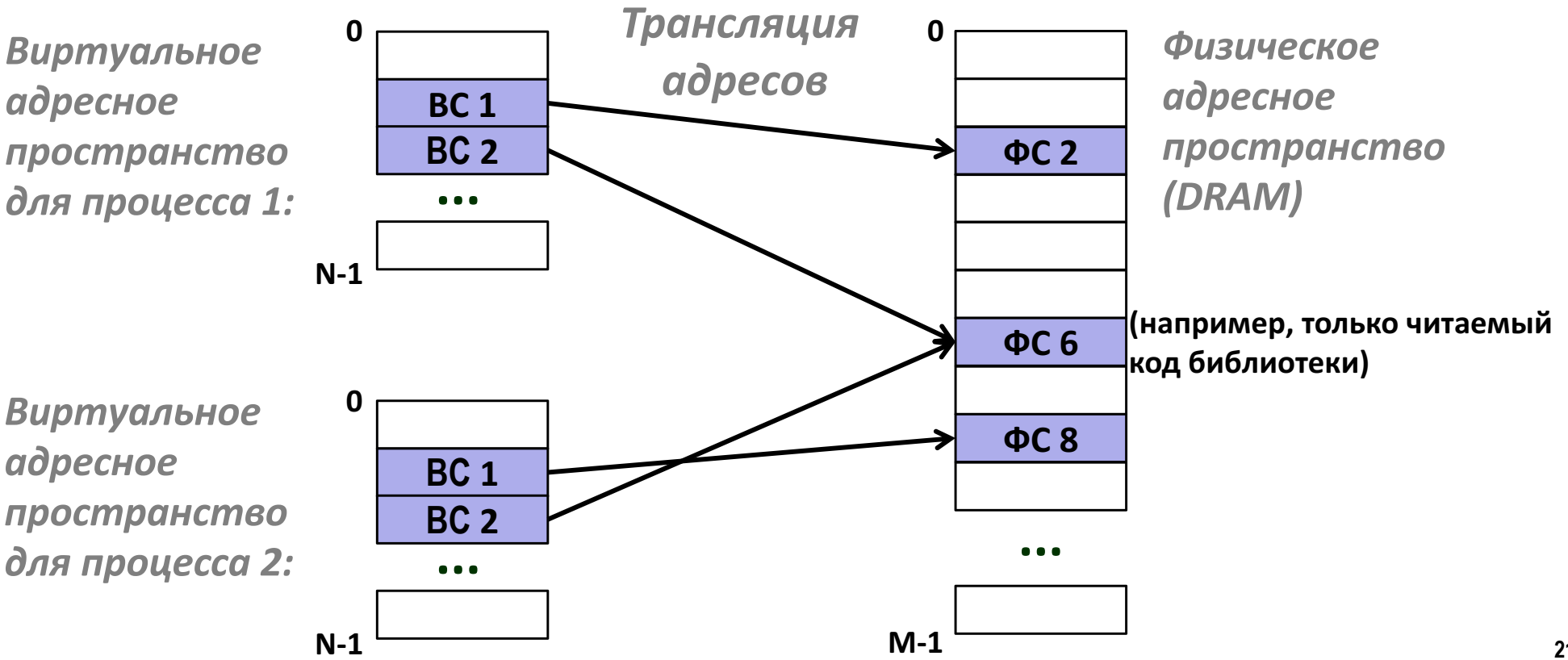
ВП как средство управления памятью - 1

- **Ключевая идея: каждому процессу – собственное виртуальное адресное пространство**
  - память представляется простым линейным массивом
  - отображение разбрасывает адреса по физической памяти
    - удачные отображения упрощают распределение и управление



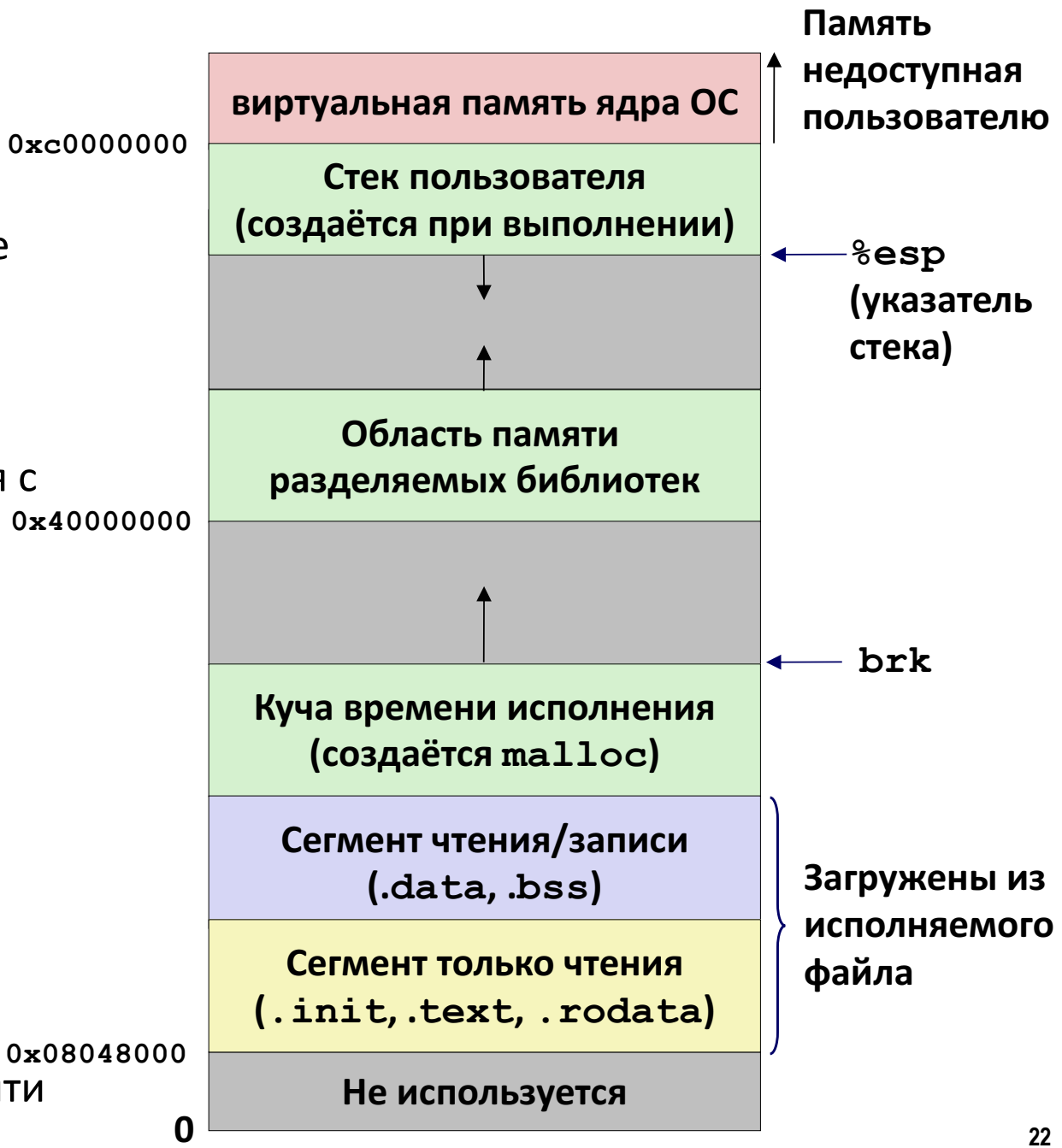
ВП как средство управления памятью - 2

- **Распределение памяти**
  - Любая виртуальная страница может отображаться на любую физическую
  - Виртуальная отображается в различные физические в разное время
- **Разделение кода и данных между процессами**
  - Отображение виртуальных страниц на одну физическую (здесь: ФС 6)



Упрощение связывания и загрузки

- **Связывание**
  - Все программы имеют сходное виртуальное адресное пространство
  - Код, стек и разделяемые библиотеки всегда начинаются с одинаковых адресов
- **Загрузка**
  - **`execve()`** лишь резервирует виртуальные страницы и для разделов .text and .data, т.е. помечает PTE как «не в кеше»
  - Разделы **`.text`** и **`.data`** подгружаются, страница за страницей по требованиям подсистемы виртуальной памяти

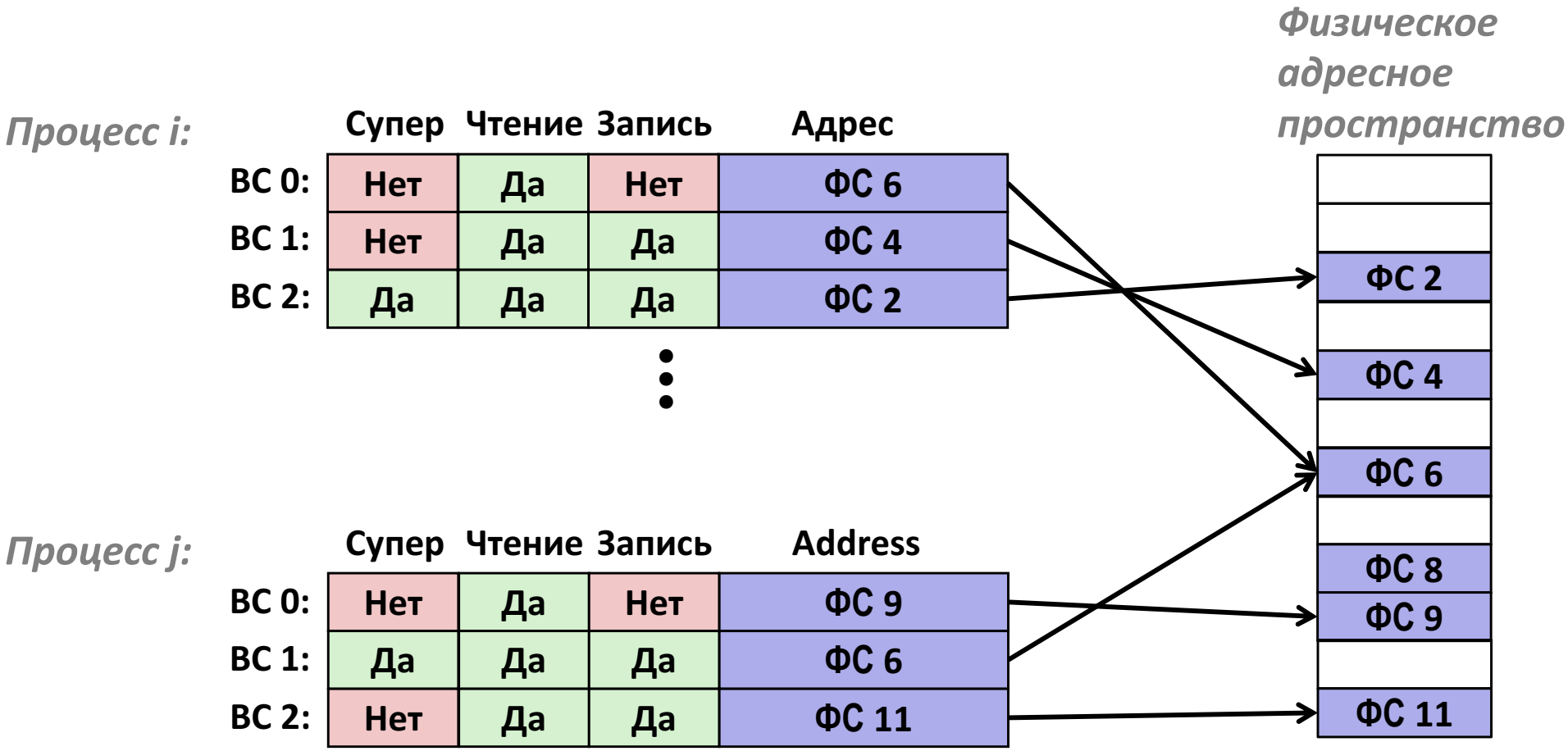


Виртуальная память – 1: понятия

- Пространства адресов
- ВП как средство кеширования
- ВП как средство управления памятью
- ВП как средство защиты памяти
- Трансляция адресов

ВП как средство защиты памяти

- **ЗТС дополняются битами-признаками разрешений**
- **Обработчик сбоев страниц проверяет перед отображением**
  - При нарушении, отправляет процессу SIGSEGV (segmentation fault)



Виртуальная память – 1: понятия

- Пространства адресов
- ВП как средство кеширования
- ВП как средство управления памятью
- ВП как средство защиты памяти
- Трансляция адресов

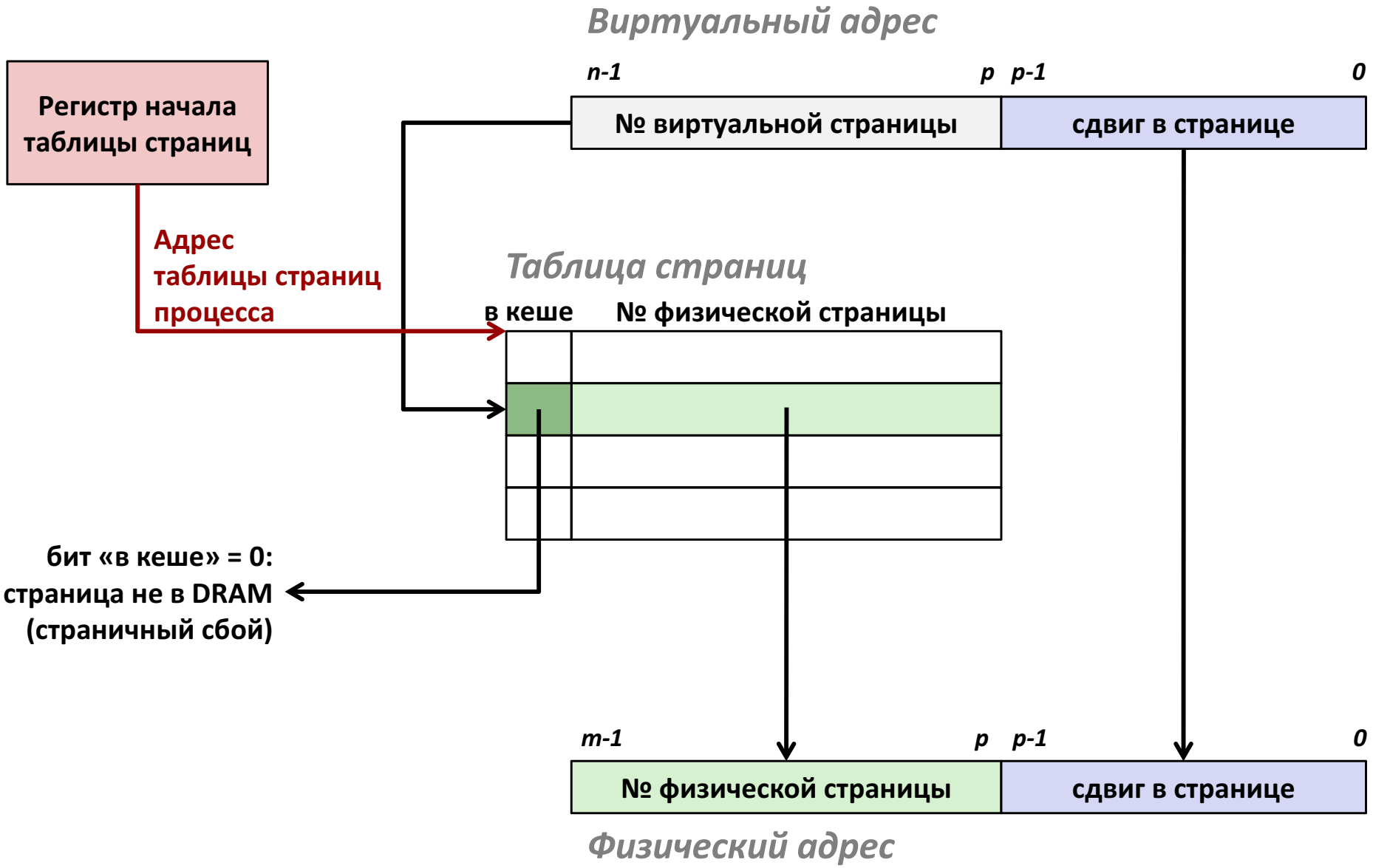
Трансляция адресов ВП

- **Виртуальное адресное пространство**
  - *V = {0, 1, …, N–1}*
- **Физическое адресное пространство**
  - *P = {0, 1, …, M–1}*
- **Трансляция адресов**
  - ***MAP: V → P U {∅}***
  - Для виртуального адреса ***a***:
    - ***MAP(a) = a'*** если данные по виртуальному адресу ***a*** находятся по физическому адресу ***a'*** в ***P***
    - ***MAP(a) = ∅*** если данные по виртуальному адресу ***a*** отсутствуют в физической памяти – или их нет совсем, или они на диске

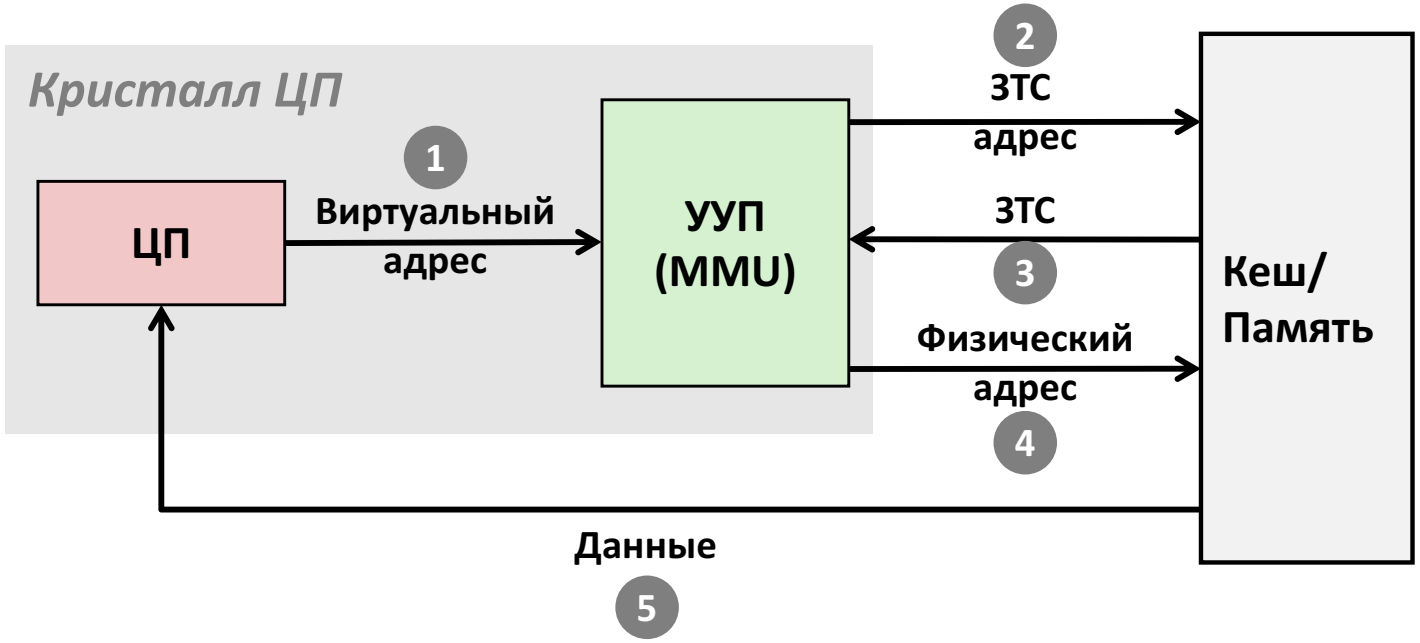
Сводка обозначений трансляции символов

- **Основные параметры**
  - **N = 2^n** : размер виртуального адресного пространства
  - **M = 2^m** : Размер физического адресного пространства
  - **P = 2^p** : Размер страницы (в байтах)
- **Части виртуального адреса**
  - **TLBI**: TLB index – индекс ближайшего буфера трансляции
  - **TLBT**: TLB tag – метка ближайшего буфера трансляции
  - **VPO**: Virtual page offset – сдвиг в виртуальной странице – СВС
  - **VPN**: Virtual page number – номер виртуальной страницы – НВС
- **Части физического адреса**
  - **PPO**: Physical page offset – сдвиг в физической странице – СФС
  - **PPN:** Physical page number – номер физической страницы – НФС

Трансляция адресов с таблицей страниц



Трансляция адресов: страничное попадание



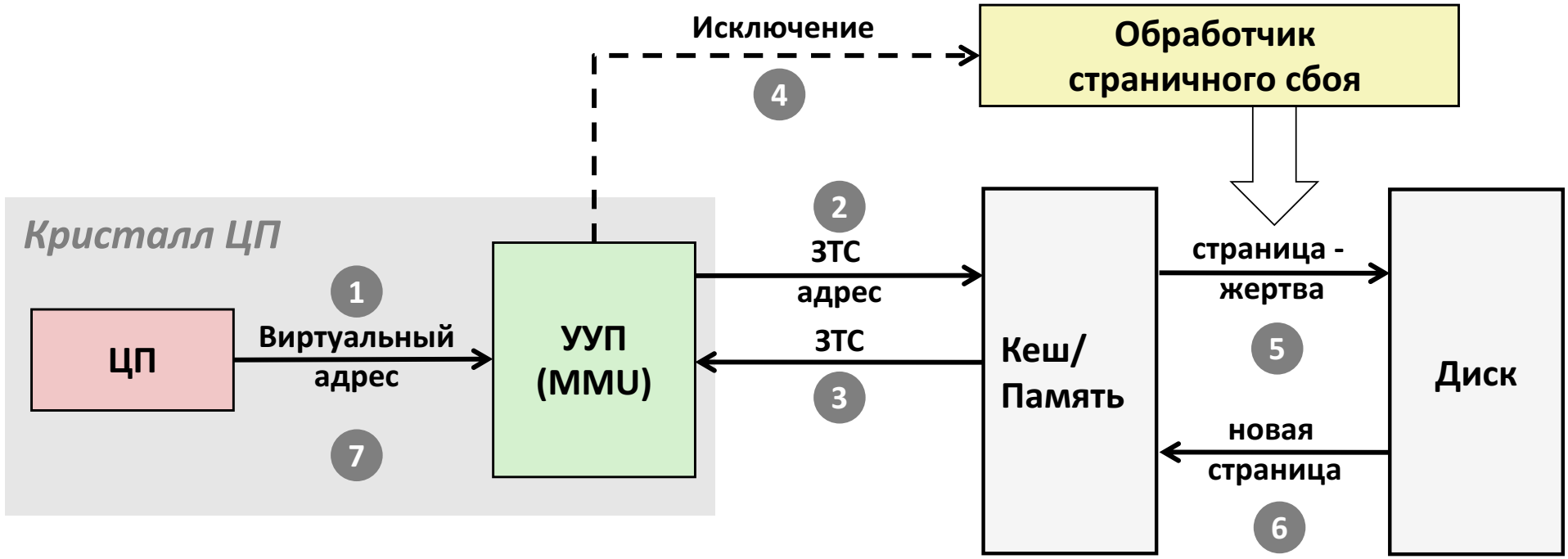
1) Процессор направляет виртуальный адрес УУП

2-3) УУП считывает ЗТС из таблицы страниц в памяти

4) УУП направляет физический адрес кешу/памяти

5) Кеш/память направляют данные процессору

Трансляция адресов: страничный сбой



1) Процессор направляет виртуальный адрес УУП

2-3) УУП считывает ЗТС из таблицы страниц в памяти

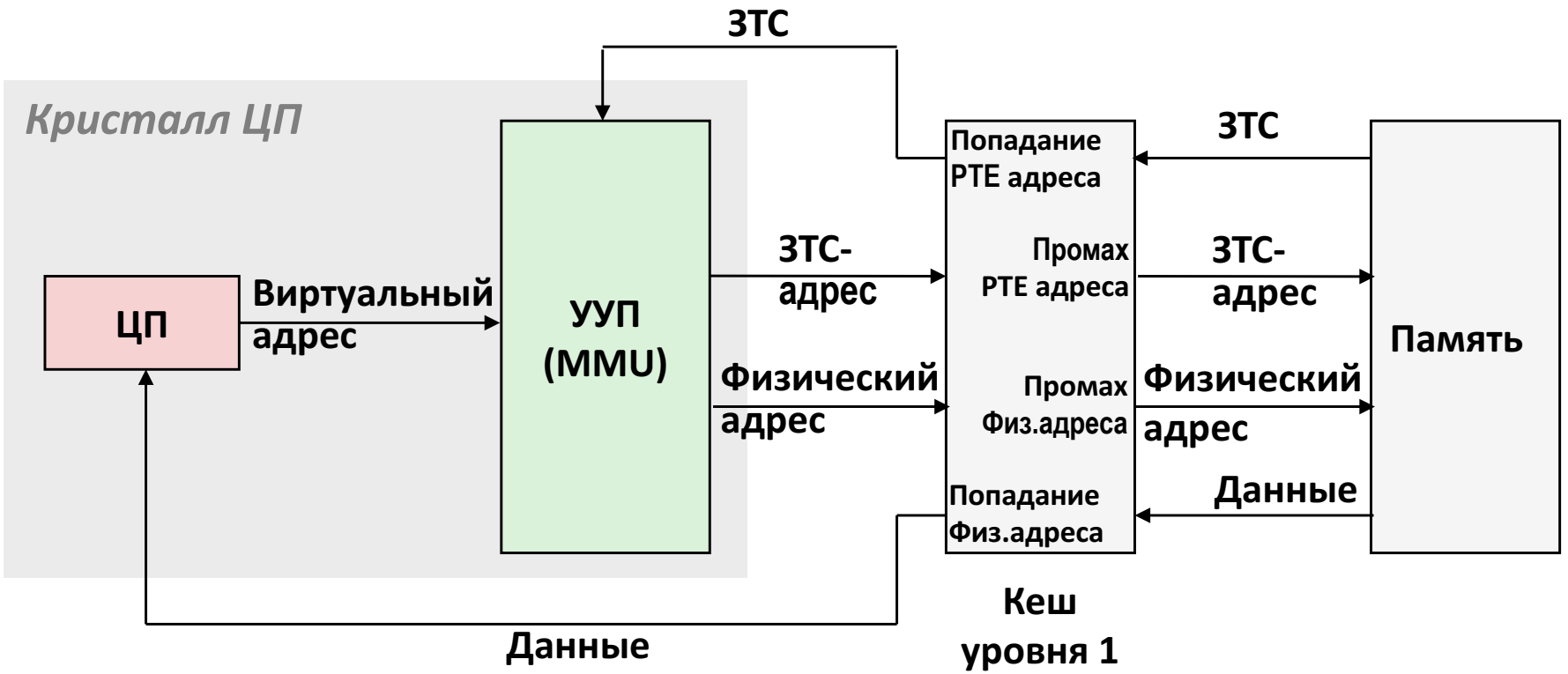
4) Бит «в кеше» - нулевой, УУП запускает исключение «страничный сбой»

5) Обработчик выбирает жертву (и, если менялась, откачивает на диск)

6) Обработчик подкачивает новую страницу и меняет ЗТС в памяти

7) Обработчик возвращается в процесс для повтора сбойной команды

Объединение ВП и кеша основной памяти



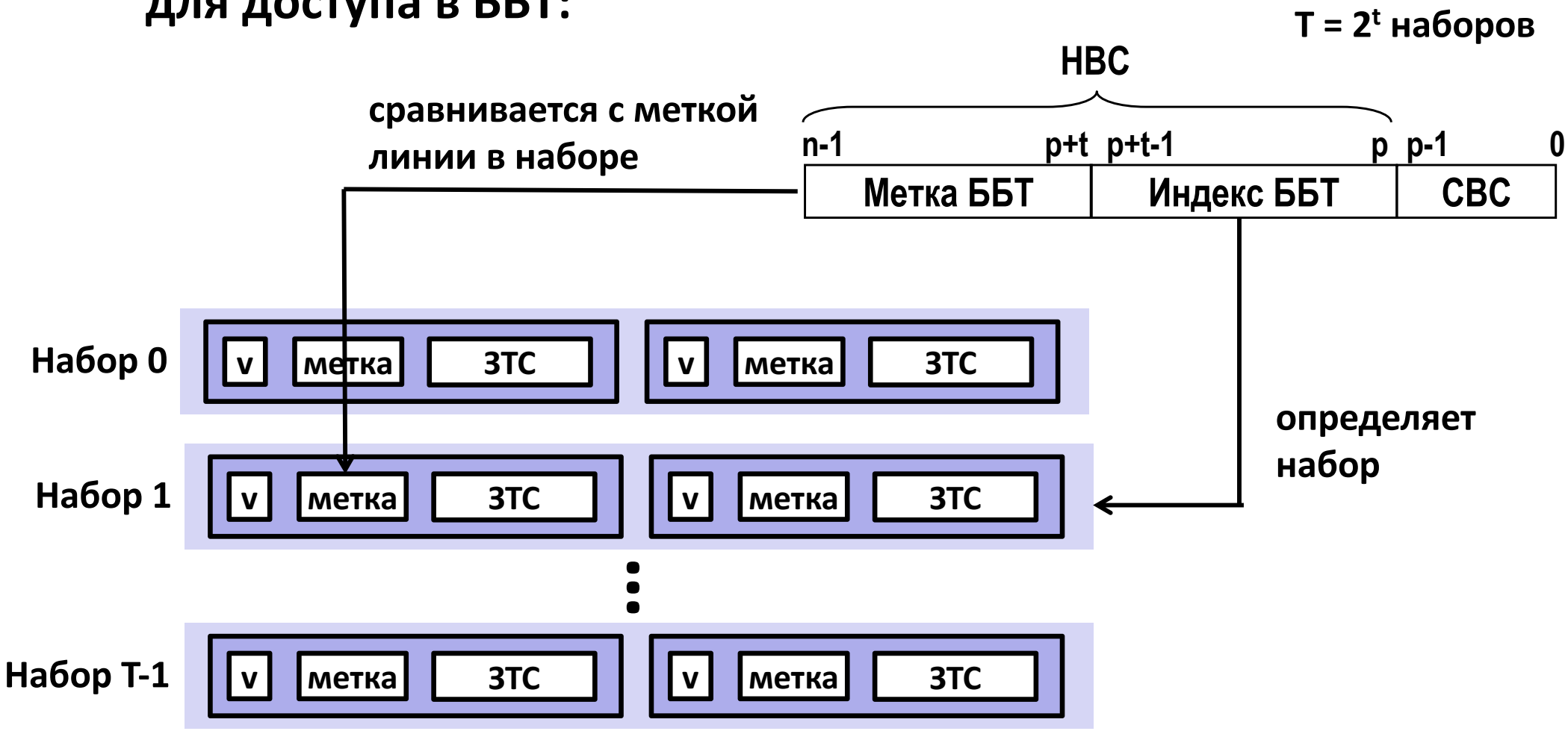
ЗТС: запись в таблице страниц

Ускорение трансляции с помощью TLB

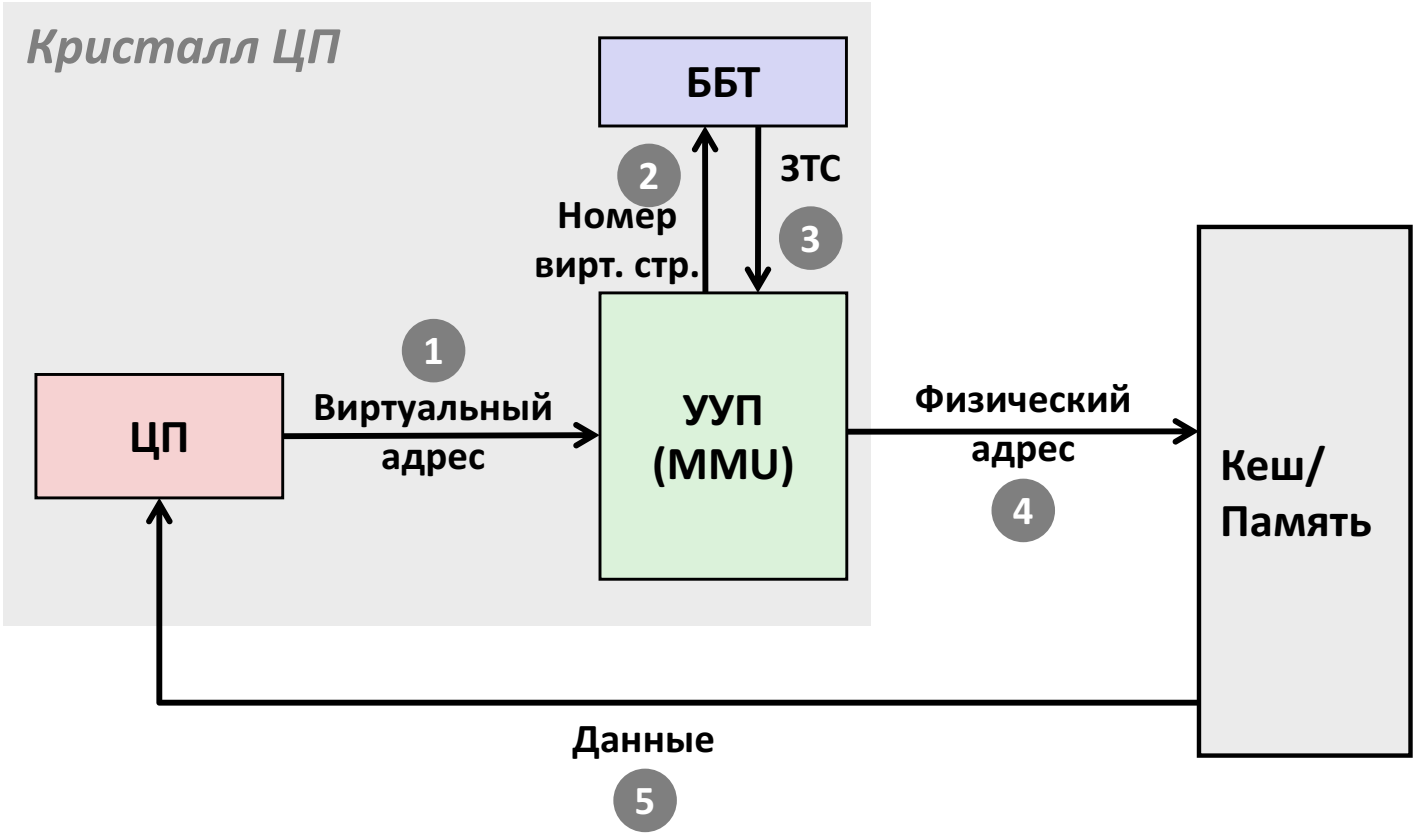
- **Записи в таблице страниц (ЗТС) кешируются в уровне 1 как любое другое слово памяти**
  - ЗТС могут выталкиваться обращениями к другим данным
  - попадание ЗТС всё ещё требует небольшой задержки уровня 1
- **Решение:** ***Ближайший буфер трансляции (ББТ, Translation Lookaside Buffer - TLB)***
  - небольшой аппаратный кеш в УУП (MMU)
  - отображает номера виртуальных страниц в номера физических
  - содержит целиком записи для небольшого количества страниц

Доступ к ББТ

- **УУП использует часть НВС часть виртуального адереса для доступа в ББТ:**

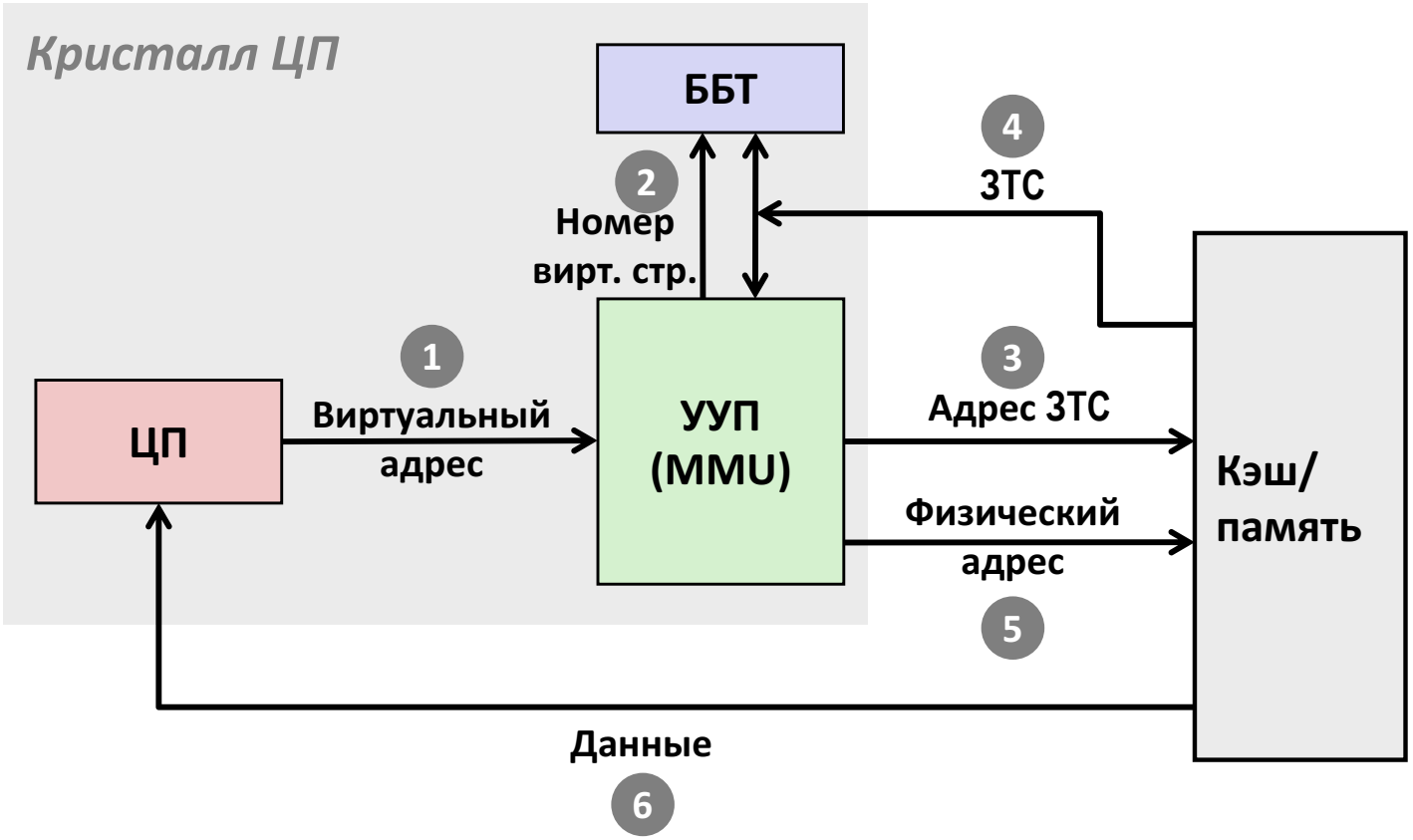


Попадание ББТ



Попадание TLB - нет дополнительного обращения в память

Промах ББТ

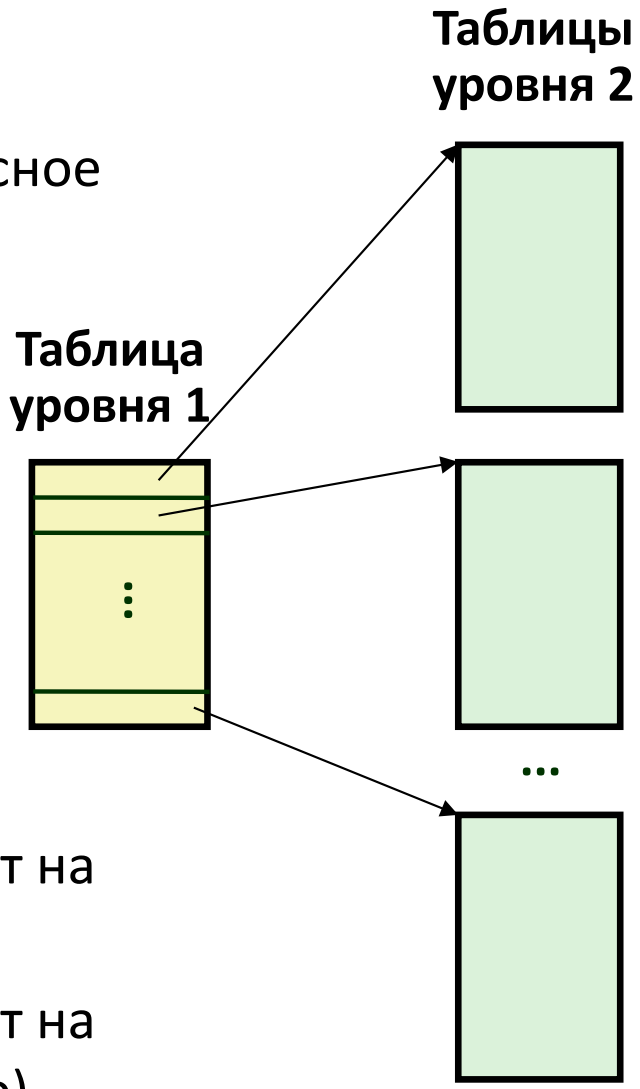


Промах TLB влечёт дополнительное обращение в память за ЗТС

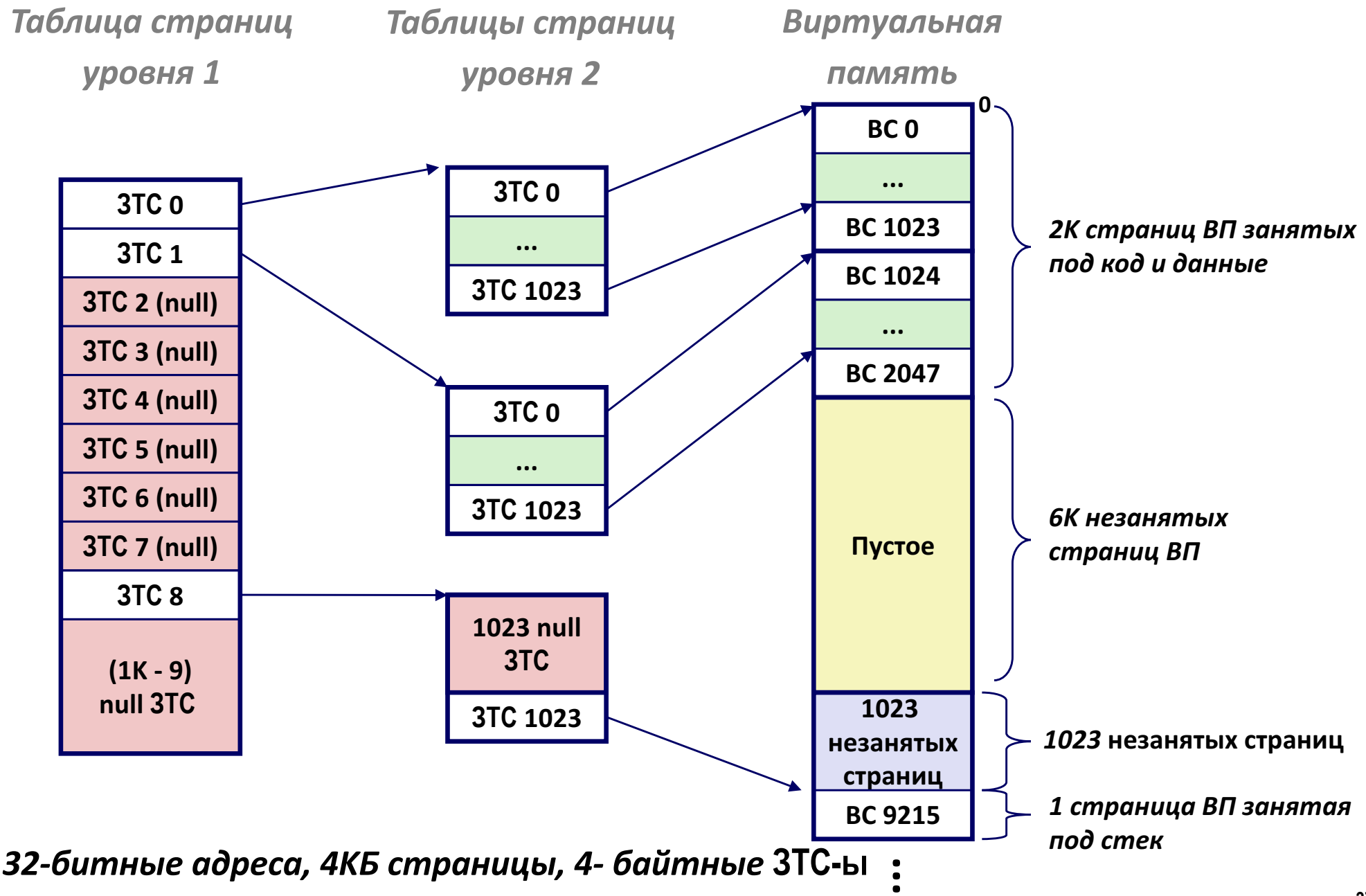
К счастью, промахи TLB редки. Почему?

Многоуровневые таблицы страниц

- **Предположим:**
  - 4KB (2^{12}) размер страницы, 48-битное адресное пространство, 8-байтные ЗТС
- **Проблема:**
  - Необходима таблица размером 512 ГБ!
    - $2^{48} * 2^{-12} * 2^{3} = 2^{39}$ байт
- **Типовое решение:**
  - Многоуровневые таблицы страниц
  - Пример: 2-уровневая таблица страниц
    - Таблица уровня 1: каждая ЗТС указывает на таблицу страниц (всегда в памяти)
    - Таблица уровня 2: каждая ЗТС указывает на страницу (подкачанную или откачанную)

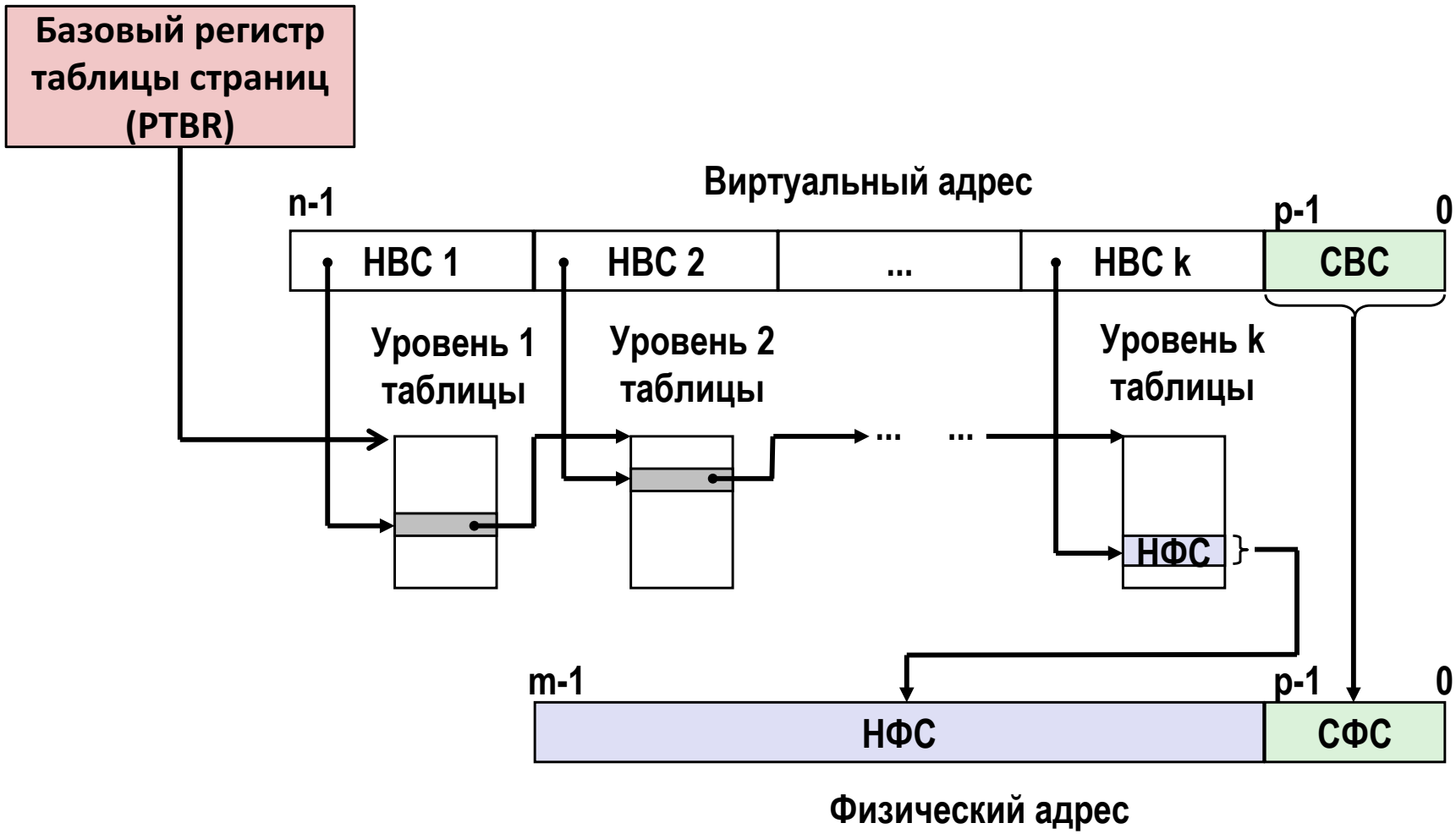


Двухуровневая иерархия таблиц страниц



32-битные адреса, 4КБ страницы, 4- байтные ЗТС-ы

Трансляция с k-уровневой таблицей страниц



Сводка

- **С точки зрения программиста, виртуальная память…**
  - даёт каждому процессу собственное линейное адресное пространство
  - не может быть повреждена другим процессом

- **С точки зрения системы, виртуальная память…**
  - эффективно использует DRAM для кеширования виртуальных страниц
    - только благодаря локальности
  - упрощает управление памятью и программирование
  - упрощает защиту памяти обеспечивая удобную контрольную точку проверки разрешений