УНИВЕРСИТЕТ ЛОБАЧЕВСКОГО

itseez

Иерархия памяти компьютера

Сергей Шишлов

3 февраля 2016 г.

План лекции

- Как сделать память быстрой и вместительной?
  - Локальность
  - Иерархическая организация
- Типичный алгоритм обращения в память
  - Виртуальная память
    - Линейный (виртуальный) адрес
    - Организация виртуальной памяти
    - Трансляция виртуального адреса и TLB
  - Кэш
    - Последовательность действий
    - Способы отображения данных в кэше
    - Действия при записи в кэш
    - Причины промахов
    - Проблема когерентности
- Влияние иерархической организации на производительность
    - Способы сокращения времени доступа и частоты промахов в кэш

Локальность обращений к память

- Чем быстрее память, тем она менее вместительна и более дорога
  - Регистры: доли наносекунды, сотни байт
  - SRAM: десятки наносекунд, мегабайты
  - DRAM: сотни наносекунд, гигабайты
  - Магнитный диск: миллисекунды, терабайты
- Как реализовать быструю и вместительную память?
- Нужно использовать свойство локальности

Временная локальность: если было обращение по некоторому адресу, в ближайшее время оно может повториться

Пространственная локальность: если было обращение по некоторому адресу, в ближайшее время возможны обращения по соседним адресам

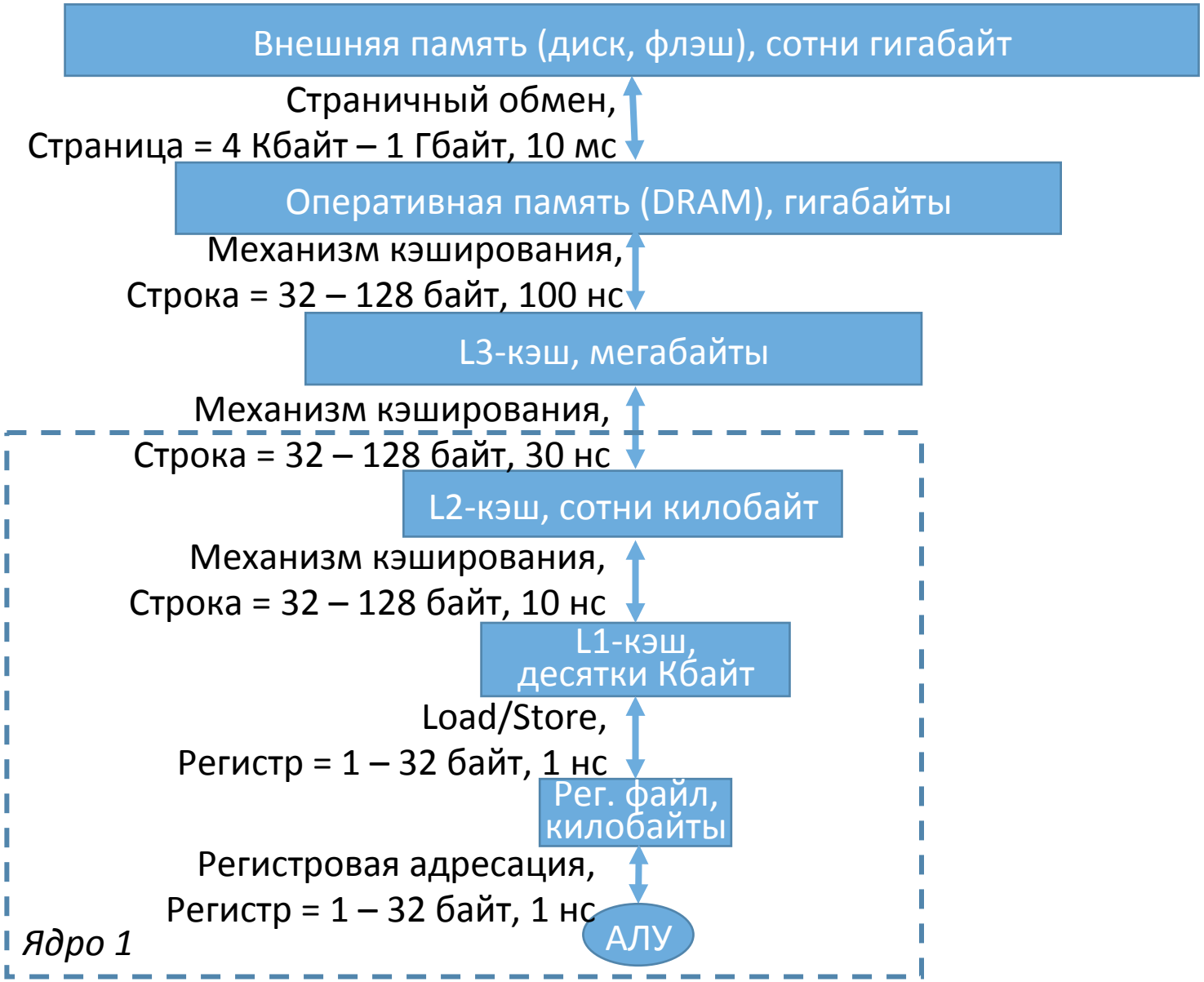
Локальность на разных уровнях реализации

- Регистры / память – традиционная эксплуатация локальности на уровне системы команд (компилятором)
- Программа не знает об объеме доступной ей физической памяти
  - Нужна эксплуатация локальности на уровне операционной системы: виртуальная память
- Локальность проявляет себя динамически
  - Нужна эксплуатация локальности в аппаратуре: кэш

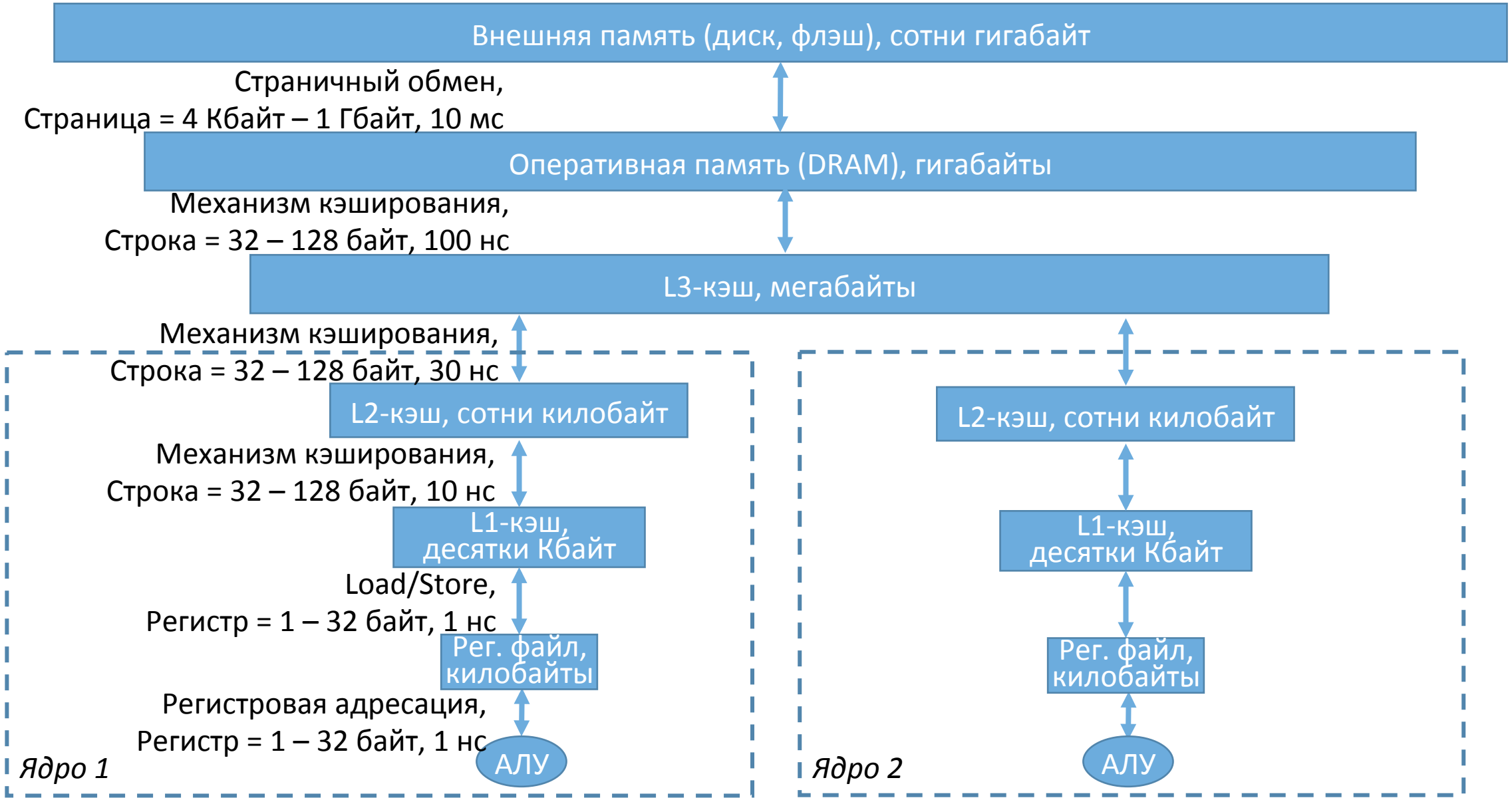
Иерархия памяти

- Локальность имеет иерархическую структуру
- Стек уровней иерархии памяти
  - С каждым следующим уровнем:
    - Больше объем
    - Больше задержка
    - Меньше пропускная способность
- При отсутствии данных на текущем уровне запрос переправляется на следующий уровень
- При подкачке данных они оседают на более близких уровнях для последующего использования (временная локальность)
- Вместе с запрошенными данными передаются и соседние (строка кэша, страница виртуальной памяти – пространственная локальность)

Иерархия памяти



Иерархия памяти



Алгоритм исполнения обращения в память. Часть 1: Вычисление линейного адреса

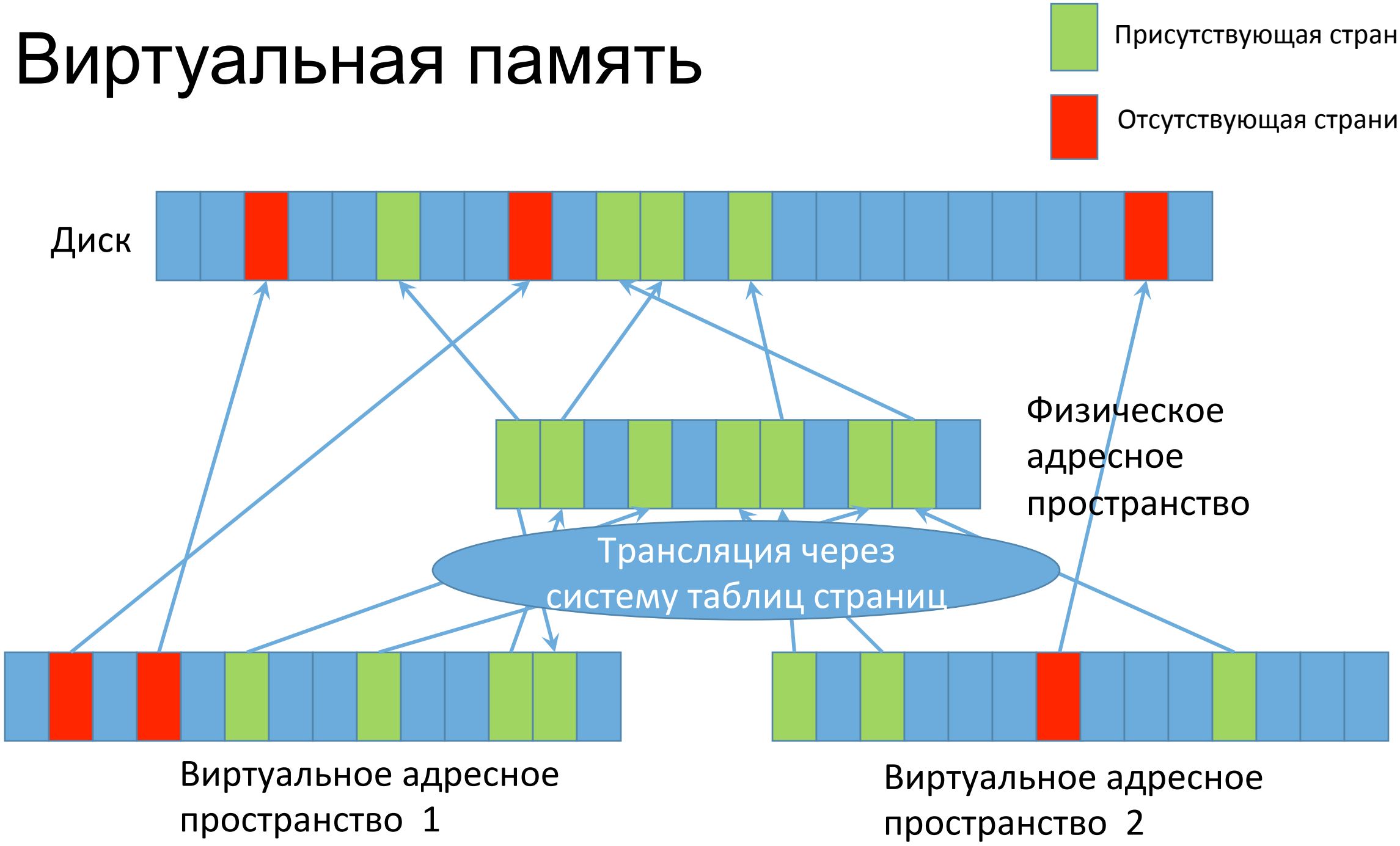
MOV RAX, [RBX+RSI]

1. Вычисляется эффективный адрес: EA = RBX + RSI
2. Вычисляется линейный (виртуальный) адрес: LA = EA + seg_base
   - В 64-разрядном режиме используется плоское линейное пространство, т.е. seg_base=0

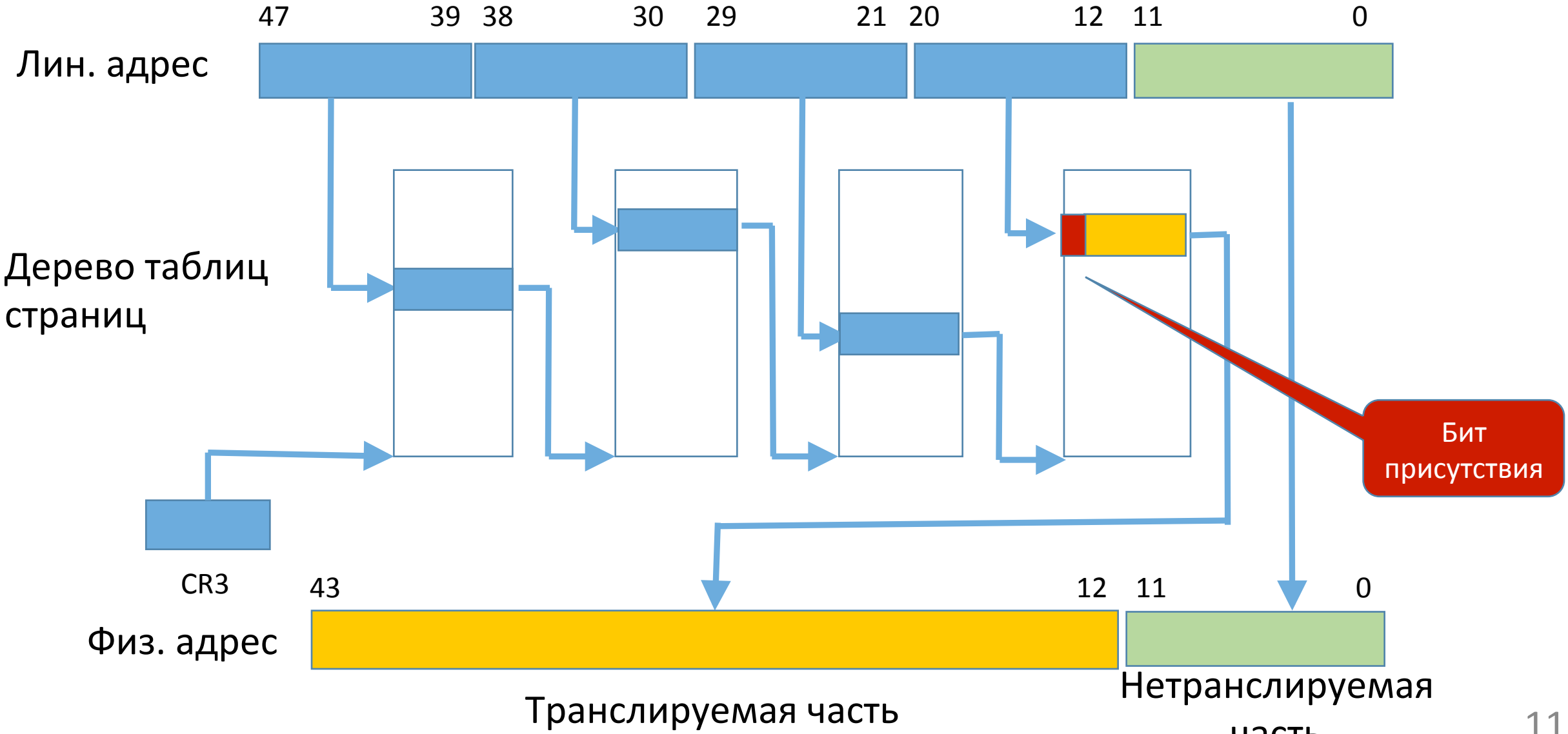
Виртуальная память

- Позволяет программе обращаться к большому адресному пространству, не зная об объеме физической оперативной памяти
- Позволяет нескольким программам одновременно содержать данные в памяти, не зная друг о друге
- Самый далекий уровень иерархии
- Требует трансляции адреса

Виртуальная память



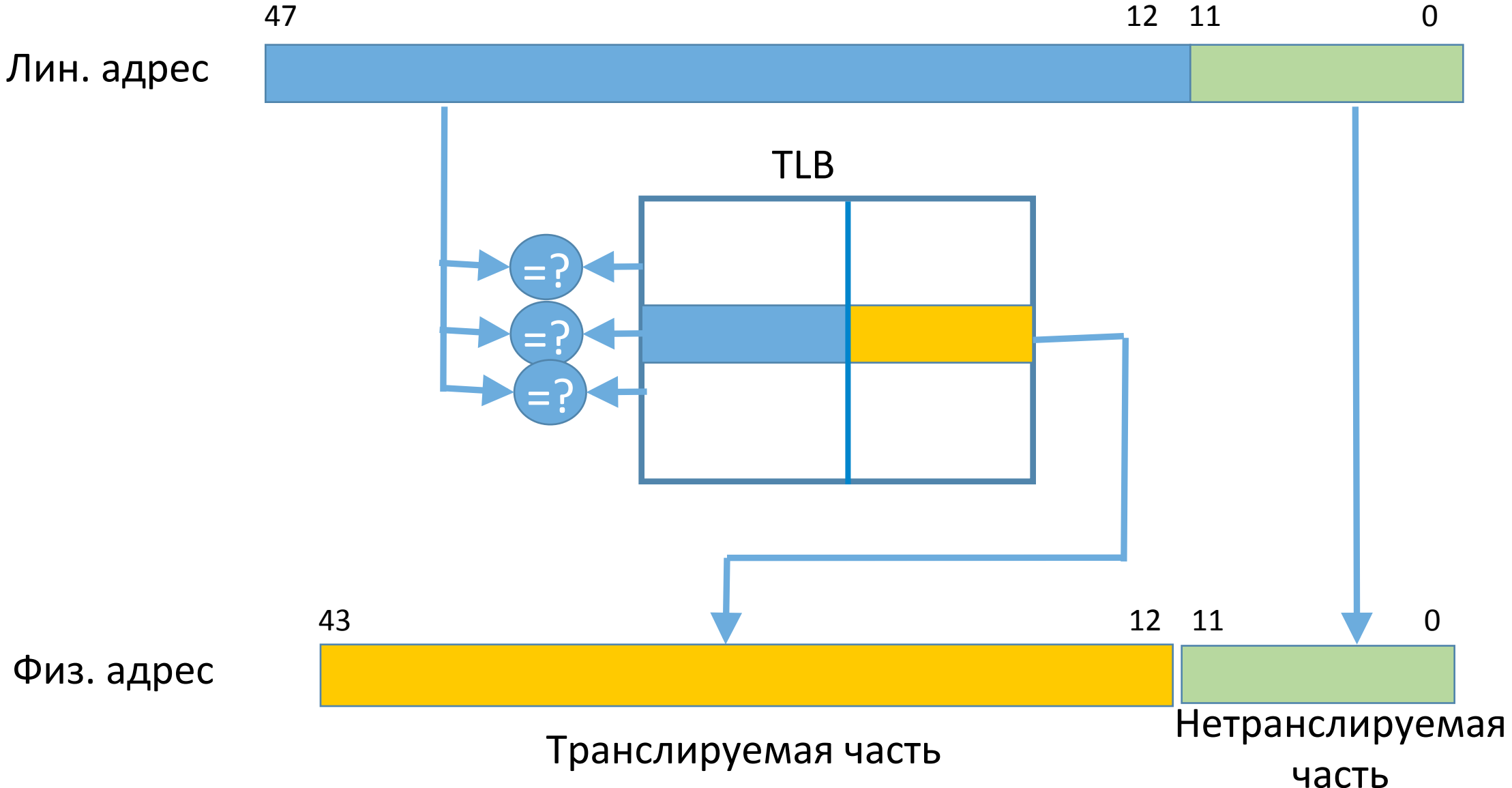
Иерархическая структура таблиц страниц в архитектуре Intel 64



Буфер трансляции адресов (TLB)

- Проход по таблицам = 5 обращений в память вместо одного
- Можно эксплуатировать локальность на уровне страниц
  - Хранить наиболее вероятные трансляции аппаратно
- Фактически еще один промежуточный уровень иерархии
- Может быть тоже реализован иерархически
- Имеет структуру, аналогичную кэшу (на следующих слайдах)

Ускорение трансляции с помощью буфера трансляции адресов (TLB)



Алгоритм исполнения обращения в память. Часть 2: Трансляция линейного адреса в физический

3. Выполняется обращение в TLB старшими битами линейного адреса
4. Если TLB содержит трансляцию для данной страницы, то физический адрес страницы читается из TLB
5. Если же в TLB трансляция отсутствует, выполняется проход по таблицам страниц
   - Если проход завершается успешно, то физический адрес страницы, считанный из таблицы страниц, подгружается в TLB и происходит повторное обращение в TLB (при этом из TLB может выгрузиться одна из ранее загруженных трансляций)
   - Если же выясняется, что страницы нет в памяти, возникает прерывание, и операционная система подгружает страницу с диска, обновляет таблицу страниц и возвращает управление на исходную команду

Кэш

- Все физическое адресное пространство разделено на блоки (строки) фиксированного размера
- Кэш хранит некоторое подмножество строк в быстрой памяти (SRAM)
- Кэш может обслуживать обращения к имеющимся (попадание – hit) или сообщать об отсутствии строки (промах – miss)
- Для нахождения строк кэш хранит вместе с каждой строкой ее адрес (тэг)

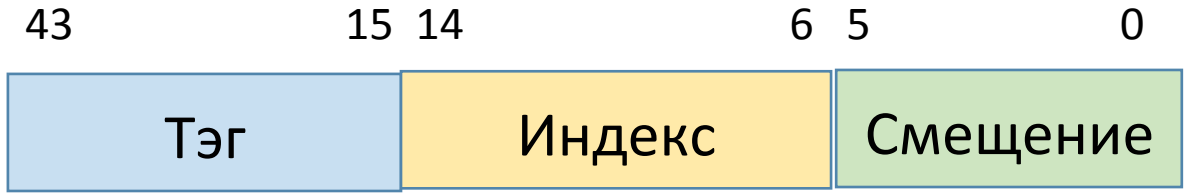
Алгоритм исполнения обращения в память

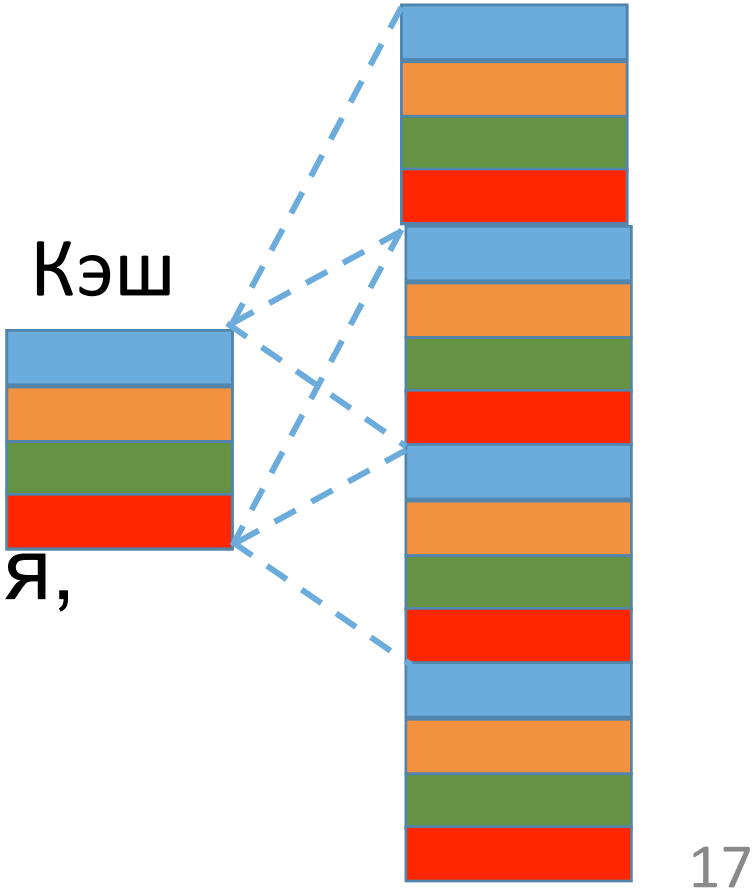
Часть 3: Обращение в кэш

6. Происходит обращение в кэш 1-го уровня (L1) по физическому адресу
7. Если данные в кэше присутствуют (попадание, hit), они читаются из кэша
8. В противном случае (промах, miss) запрос передается в кэш 2-го уровня и так далее.
9. Если данных не находится ни в одном кэше, они читаются из оперативной памяти
10. В случае чтения данных из памяти или из далеких кэшей, строка, содержащая данные, прописывается во все близлежащие кэши (это может привести к вытеснению какой-то другой строки)

Как разместить (и найти) данные в кэше?

Вариант 1: Прямое отображение:
каждой строке адресного пространства соответствует только один блок в кэше

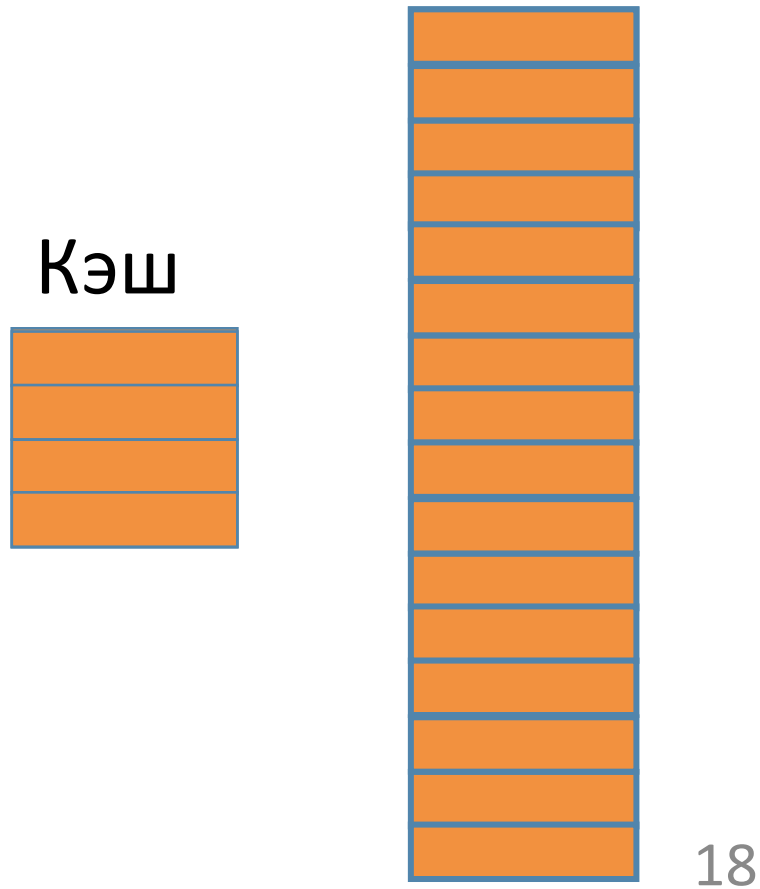




Номера битов проставлены для случая,
когда размер строки = 64 байта,
размер кэша = 32 Кбайт
(т.е. емкость кэша = 512 строк)

Как разместить (и найти) данные в кэше?

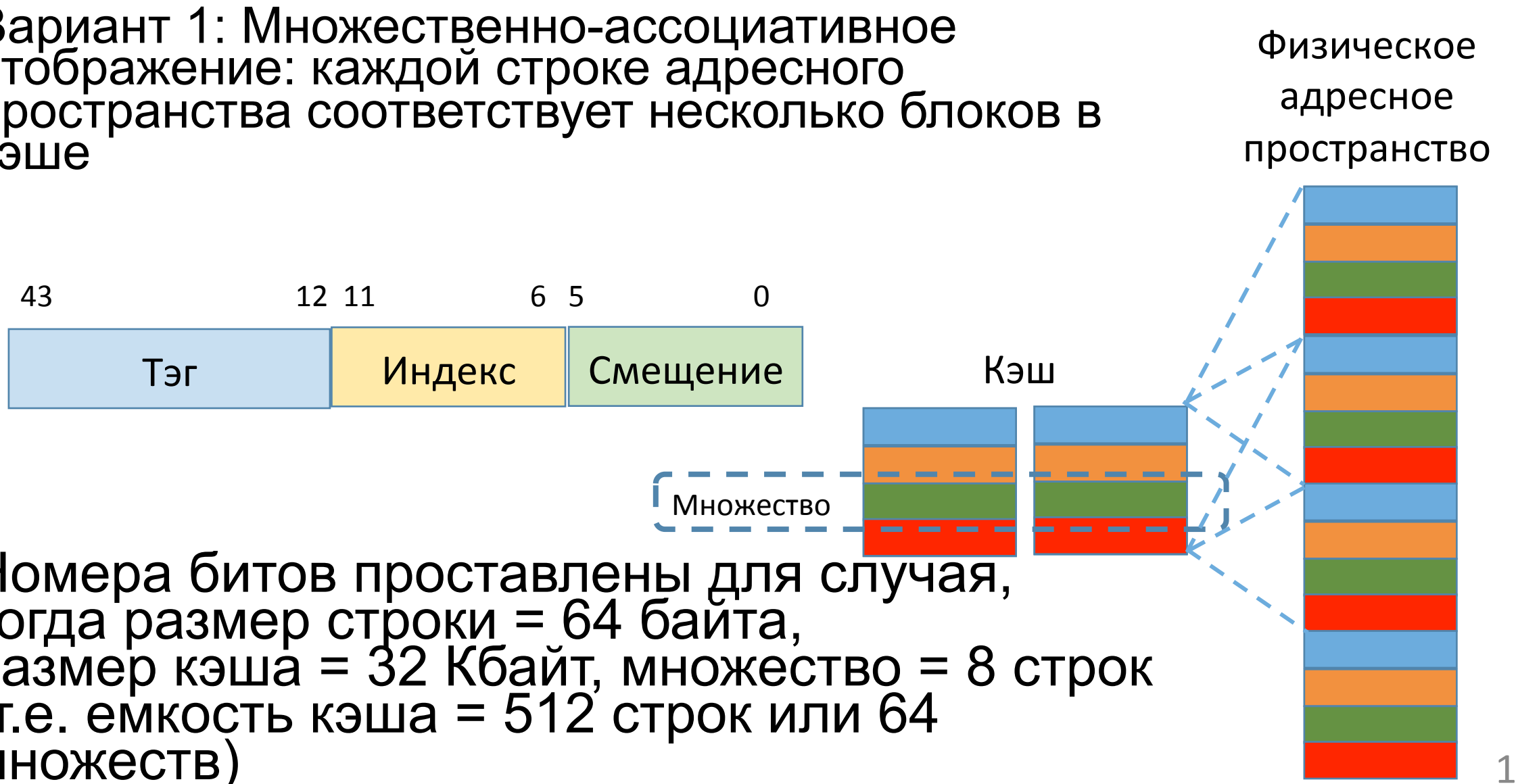
Вариант 2: Полностью ассоциативное отображение: любая строка адресного пространства может находиться в любом блоке в кэше



Номера битов проставлены для случая, когда размер строки = 64 байта

Как разместить (и найти) данные в кэше?

Вариант 1: Множественно-ассоциативное отображение: каждой строке адресного пространства соответствует несколько блоков в кэше



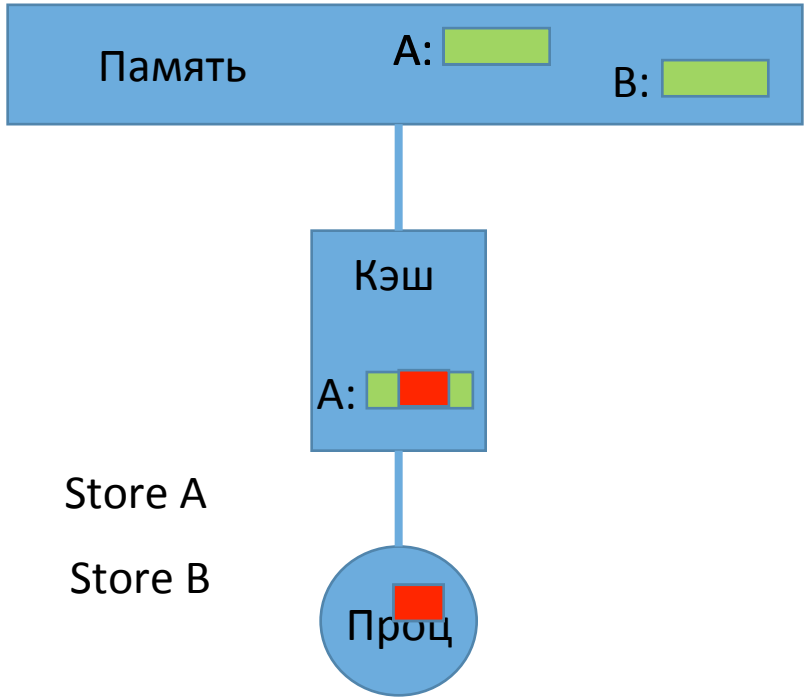
Номера битов проставлены для случая, когда размер строки = 64 байта, размер кэша = 32 Кбайт, множество = 8 строк (т.е. емкость кэша = 512 строк или 64 множеств)

Как выбрать строку для вытеснения?

- Неактуально для кэша с прямым отображением
- Для кэша с ассоциативным отображением
- Вариант 1: случайно
- Вариант 2: FIFO: наиболее долго находящуюся в кэше
- Вариант 3: LRU (least recently used): наиболее долго неиспользуемую

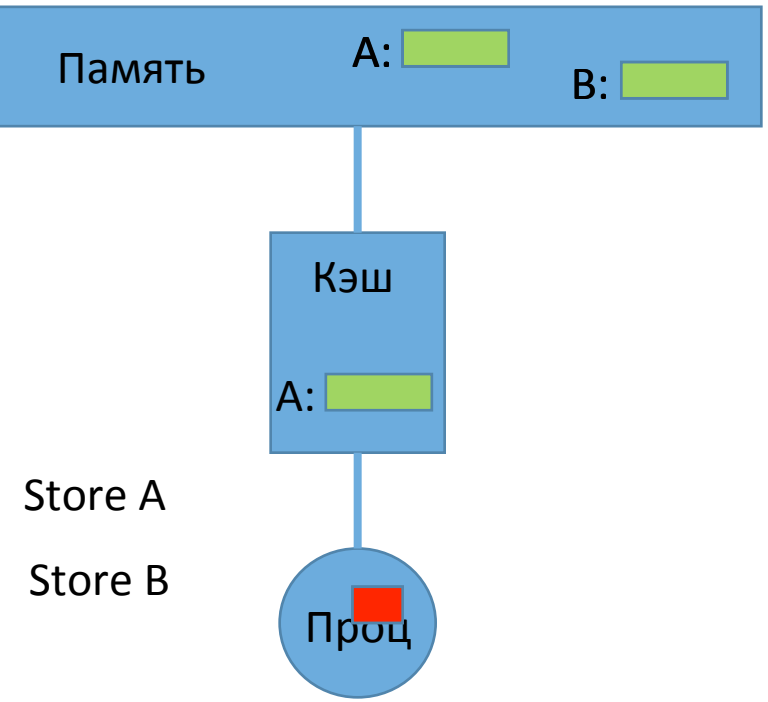
Что происходит с кэшем при записи в память?

- Вариант 1: Сквозная запись без размещения

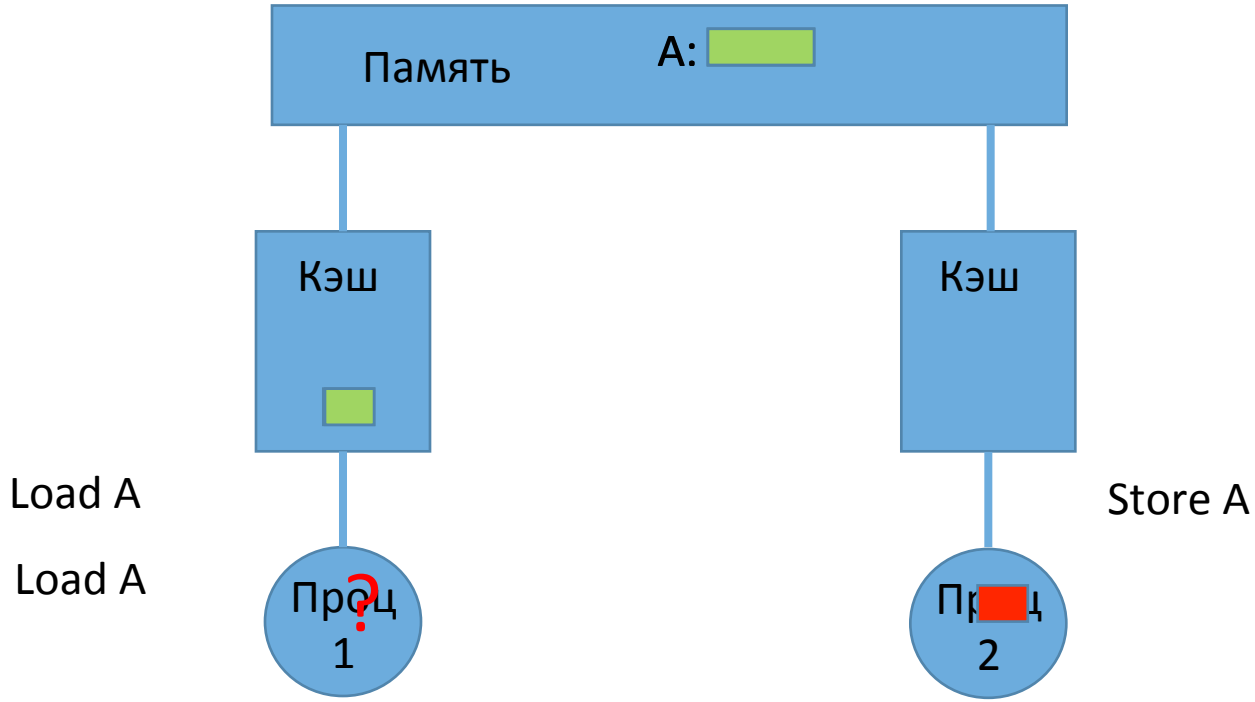


Что происходит с кэшем при записи в память?

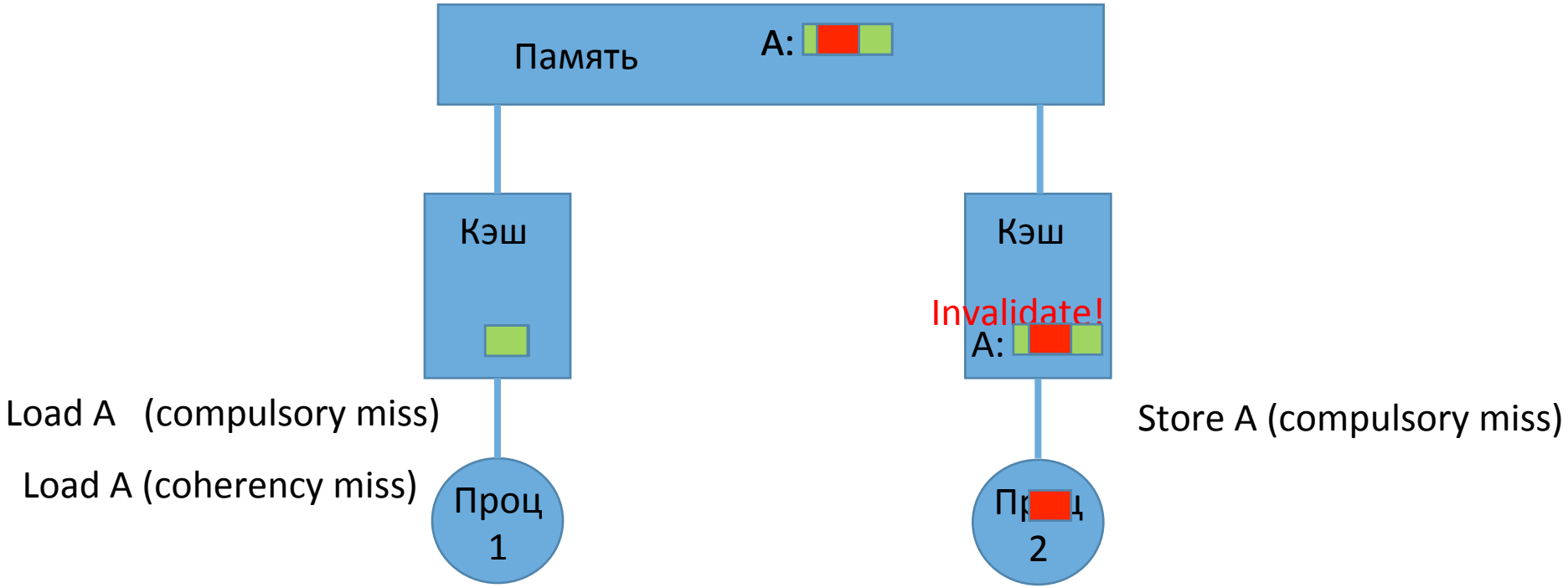
- Вариант 2: Обратная запись с размещением



Проблема когерентности



Проблема когерентности: решение



Протокол когерентности

- Каждая строка в кэше хранит свое состояние:
  - Invalid – в данном блоке кэша нет никакой строки
  - Exclusive – никакой другой кэш не содержит данной строки, и в памяти имеется действительная копия
  - Modified – строка обновлена, копия в памяти недействительна
  - Shared – строка может присутствовать и в других кэшах, в памяти имеется действительная копия
  - Возможны и другие состояния
- Протокол обеспечивает правильность обновления состояний и выполнение необходимых действий по обнулению и вытеснению строк

Инклюзивность/эксклюзивность

- Инклюзивность: один уровень иерархии полностью включает в себя другие
- Эксклюзивность: данные располагаются только на каком-то одном уровне иерархии
- Возможны прочие варианты

Иерархия памяти и производительность

- Основное уравнение производительности:

$$P = \frac{1}{N * Tclk * CPI} = \frac{1}{N * (Tclk * CPIcpu + AMAT)}$$

- Среднее время доступа в память:

AMAT = Mtlb * Tpage_walk + Hl1 * Tl1 + Hl2 * Tl2 + Hl3 * Tl3 + Hmem * Tmem

Частота промахов в TLB

Среднее время прохода по таблицам

Частота попаданий в L1-кэш

Время доступа в L1-кэш

Частота обращений в основную память

Среднее время доступа к основной памяти

Как сократить время доступа?

- Параллельное обращение в TLB и L1-кэш (индекс не должен транслироваться)
- Уменьшение ассоциативности и размера кэша
- Предсказание канала ассоциативного кэша
  - В случае удачи - задержка как у кэша с прямым отображением
- Наложение задержек разных команд друг на друга
  - Неблокирующий кэш
  - Расслоение
  - Конвейер

По каким причинам может случиться промах?

- 4C:
  - ***C**ompulsory* (***C**old*): при первом обращении к данной строке в программе
  - ***C**apacity*: если размер рабочего множества превышает размер кэша
  - ***C**onflict* (***C**ollision*): если рабочее множество включает в себя больше строк, относящихся к одному множеству, чем размер множества
  - ***C**oherency*: если данные были вытравлены из кэша другим процессом (см. следующий слайд)

Как сократить количество промахов?

- Повышение ассоциативности и размера кэша
- Увеличение размера строки (не всегда)
- Предподкачка
- Компиляторные оптимизации

Спасибо за внимание!

Backup

Какую строку вытеснять?

- Для прямого отображения вопрос не актуален
- Для ассоциативного отображения:
  - Вариант 1: случайный выбор
  - Вариант 2: FIFO (та строка, которая была размещена в кэше раньше других)
  - Вариант 3: LRU (та строка, к которой дольше всего не было обращений)

Проблема когерентности