# Иерархия памятей

Основы информатики

**Компьютерные основы программирования**

**u.to/DbCmFA**

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

**u.to/XoKmFA**

Лекция 9, 07 апреля 2023

Лектор:

Дмитрий Северов, кафедра информатики 608 КПМ

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

# Иерархия памятей

- **Технологии и тренды**
- **Локальность обращения**
- **Кеширование в иерархии памятей**

# Память с произвольным доступом Random-Access Memory (RAM)

- **Ключевые особенности**
  - RAM обычно изготавливается в виде (части) интегральной схемы.
  - Базовая единица хранения - ячейка (1 бит в ячейке).
  - Несколько ИС RAM образуют память.
- **2 классических вида памяти с произвольным доступом**
  - Статическая RAM (SRAM)
  - Динамическая RAM (DRAM)

# Сводка по свойствам SRAM и DRAM

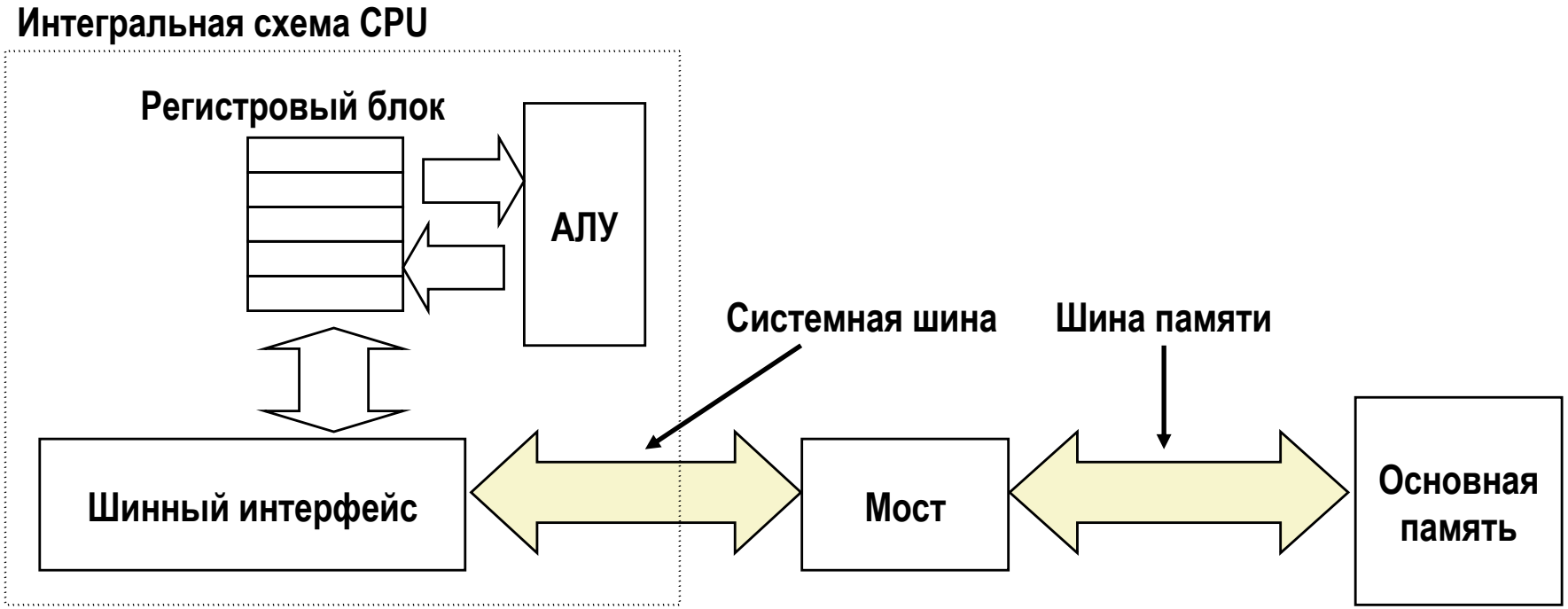
| | Транз. на бит | Время доступа | Регене-рация | Коррек-ция? | Стои-мость | Применения |
|---|---|---|---|---|---|---|
| SRAM | 4 или 6 | 1X | Нет | Не обяз. | 100x | Кэш-память |
| DRAM | 1 | 100X | Да | Да | 1X | Основная память, видео-память |

# Постоянная твердотельная память

- **DRAM и SRAM – непостоянная память**
  - Теряют информацию при выключении электропитания
- **Постоянная память сохраняет значение при выключении**
  - Read-only memory (ROM): программируется при изготовлении
  - Programmable ROM (PROM): однократно-программируемая
  - Eraseable PROM (EPROM): может стираться целиком (УФ, рентген)
  - Electrically eraseable PROM (EEPROM): электрически стираемая
  - Flash memory: EEPROM с частичным (секторным) стиранием
    - Выходит из строя после 0.1-1M стираний.
- **Применение постоянной памяти**
  - «Встроенные» программы, хранимые в *ROM (BIOS, модули/платы расширения, подсистемы безопасности…)
  - Твердотельные диски Solid state disks (замена вращающихся дисков в «брелках», смартфонах, плеерах, планшетах, ноутбуках…)
  - Дисковые кеши

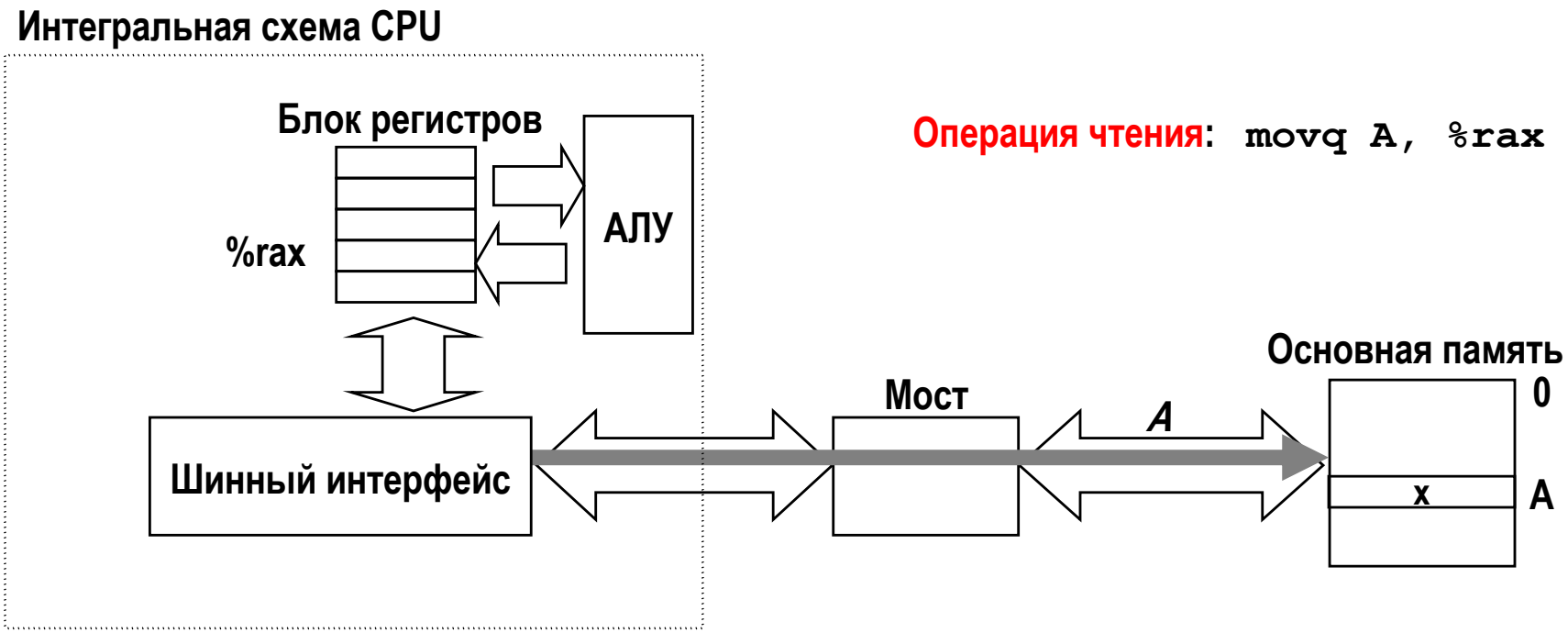
# Традиционное подключение ЦП к ОЗУ

- **Шина набор параллельных проводников, несущих сигналы адреса, данных и управления.**
- **Обычно к шине подключаются несколько устройств**



# Чтение из памяти - 1

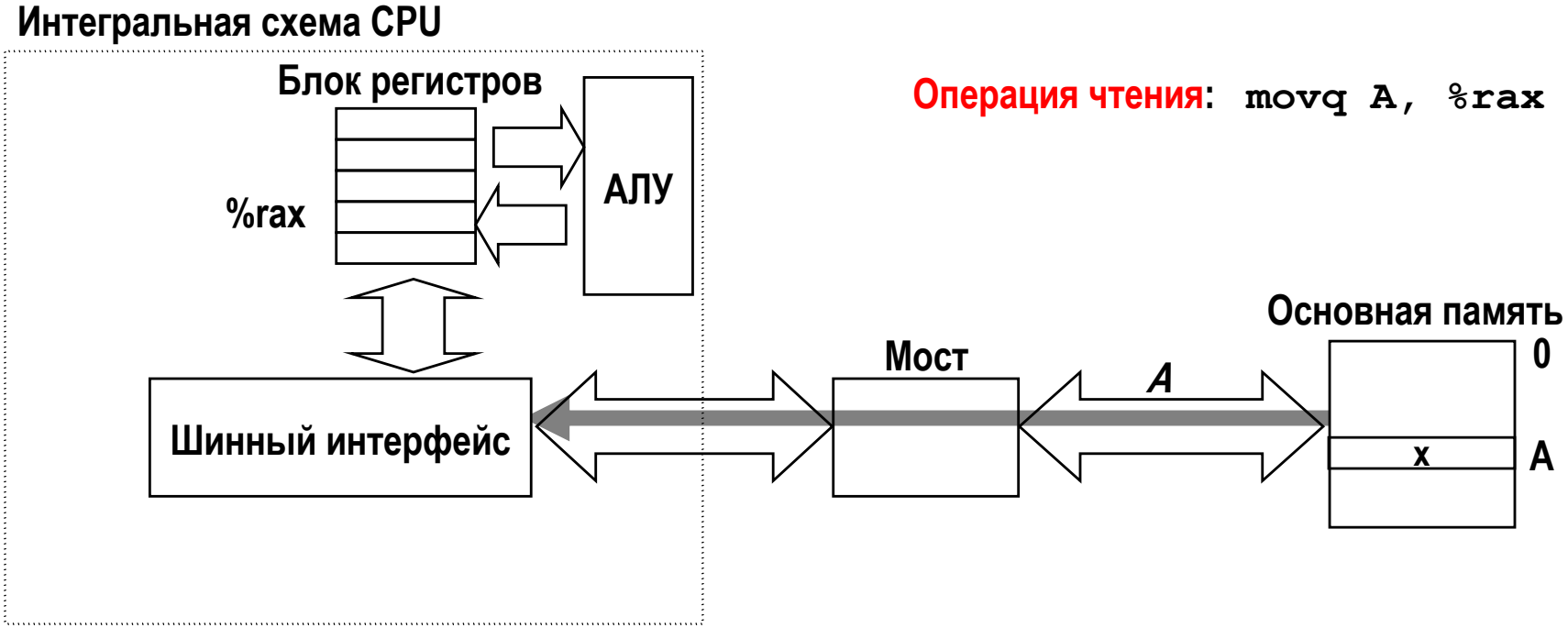
- ЦП выставляет адрес A на шине памяти




Операция чтения: `movq A, %rax`

# Чтение из памяти - 2

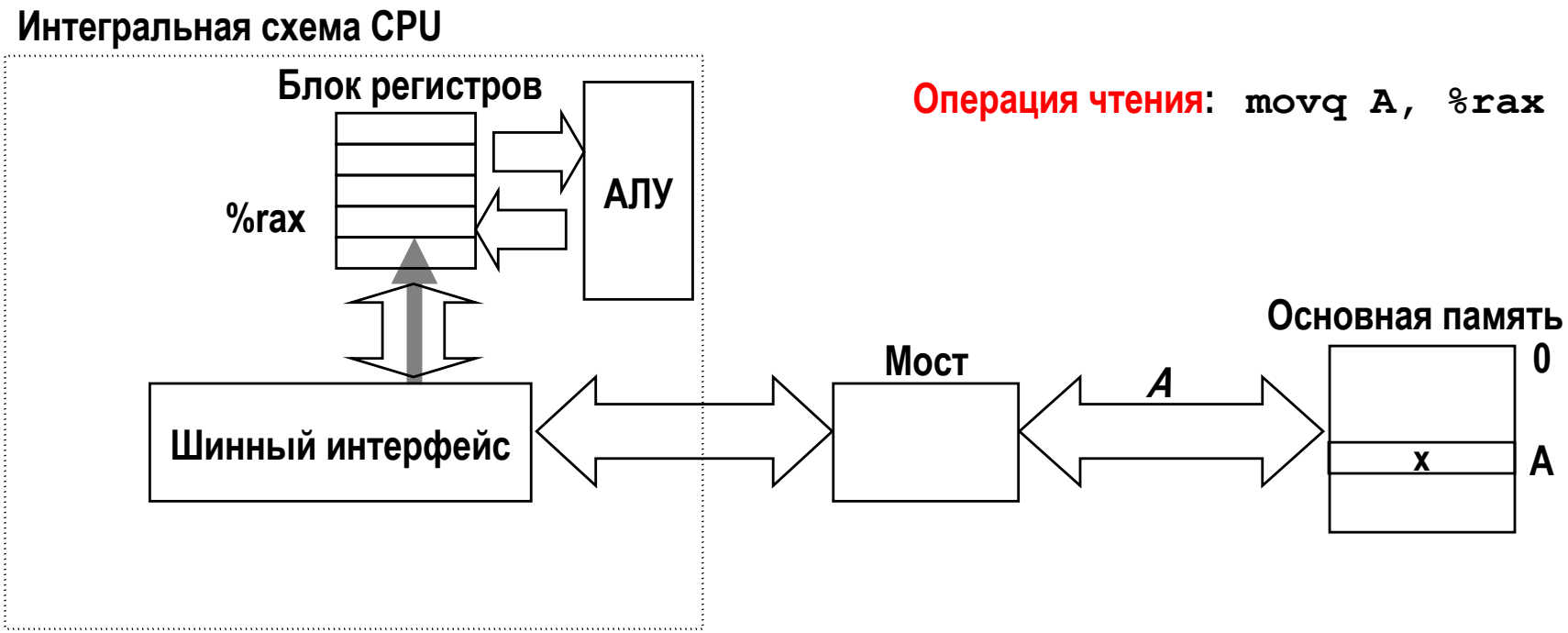
- Основная память принимает A с шины памяти, выбирает слово x, и выставляет его на шину




Операция чтения: `movq A, %rax`

# Чтение памяти - 3

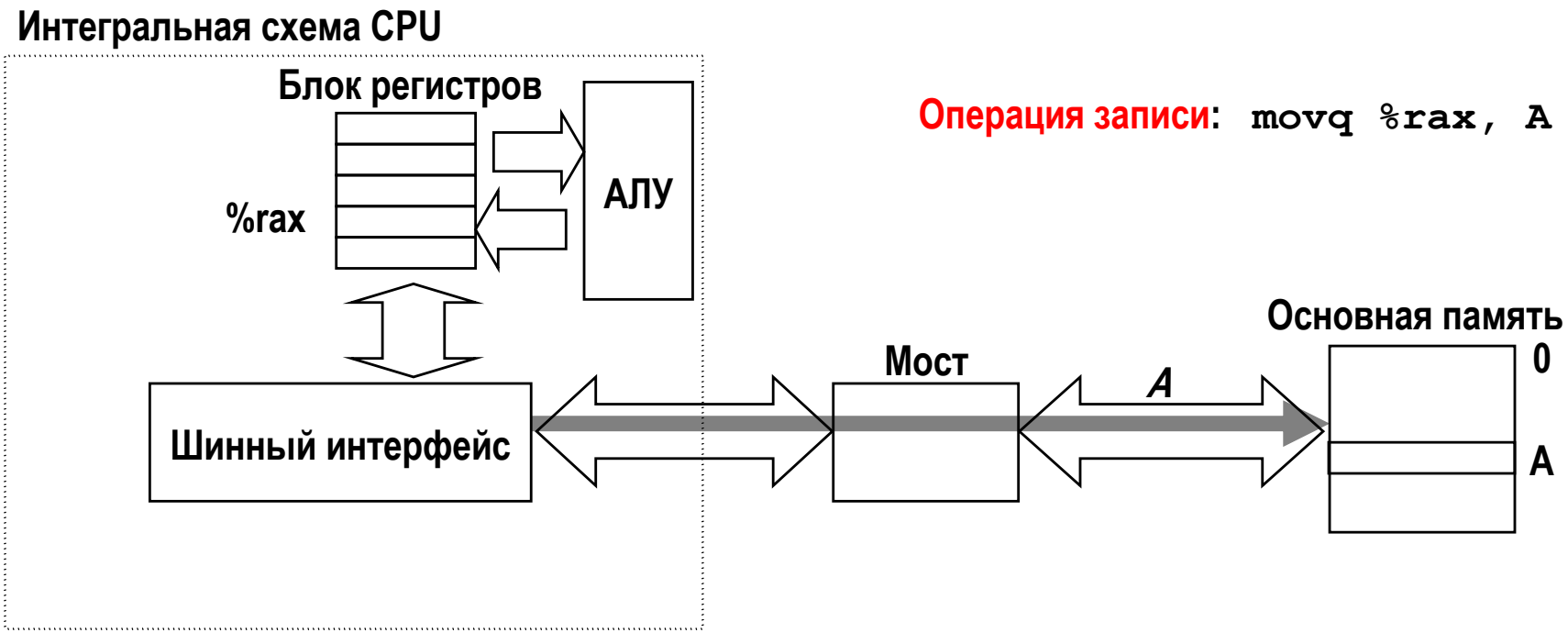
- ЦП считывает слово x с шины и помещает в регистр %rax.




Операция чтения: `movq A, %rax`

# Запись в память - 1

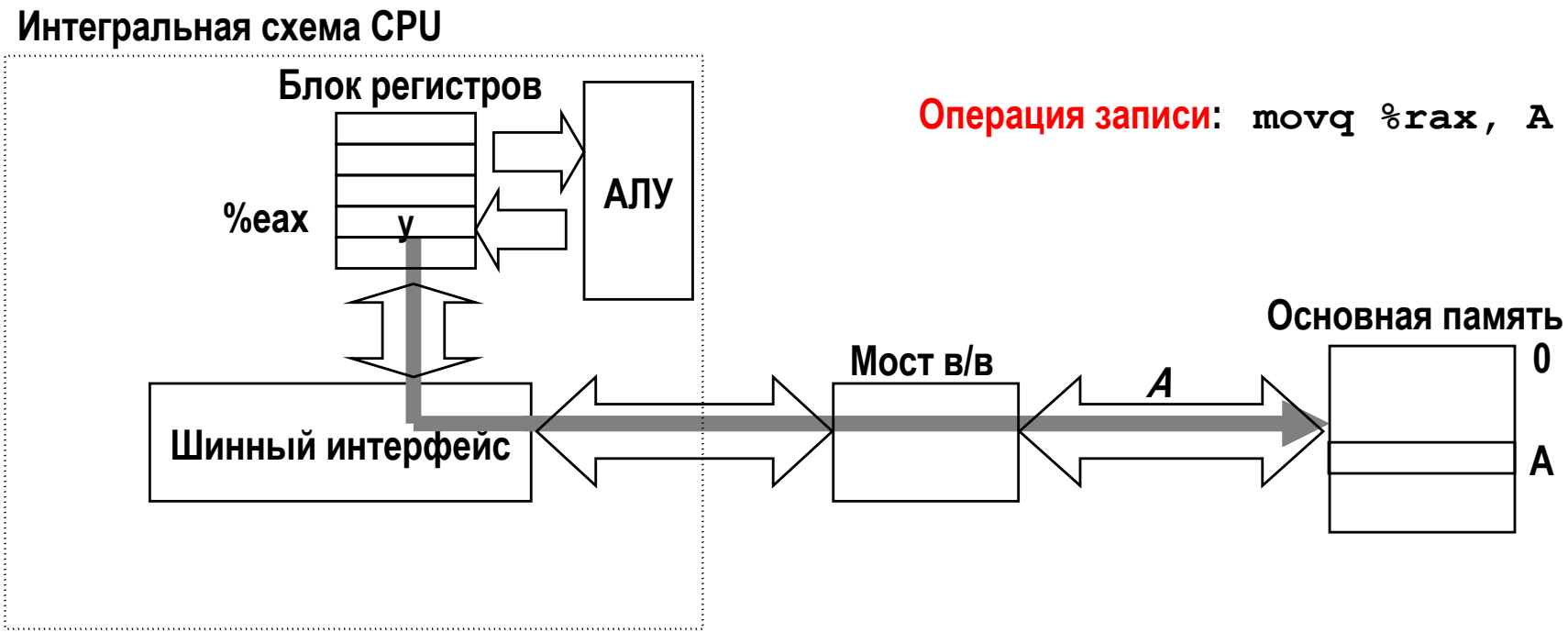
- ЦП выставляет на шину адрес A. Основная память считывает его и ожидает слова данных.




Операция записи: `movq %rax, A`

# Запись в память – 2

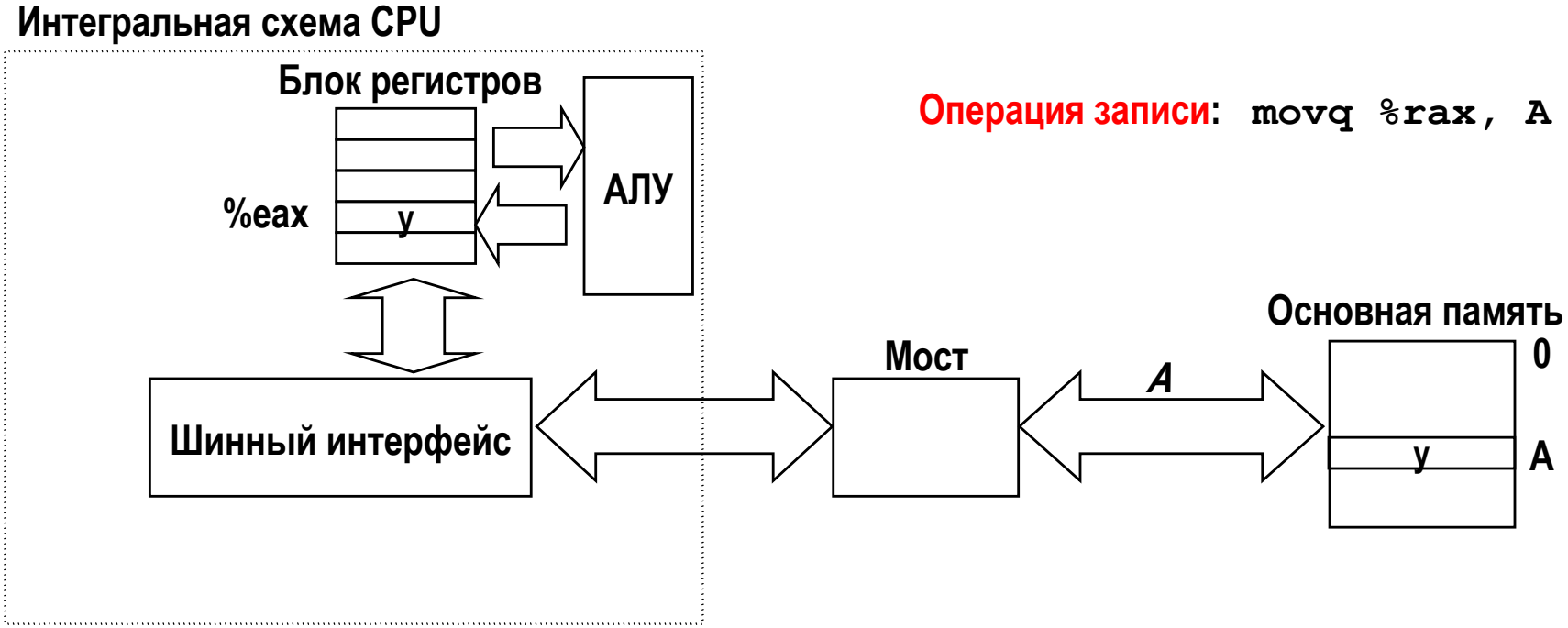
- **ЦП CPU выставляет на шину слово данных y.**




Операция записи: `movq %rax, A`

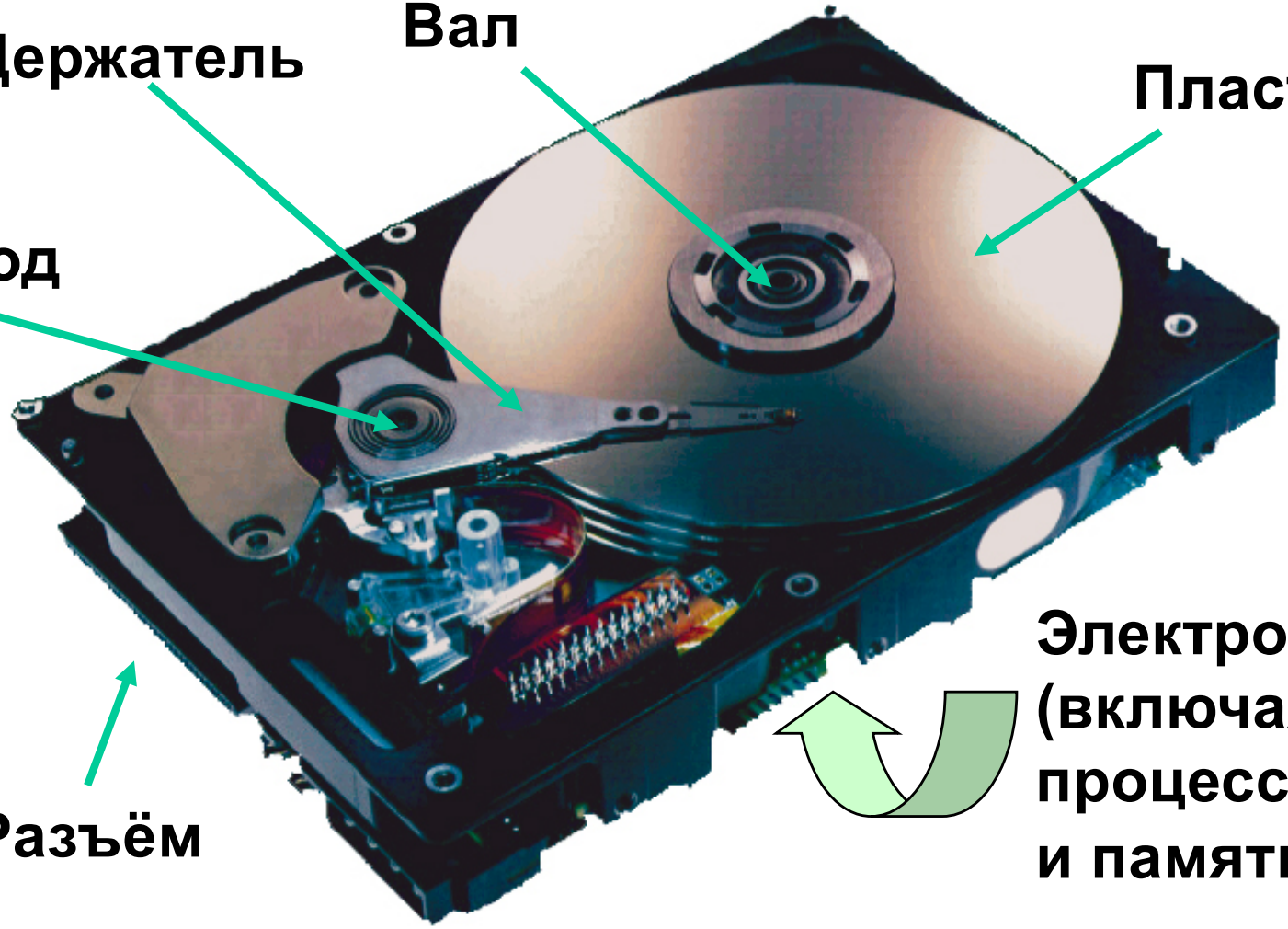
# Запись в память - 3

- Память считывает с шины слово данных y и размещает по адресу A.




Операция записи: `movq %rax, A`

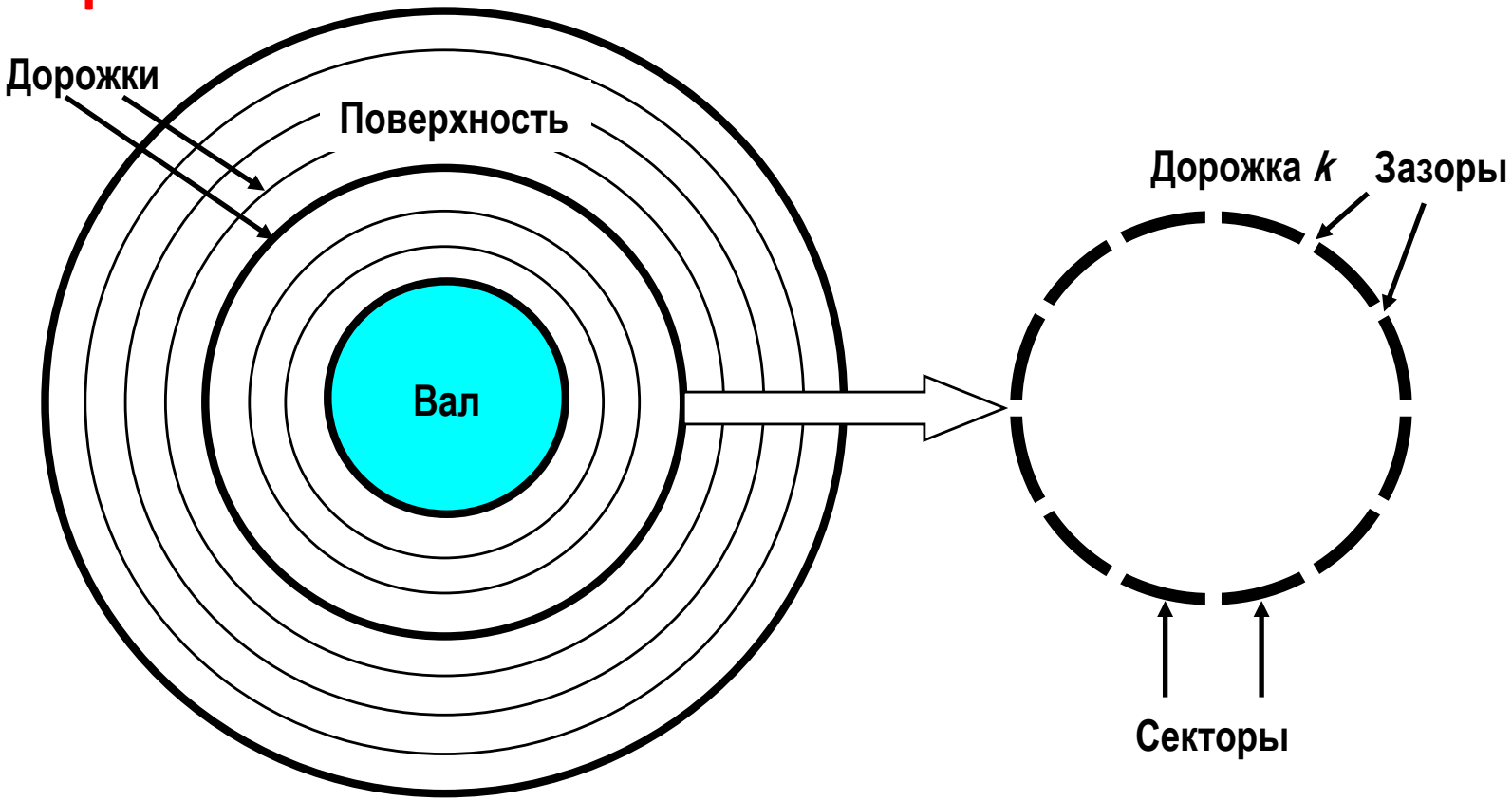
# Что внутри жёсткого диска?




□□□□□Seagate Technology

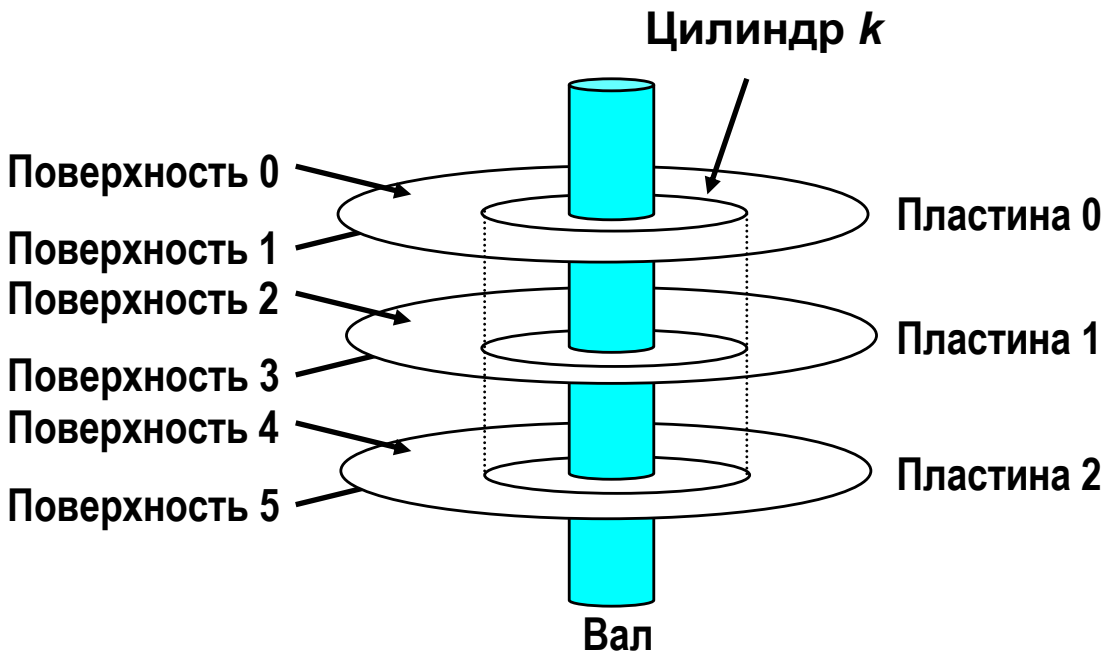
# Геометрия диска - 1

- Диск состоит из пластин, по две поверхности на каждой
- Каждая поверхность состоит из концентрических дорожек
- Каждая дорожка состоит из секторов разделённых зазорами



# Геометрия диска - 2

- Одинаковые дорожки образуют цилиндр

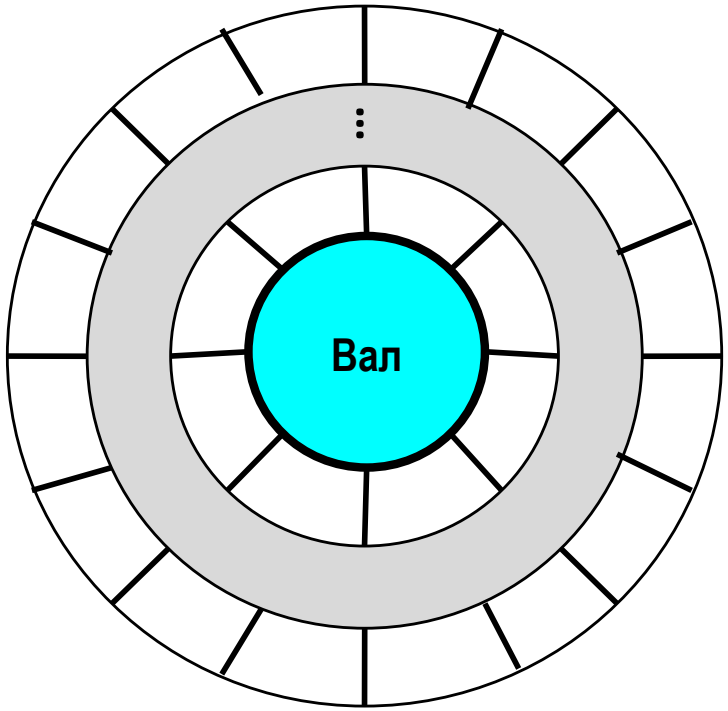


# Ёмкость диска

- **Ёмкость: максимальное количество хранимых байт**
  - Изготовители считают ёмкость в десятичных единицах
    - 1 GB = $10^9$ байт.
    - 1 TB = $10^{12}$ байт
- **Ёмкость определяется технологией:**
  - Плотность записи (бит/дюйм): количество бит в 1-дюймовом сегменте дорожки
  - Плотность дорожек (дорожек/дюйм): количество дорожек пересекающих 1 дюйм радиуса
  - Поверхностная плотность (бит/кв.дюйм): произведение плотностей записи и дорожек.

# Зоны записи

- **Современные диски разделяют всё множество дорожек на зоны записи**
  - В каждой дорожке одной зоны – одинаковое количество секторов определяемое самой внутренней дорожкой.
  - В каждой зоне своё своё значение секторов/дорожку.
    - во внешних – больше,
    - во внутренних – меньше
  - Для вычисления ёмкости используется **среднее** значение



# Вычисление ёмкости диска

**Ёмкость = (к-во байт/сектор) x (среднее к-во секторов/дорожку) x (к-во дорожек/поверхность) x (2 поверхности/пластину) x (к-во пластин/диск)**

**Пример:**

- 512 байт в секторе
- 300 секторов на дорожке (в среднем)
- 20 000 дорожек на поверхность
- 2 поверхности на пластине
- 5 пластин в диске

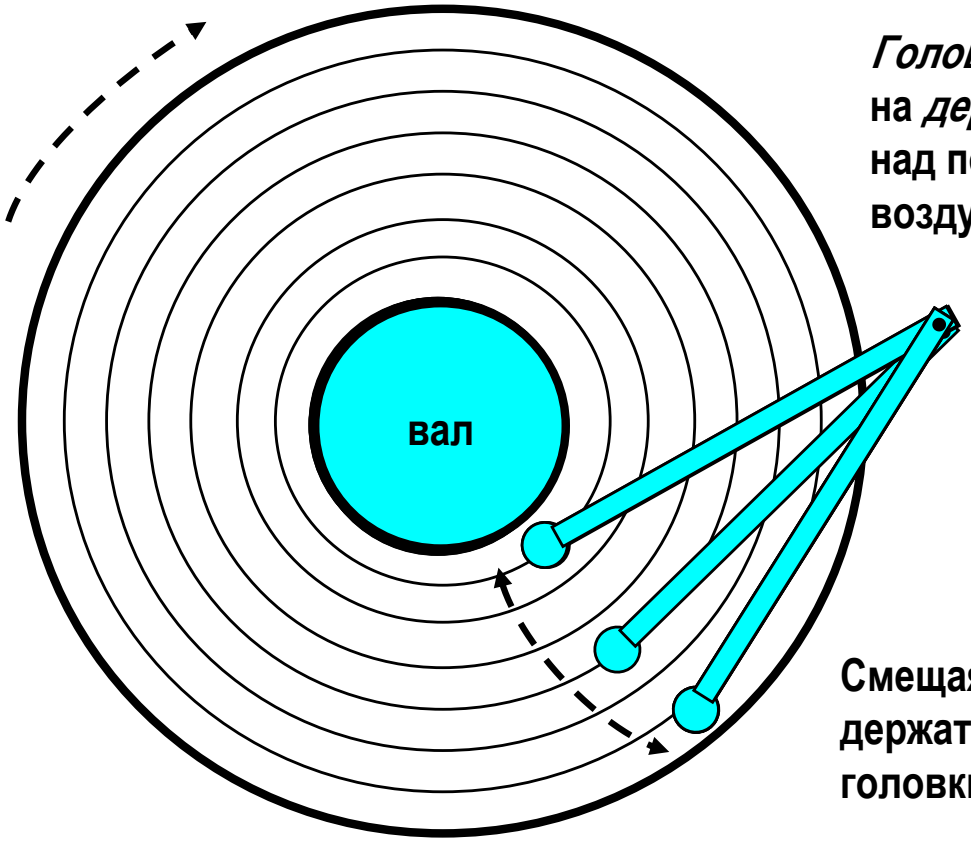
**Ёмкость = 512 x 300 x 20000 x 2 x 5**

**= 30,720,000,000**

**= 30.72 GB**

# Работа диска - 1

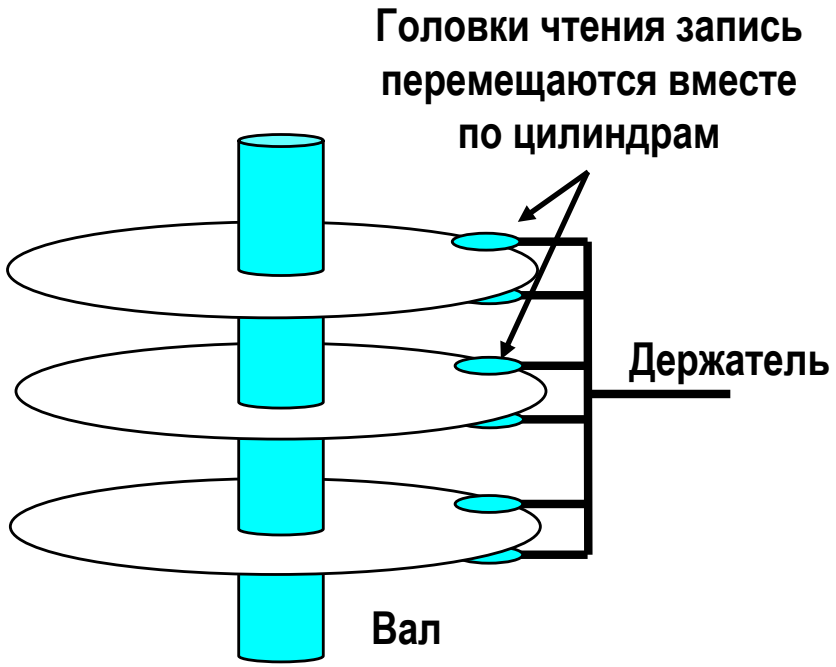
Поверхности вращаются с постоянной угловой скоростью



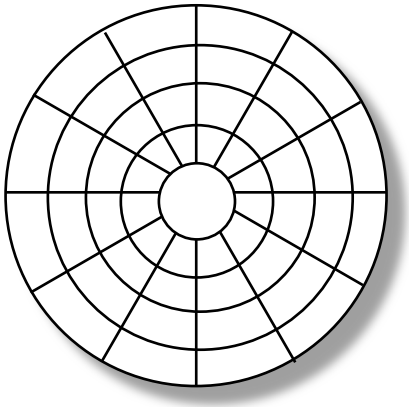

*Головки* чтения/записи на *держателях* «плавают» над поверхностью на прослойке воздуха

Смещаясь по радиусу держатель может разместить головки над любой дорожкой

# Работа диска - 2



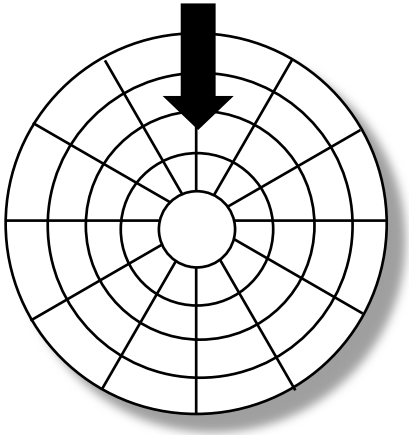
# Структура диска – вид на пластину



**Поверхность делится на дорожки**

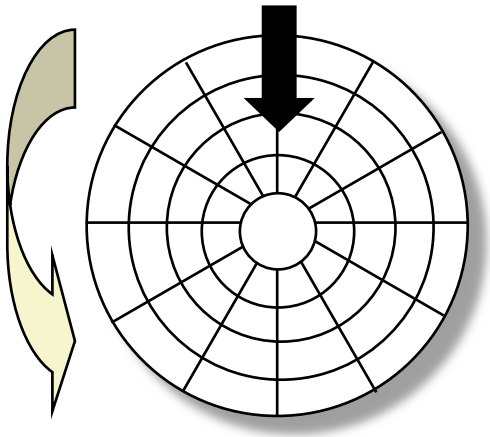
**Дорожки делятся на сектора**

# Доступ к диску - 1



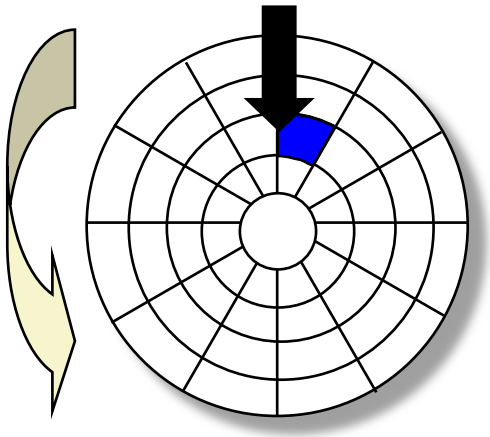
**Головка размещается над дорожкой**

# Доступ к диску - 2



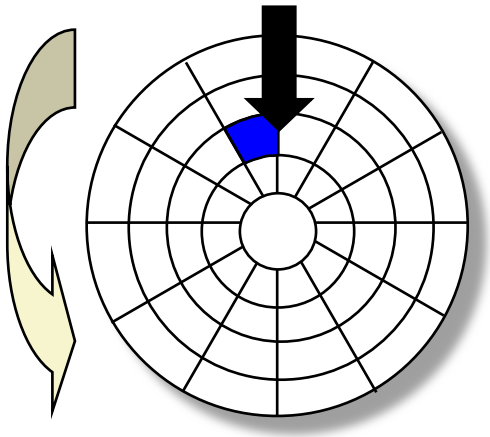
**Вращение против часовой стрелки**

# Доступ к диску - 3



**Перед чтением синего сектора**

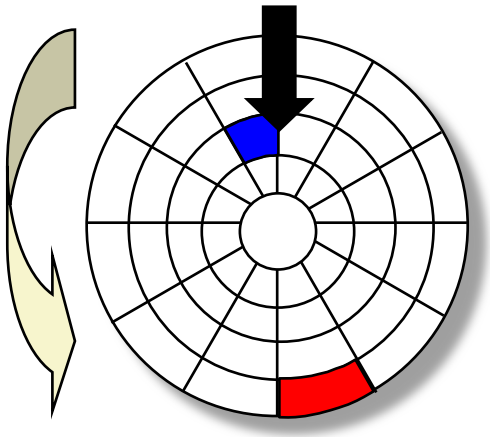
# Доступ к диску - 4



После чтения
СИНЕГО

## Данные переданы

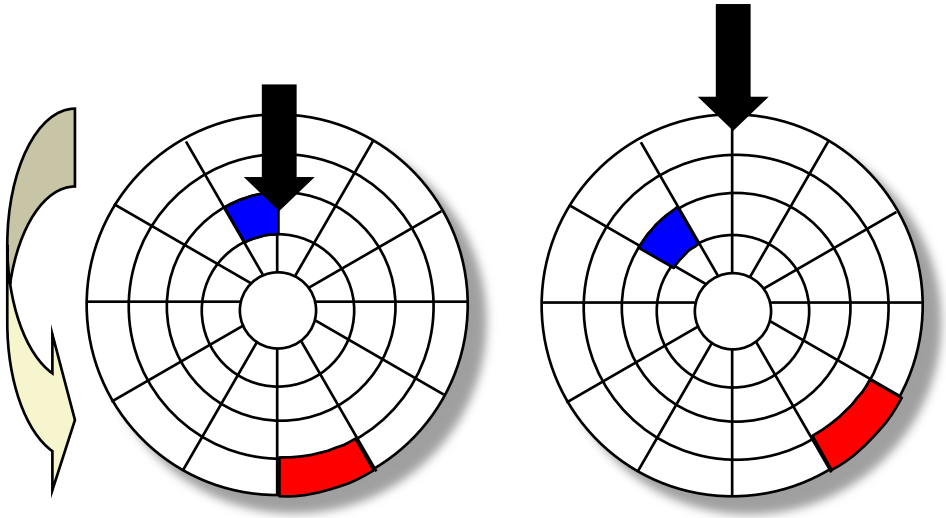
# Очередной запрос



После чтения
СИНЕГО

**Следующим требуется прочитать красный**

# Подвод

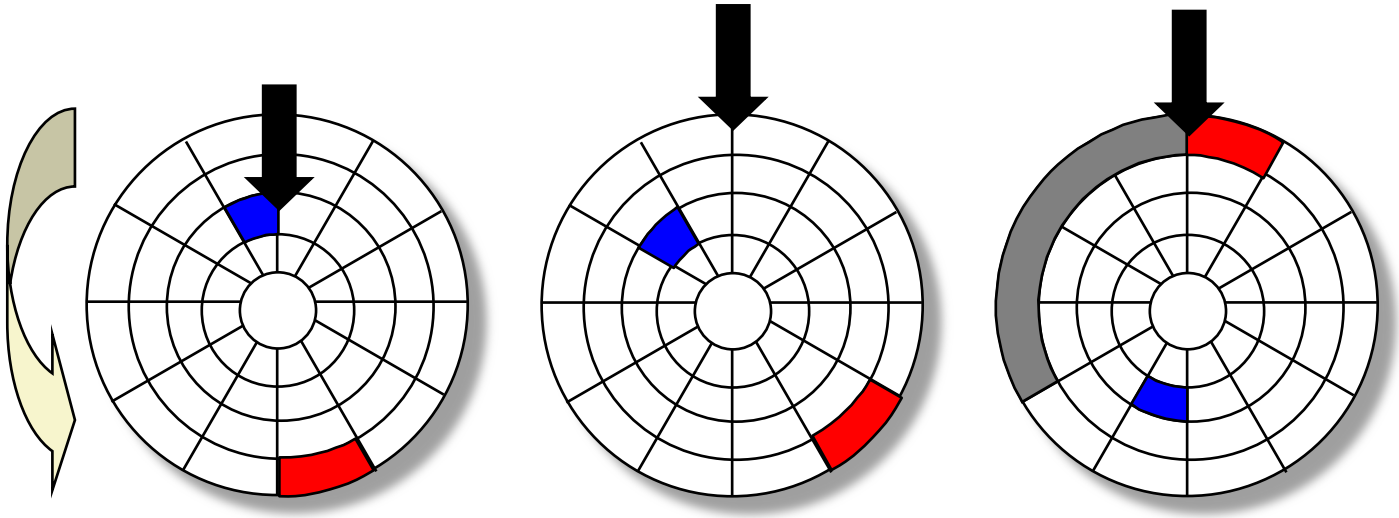


После чтения
СИНЕГО

Подвод к
КРАСНОМУ

**Подвод головки к дорожке красного**

# Доворот



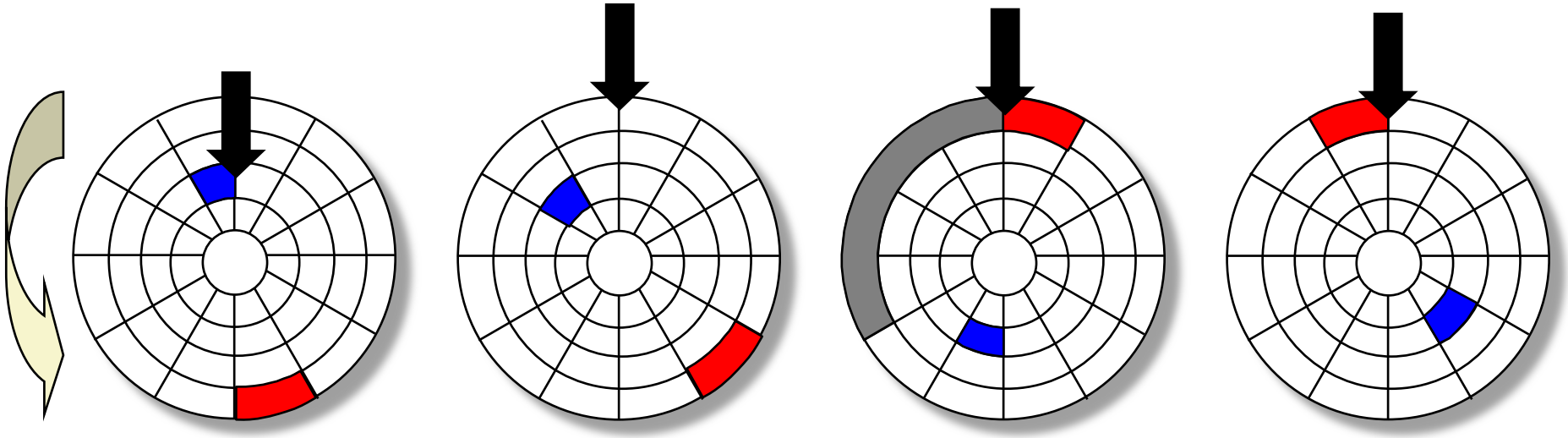
После чтения
СИНЕГО

Подвод к
КРАСНОМУ

Доворот к
КРАСНОМУ

## Ожидание красного сектора

# Чтение



После чтения
СИНЕГО

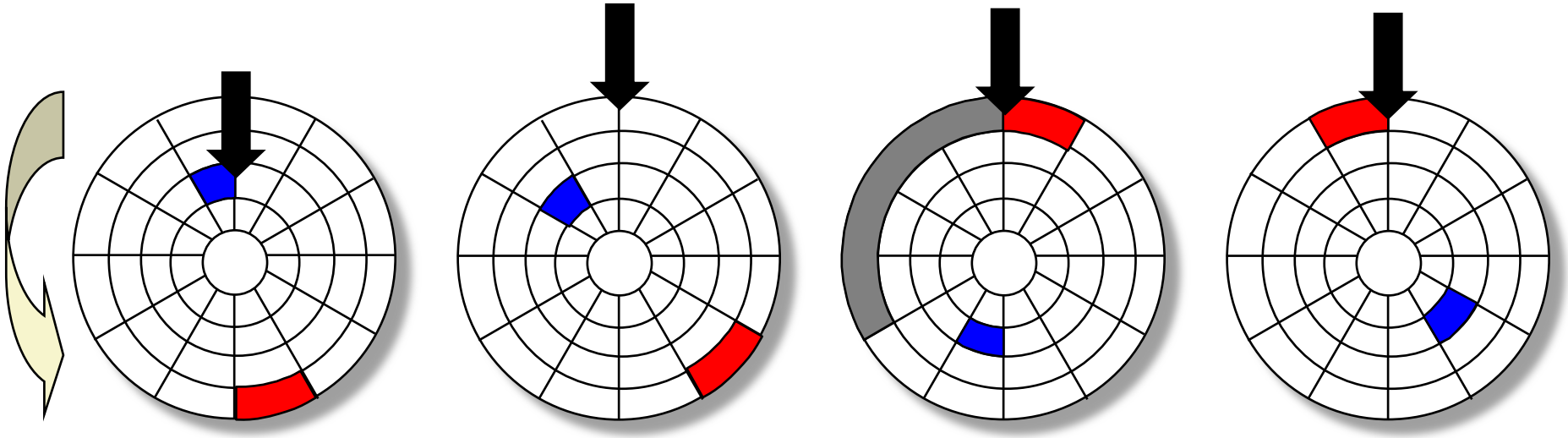
Подвод к
КРАСНОМУ

Доворот к
КРАСНОМУ

После чтения
КРАСНОГО

## Чтение и передача данных

# Составляющие времени выполнения



После чтения
СИНЕГО

Подвод к
КРАСНОМУ

Доворот к
КРАСНОМУ

После чтения
КРАСНОГО

Время
Подвода

Время
доворота

Время
передачи

# Время доступа к диску

- **Среднее время доступа к сектору :**
  - Т доступа ср. = Т подвода ср. + Т доворота ср. + Т чтения ср.
- **Время подвода головок (Т подвода ср.)**
  - Типичное Т подвода ср. = 3—9 мс
- **Время доворота пластин (Т доворота ср.)**
  - Т доворота ср. = 1/2 x 1/(частота вращения) x 60 сек/мин
  - Типичная частота вращения = 5400-15000 об/мин
- **Время чтения сектора (Т чтения ср.)**
  - Время считывания бит нужного сектора.
  - Т чтения ср. = 1/(частота вращения) x 1/(ср. к-во секторов/дорожку) x 60 сек/мин

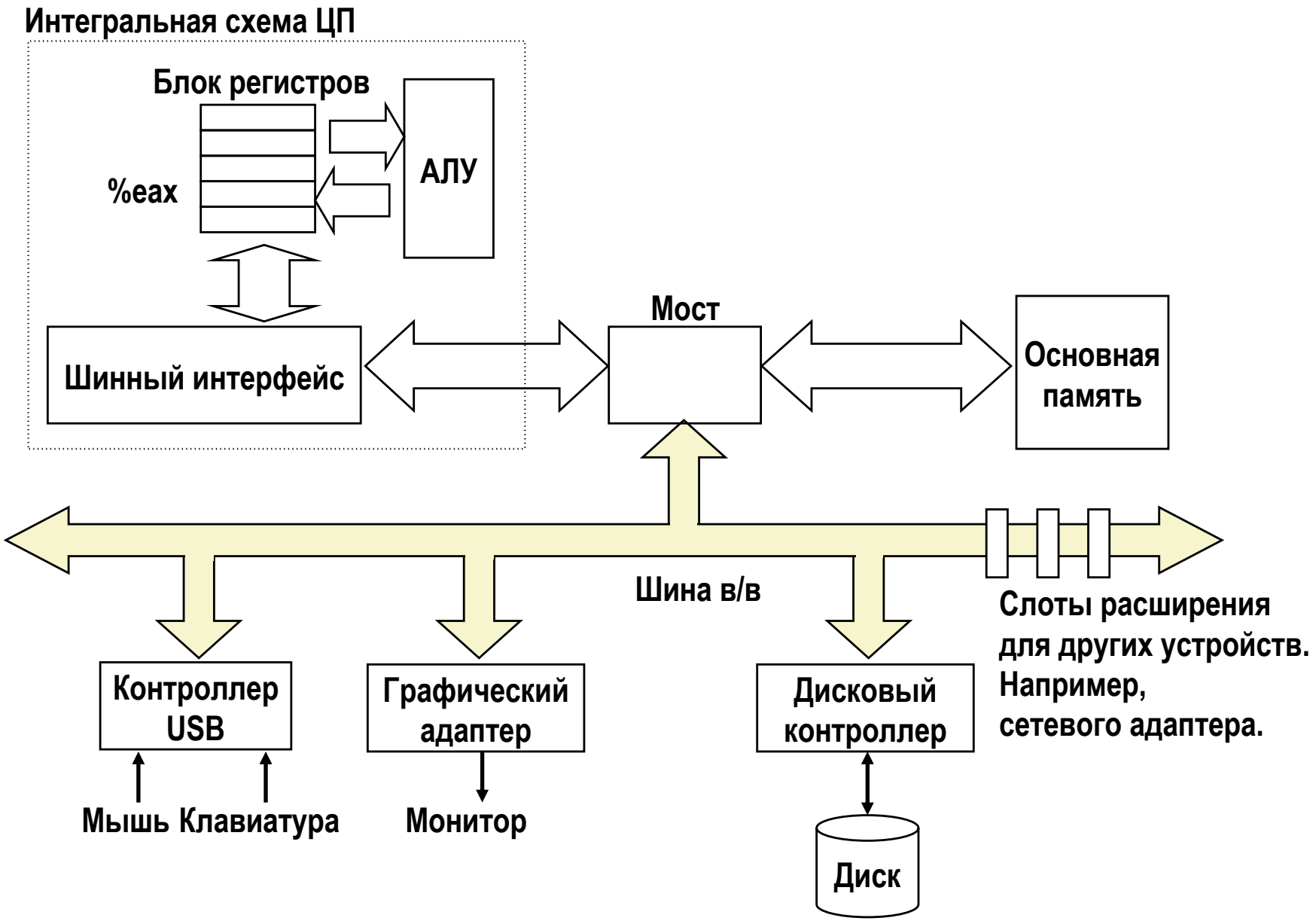
# Время доступа к диску: Пример

- **Дано:**
  - Скорость вращения = 7,200 об/мин
  - Среднее время подвода = 9 мс
  - Среднее количество секторов/дорожку = 400.
- **Считаем:**
  - Т доворота ср.= 1/2 x (60/7200) x 1000 ms/sec = 4 мс
  - Т чтения ср. = 60/7200 x 1/400 x 1000 = 0.02 мс
  - Т доступа ср. = 9 мс + 4 мс + 0.02 мс
- **Важно:**
  - Основные затраты времени на подвод и доворот
  - Доступ к первому биту в секторе – дорого, к остальным - даром.
  - Время доступа к SRAM - 4 нс, DRAM - 60 нс
    - Диск в 40 000 раз медленнее SRAM,
    - В 2 500 раз медленнее DRAM.

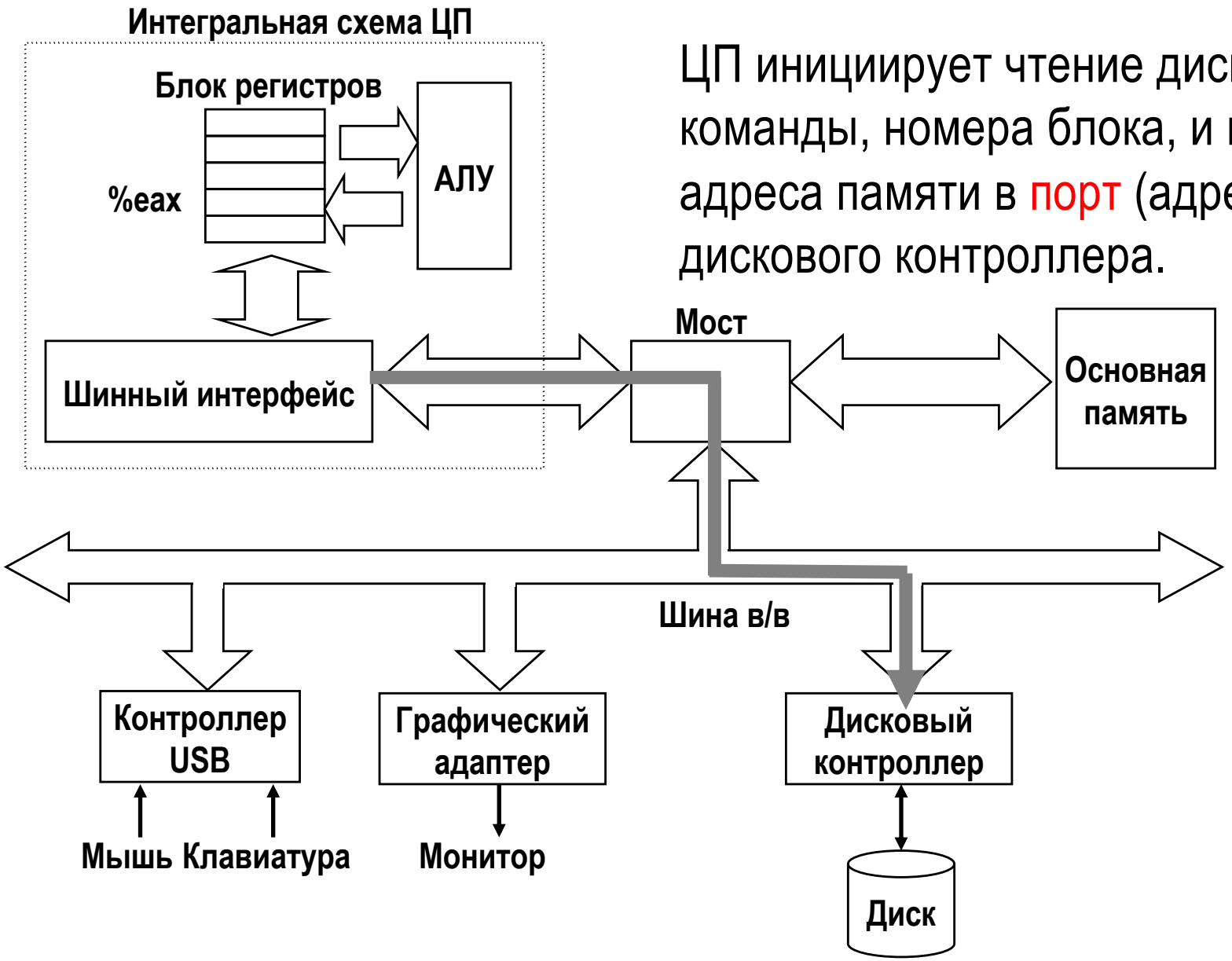
# Логические блоки диска

- **Современные диски предлагают более простой абстрактный вид геометрии :**
  - Набор доступных секторов представляется последовательностью логических блоков (0, 1, 2, ...)
- **Отображение логических блоков на физические сектора**
  - Выполняется отдельным устройством – дисковым контроллером
  - Преобразуется запрос логического блока в физическую триаду (поверхность, дорожка, сектор)
- **Контроллер резервирует несколько цилиндров в каждой зоне**
  - Различаются «форматированная» и «максимальная» ёмкости

# Шина ввода/вывода



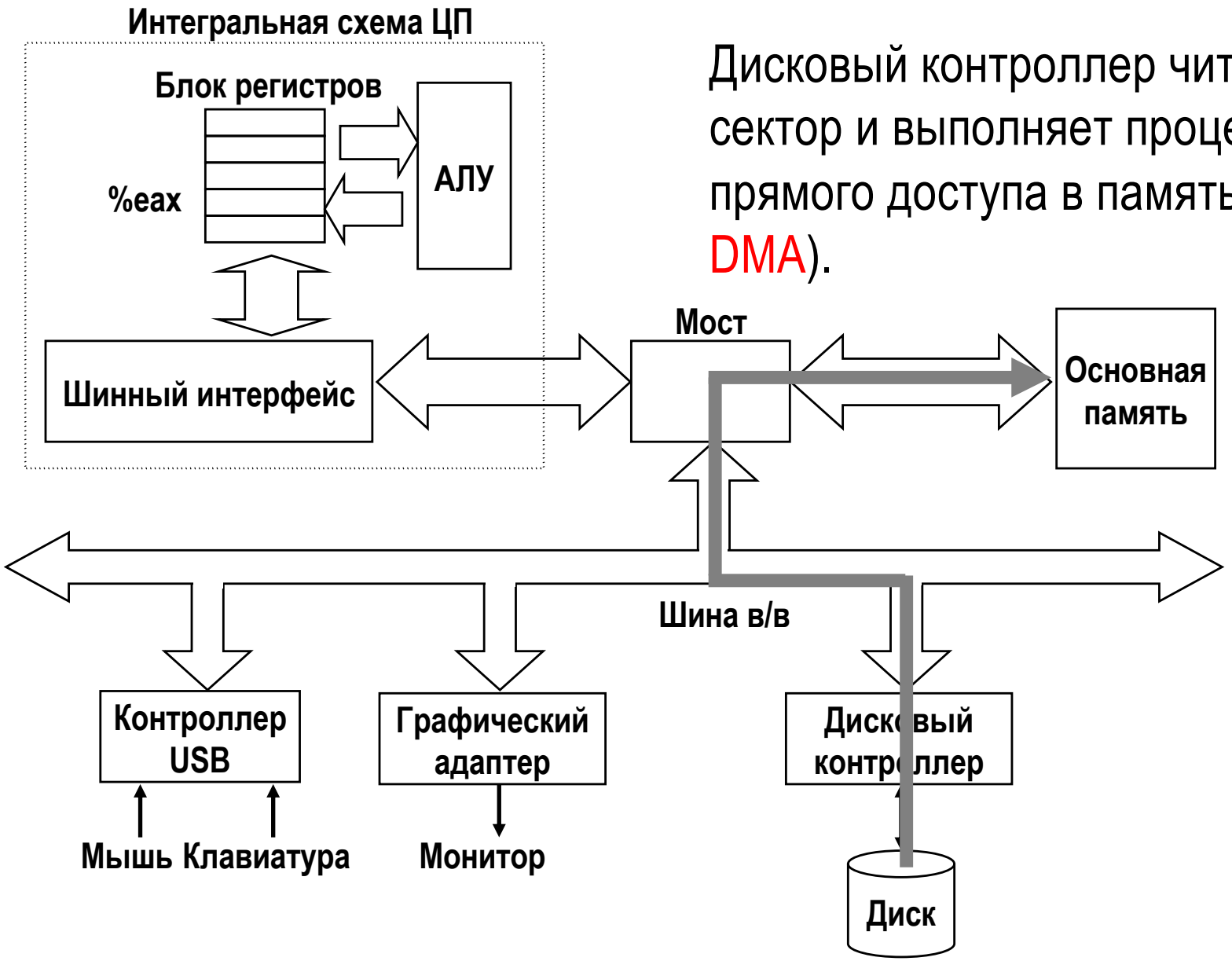
# Чтение сектора диска - 1




ЦП инициирует чтение диска записью команды, номера блока, и целевого адреса памяти в порт (адрес) дискового контроллера.

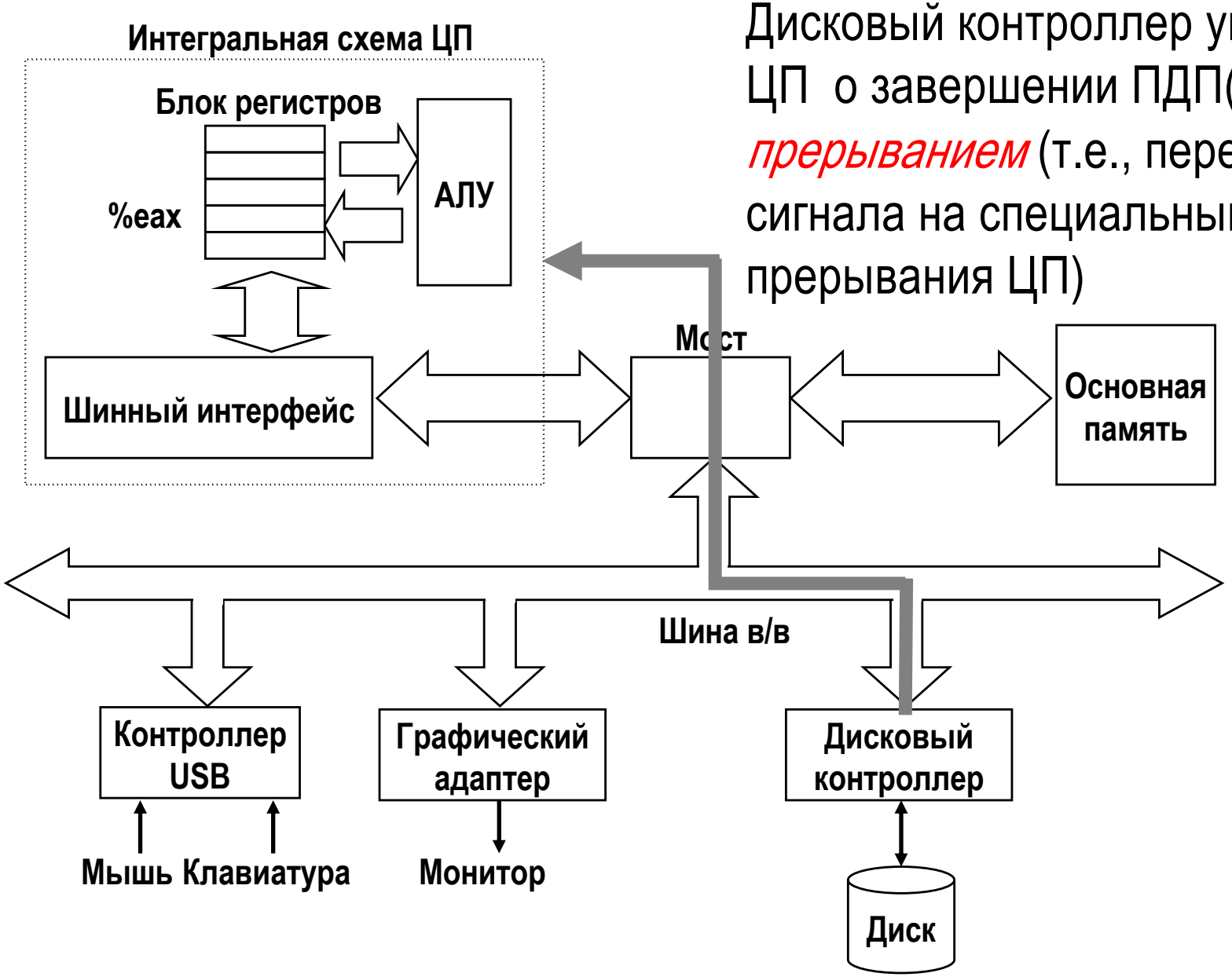
# Чтение сектора диска - 2

Дисковый контроллер читает сектор и выполняет процедуру прямого доступа в память (ПДП, DMA).

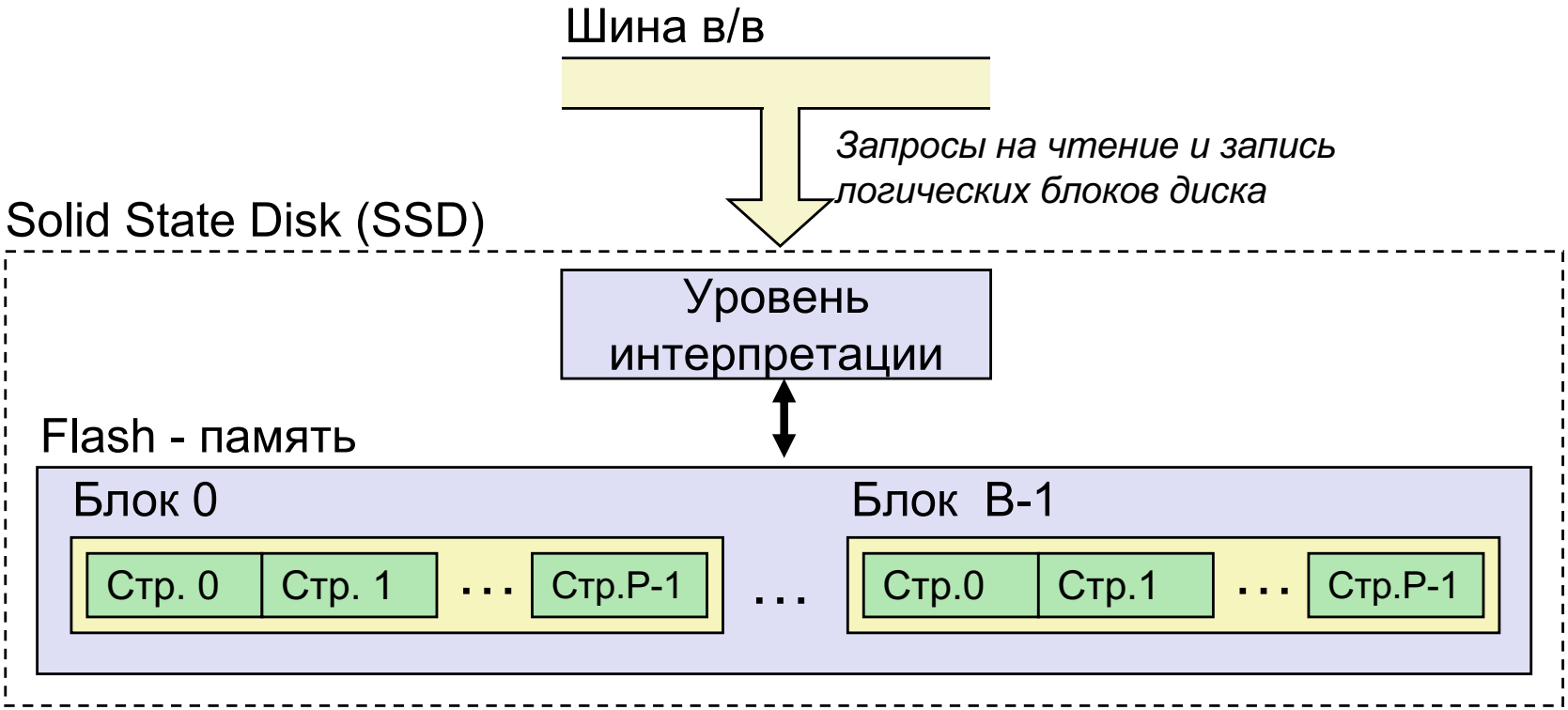


# Чтение сектора диска - 3

Дисковый контроллер уведомляет ЦП о завершении ПДП(DMA) *прерыванием* (т.е., передачей сигнала на специальный вывод прерывания ЦП)



# Твердотельные диски (SSDs)




- **Страницы: 512B - 4KB, Блоки: 32 - 128 страниц**
- **Чтение/запись страницами**
- **Страница м.б. записана только после стирания блока**
- **Блок изнашивается после 100 000 повторных записей**

# Быстродействие SSD

| | | | |
|---|---|---|---|
| **Последоват. чтение** | **550 МБ/сек** | **Последоват. Запись** | **470 МБ/сек** |
| **Произвольное чтение** | **365 МБ/сек** | **Произвольная запись** | **303 МБ/сек** |
| **Доступ к посл. чтению** | **50 мкс** | **Доступ к посл. записи** | **60 мкс** |

- **Последовательный доступ быстрее случайного**
- **Случайная запись немного медленнее чтения**
  - Блок стирается медленно (~1 ms)
  - Запись в страницу вызывает копирование всех остальных страниц блока в новый
  - В ранних SSD различия в скоростях записи и чтения много больше

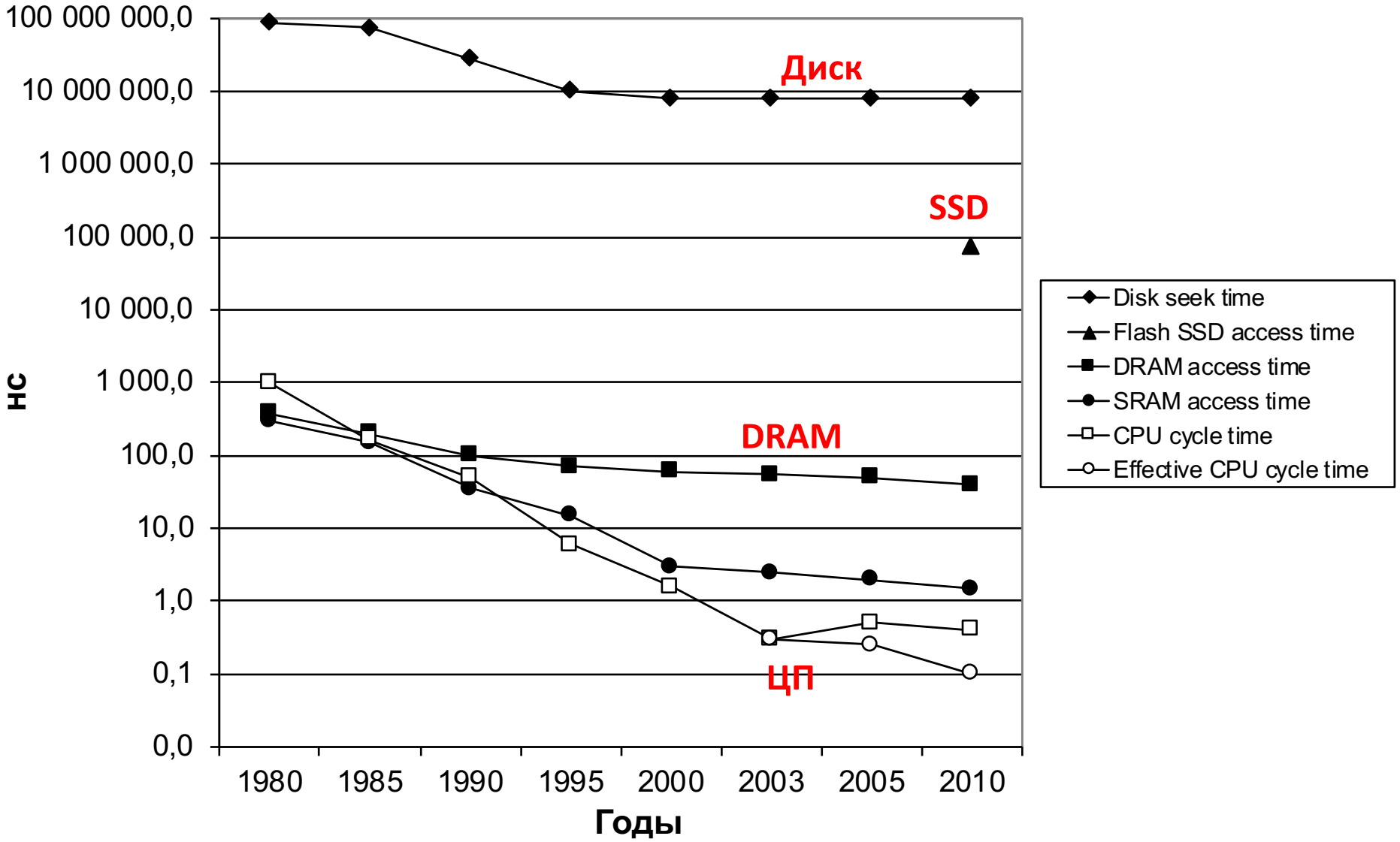
**Источник: спецификация Intel SSD 730**

# Особенности твердотельных дисков

- **Преимущества**
  - Без движущихся частей → быстрее, экономнее, устойчивее
- **Недостатки**
  - Изнашивание
    - Нивелируется “логикой равномерного износа” на уровне интерпретации
    - Например Intel SSD 730 гарантирует 128 петабайт ($128 \times 10^{15}$) байт) записей до износа
  - В 2015, удельная стоимость в ~30 раз выше
- **Применения**
  - MP3 плееры, смартфоны, планшеты, ноутбуки
  - Высокопроизводительные настольные ПК и сервера

# Пропасть ЦП-память

## Растёт пропасть между скоростями DRAM, диска, и ЦП



# Спасение в локальности!

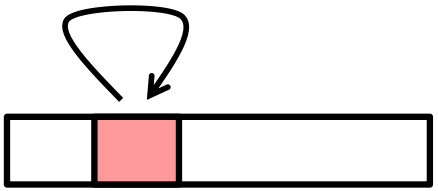
**Ключ к переходу через пропасть память-процессор – это фундаментальное свойство компьютерных программ известное как локальность**

# Иерархия памятей

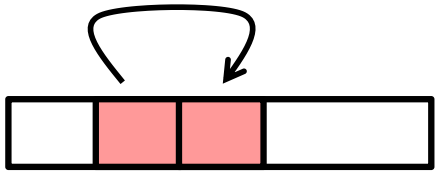
- Технологии и тренды
- Локальность обращения
- Кеширование в иерархии памятей

# Локальность

- **Принцип локальности : Программы чаще обращаются к данным и командам, чьи адреса близкие к тем которые недавно использовались**



- **Временна́я локальность:**
  - Недавно использованные ячейки вероятно вскоре будут использованы вновь



- **Пространственная локальность:**
  - К ячейкам с ближайшими адресами обращения произойдут в ближайшее время

# Пример локальности

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

- **Обращение к данным**
  - Последовательная выборка элементов массива — **Пространственная**
  - Обращение к `sum` в каждой итерации — **Временна́я**
- **Обращение к командам**
  - Последовательная выборка команд — **Пространственная**
  - Повторение команд цикла — **Временна́я**

# Качественная оценка локальности

- **Утверждение:** Способность взглянув на код получить качественную оценку его локальности – ключевой навык профессионального программиста.

- **Вопрос:** Эта функция имеет хорошую локальность в отношении массива `a`?

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

# Пример локальности

- **Вопрос: Эта функция имеет хорошую локальность в отношении массива `a`?**

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

# Пример локальности

- **Вопрос: Вы можете переставить циклы так, чтобы функция сканировала 3-мерный массив `a` также эффективно как одномерный? С хорошей пространственной локальностью.**

```
int sum_array_3d(int a[M][N][N])
{
    int i, j, k, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            for (k = 0; k < N; k++)
                sum += a[k][i][j];
    return sum;
}
```

# Иерархия хранения данных

- **Некоторые фундаментальные и стабильные свойства аппаратуры и ПО:**
  - Более быстрые технологии хранения удельно дороже, менее ёмкие, и более энергоёмкие (нагрев!).
  - Пропасть «процессор-память» растёт.
  - Хороший код демонстрирует хорошую локальность.

- **Эти фундаментальные свойства замечательно дополняют друг друга**

- **Это подсказывает подход к организации подсистемы памяти известный как иерархия хранения данных.**

# Иерархия памятей

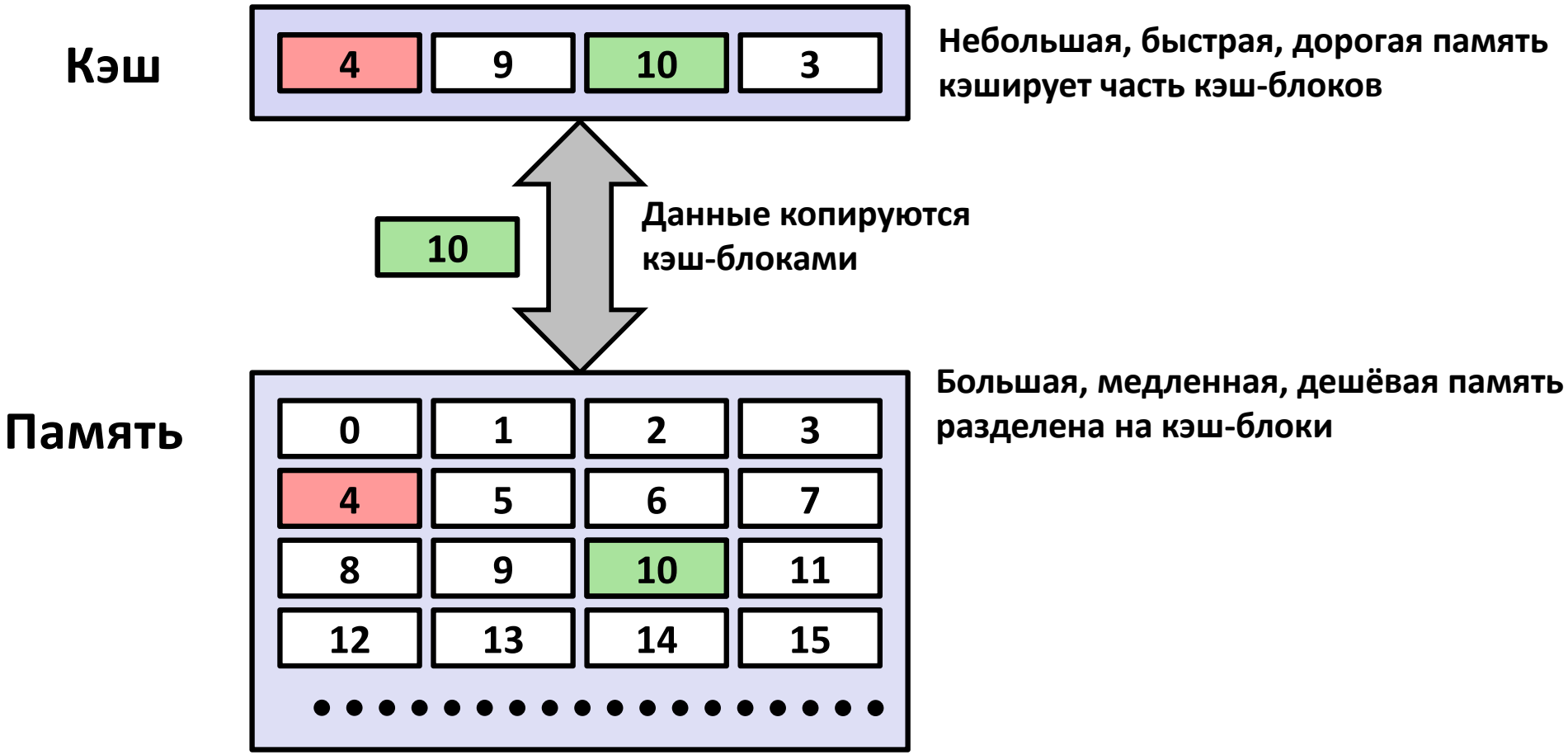
- Технологии и тренды
- Локальность обращения
- Кеширование в иерархии памятей

# Пример иерархии памятей

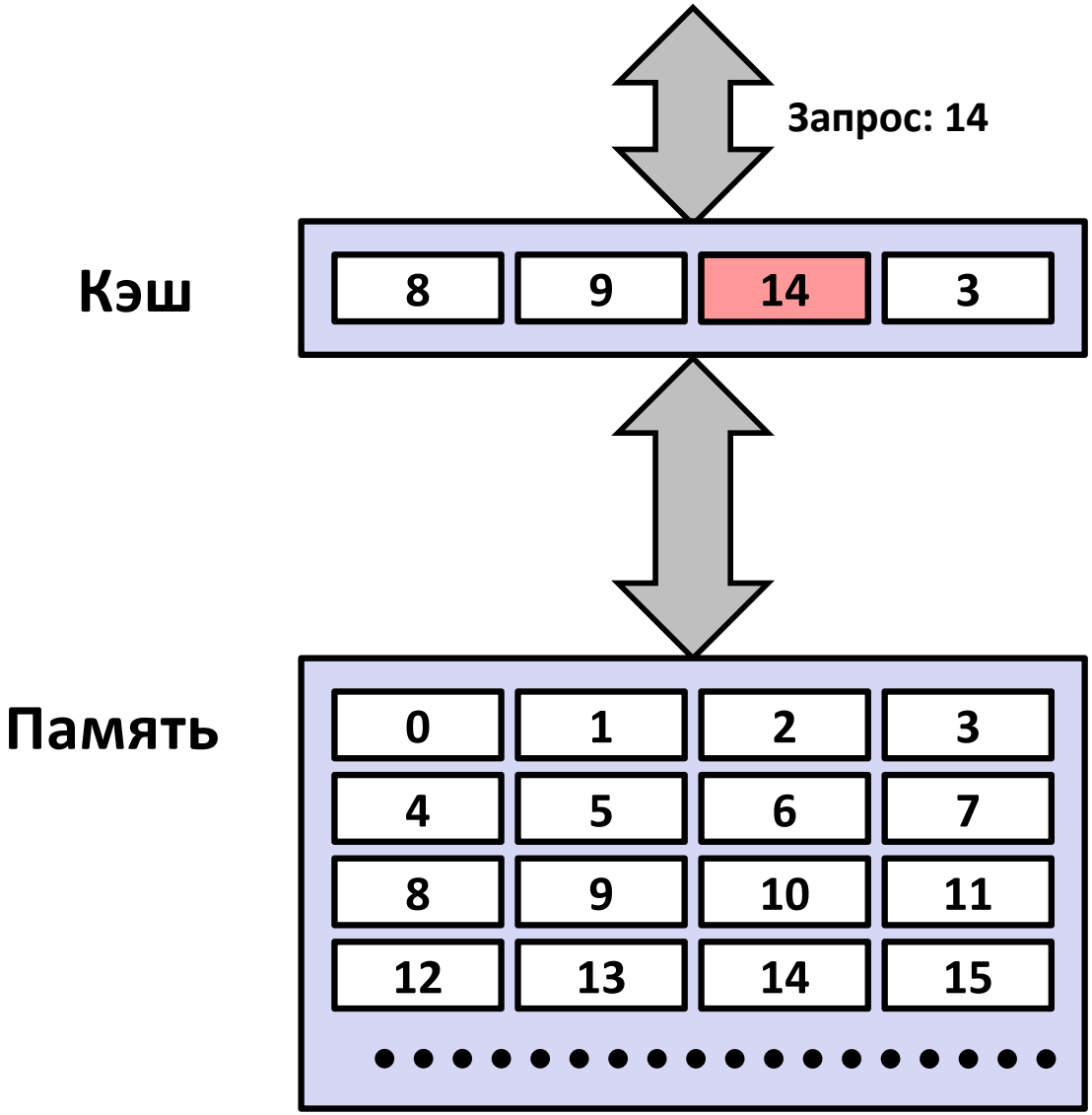
# Кеши

- ***Кеш(Cache):*** **Устройство хранения меньшей ёмкости и большего быстродействия, действующее как вспомогательное для доступа к части данных более крупного и медленного устройства.**
- **Фундаментальная идея иерархии:**
  - Устройство уровня k служит кешем для устройства уровня k+1.
- **Почему работает иерархия хранения данных?**
  - Благодаря локальности программы к данным уровня k доступ происходит чаще, чем к данным уровня k+1.
  - Значит уровень хранения k+1 может быть медленнее, а значит больше и дешевле (на единицу хранения)
- ***Идеал:*** **Иерархия предоставляет большой объём хранения по цене экономичного нижнего уровня с скоростью доступа верхнего уровня для программ.**

# Кэш в общем


Кэш
4
9
10
3
Небольшая, быстрая, дорогая память кэширует часть кэш-блоков
10
Данные копируются кэш-блоками
Память
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Большая, медленная, дешёвая память разделена на кэш-блоки

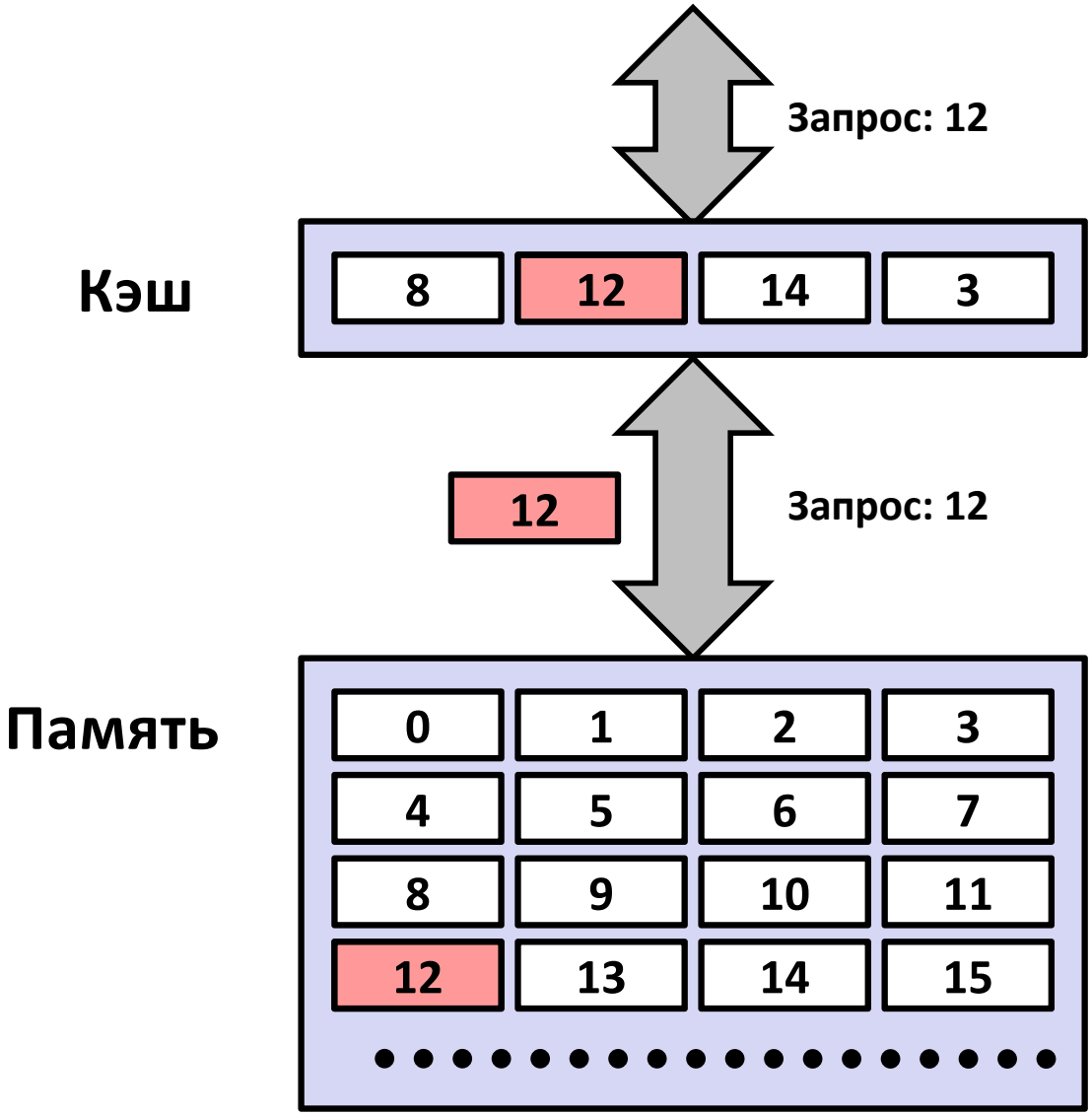
# Кэш в общем: Попадание




*Нужен блок данных b*

*Блок b в кэше есть:*
*Попадание!*

# Кэш в общем: Промах




***Нужен блок данных b***

***Блока b в кэше нет:***
***Промах!***

***Блок b поступает из памяти***

***Блок b помещается в кеше***

- **Политика размещения**: определяет, где b в кэше
- **Политика замещения**: определяет удаляемый блок (жертву)

# Кэш в общем: типы промахов

- **Холодный промах**
  - Кеш пуст. Например при старте.
- **Конфликтный промах**
  - Большинство кэшей ограничивает (вплоть до 1) количество позиций размещения в кэше для каждого кэш-блока памяти.
    - Пример: блок i в памяти должен размещаться в (i mod 4) позиции кэша.
  - Конфликтный промах, если в кэше есть место, но несколько блоков памяти претендуют на одно место в кэше.
    - Пример: обращения к блокам 0, 8, 0, 8, 0, 8, ... промахиваются всегда.
- **Промах ёмкости**
  - Множество используемых кэш-блоков, рабочий набор (working set) превышает размер кэша.

# Пример кэширования в иерархии

| Тип кэша | Что кэшируем? | Где кэшируем? | Задержка (циклов) | Кто управляет? |
|---|---|---|---|---|
| Регистры | Слова 4-8 байт | Ядро ЦП | 0 | Компилятор |
| TLB | Трансляция адресов | TLB на кристалле ЦП | 0 | Аппаратура |
| Кэш уровня 1 | Блок 64 байта | На кристалле ЦП | 1 | Аппаратура |
| Кэш уровня 2 | Блок 64 байта | На кристалле ЦП | 10 | Аппаратура |
| Виртуальная пам. | Страница 4-КБ | Основная память | 100 | Аппарат.+ОС |
| Дисковый кэш ОС | Части файлов | Основная память | 100 | ОС |
| Кэш на диске | Сектора дисков | Контроллер диска | 100,000 | ВПО в диске |
| Сетевой | Части файлов | Местный диск | 10,000,000 | Клиент сетевой ФС |
| Кэш веб-браузера | Части веб-страниц | Местный диск | 10,000,000 | Веб браузер |
| Кэш веб-прокси | Части веб-страниц | Удалённый сервер | 1,000,000,000 | Сервер веб-прокси |

# Сводка

- **Различия в скорости конструктивно разделённых ЦП, ОЗУ и дисков продолжают нарастать.**

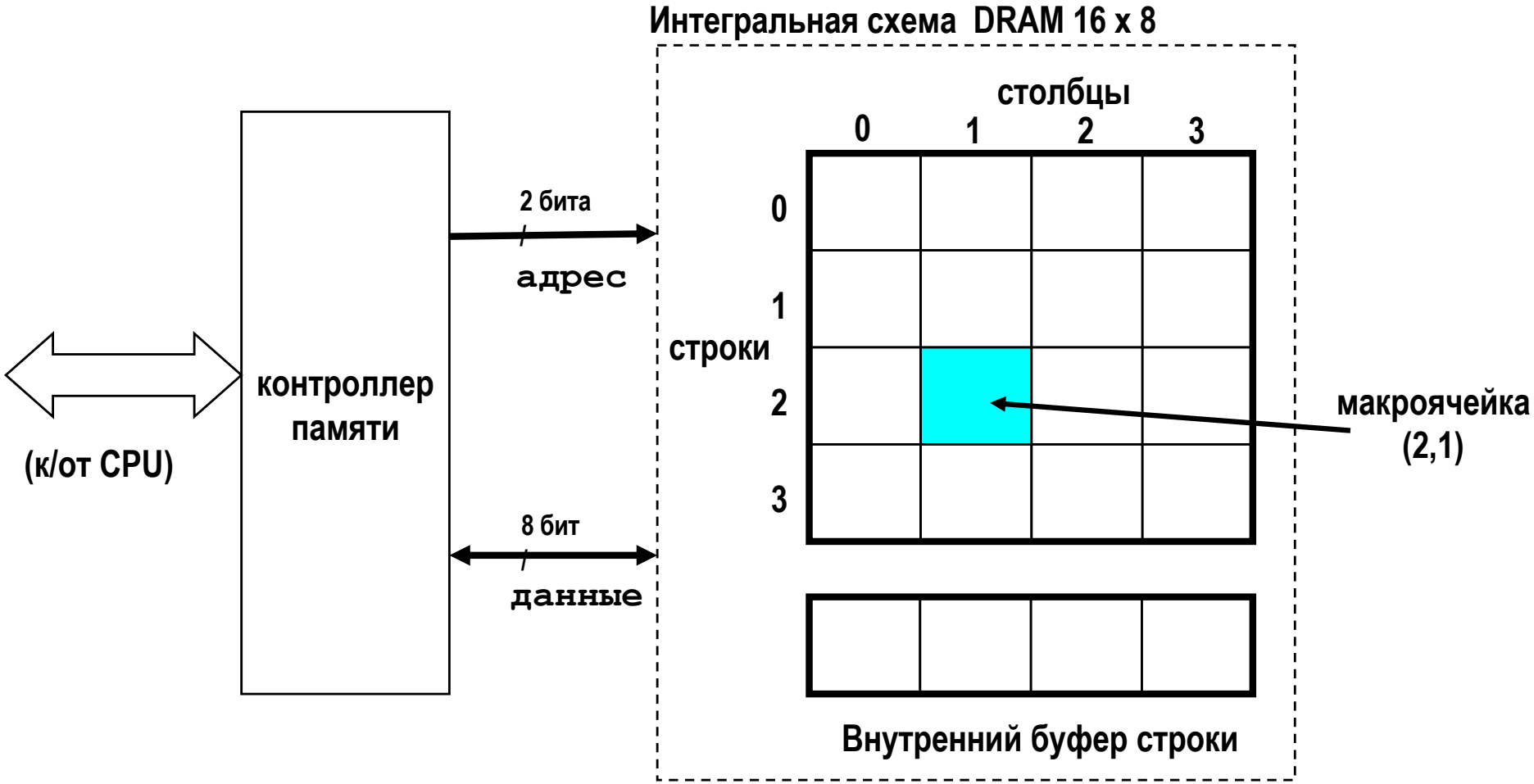
- **Хорошо написанные программы, алгоритмы, постановки задач демонстрируют свойство, называемое «локальность».**

- **Иерархии памятей основанные на кешировании нивелируют различия видов памяти используя локальность.**

# Дополнительные слайды

# Обычная организация DRAM

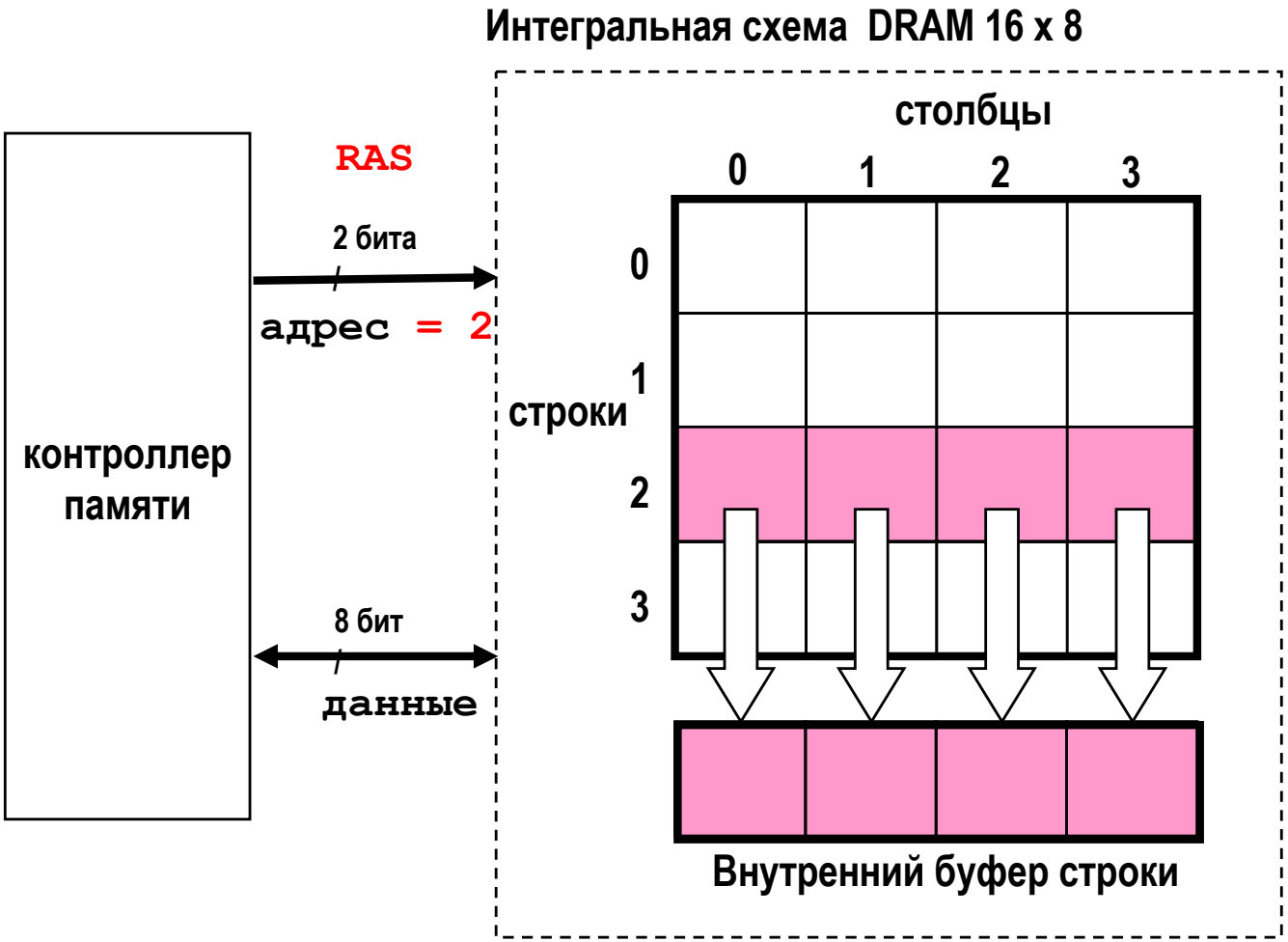
- **d x w DRAM:**
  - dw бит организуются в d макроячеек размером w бит



# Чтение макроячейки DRAM (2,1)

**Шаг 1(a): Выбор строки 2 по сигналу «Row access strobe» (RAS).**

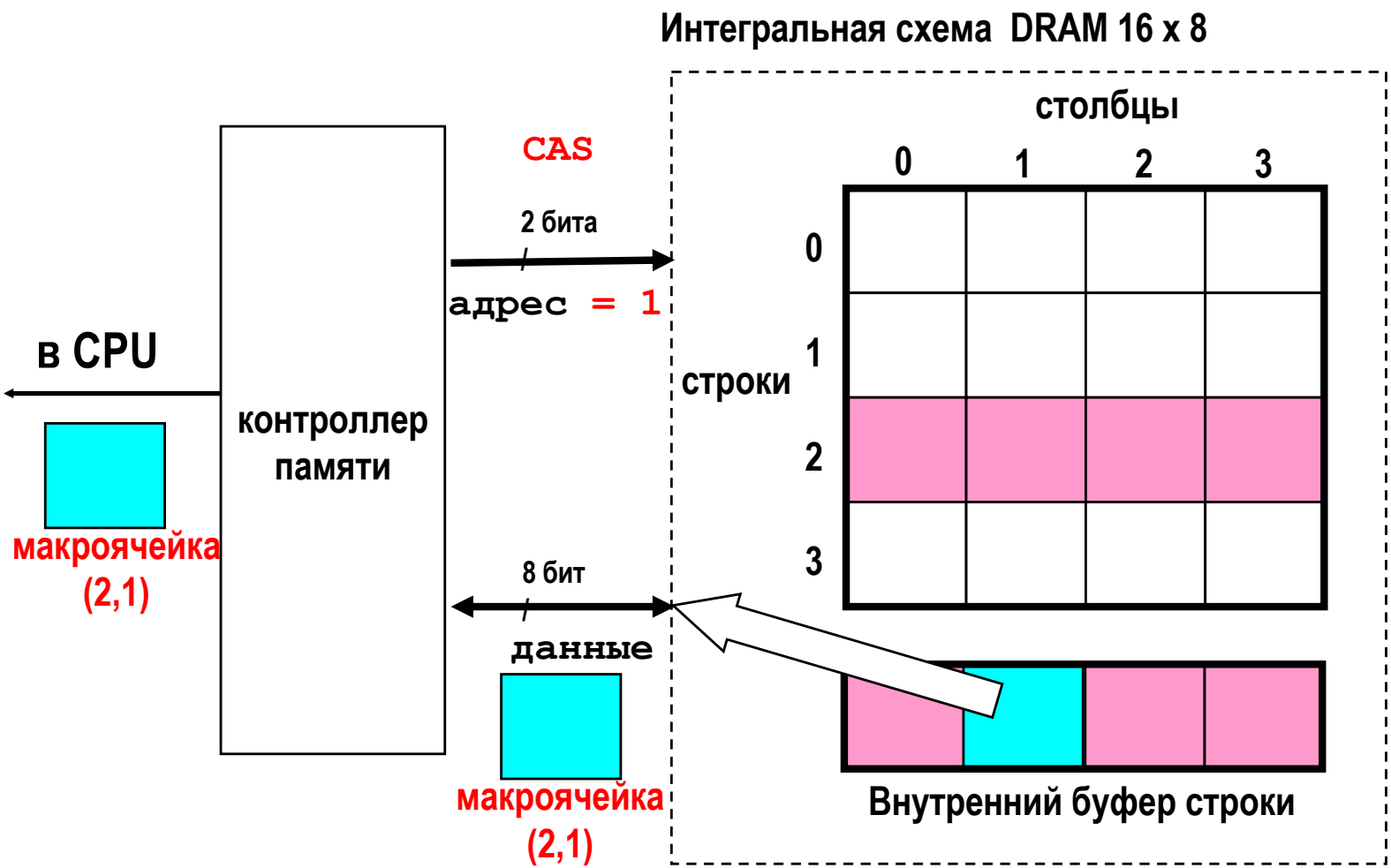
**Шаг 1(b): Строка 2 копируется из массива DRAM в буфер стоки.**



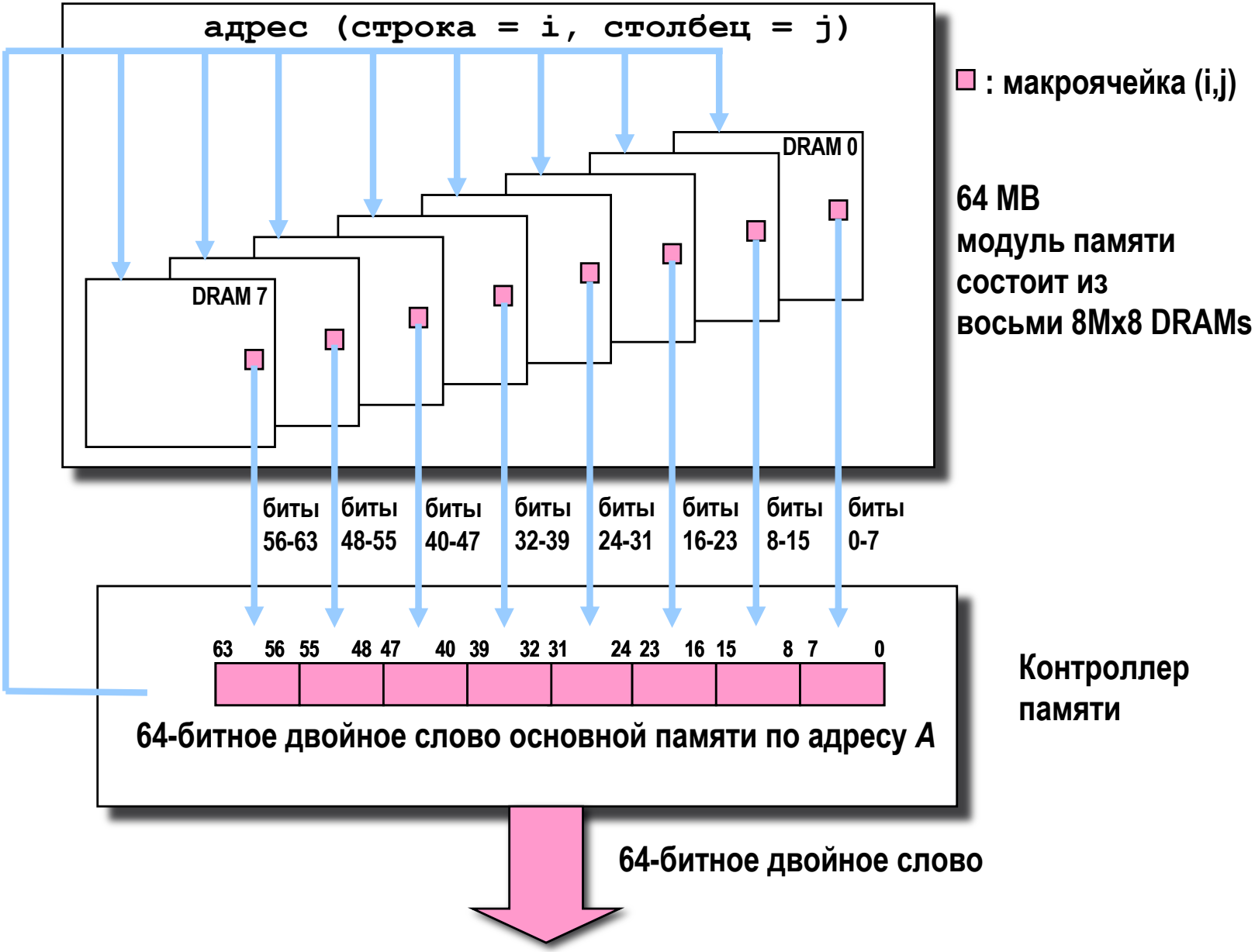
# Чтение макроячейки DRAM (2,1)

Шаг 2(a): Выбирается столбец 1 по сигналу «Column access strobe» (CAS)

Шаг 2(b): Макроячейка (2,1) копируется из буфера на шину данных, и затем в CPU.



# Модули памяти



# Усовершенствованные DRAM

- **Основная ячейка DRAM практически не менялась с момента изобретения в 1966.**
  - Коммерциализована Intel в 1970.
- **DRAM-массивы с улучшенной логикой ввода/вывода :**
  - Синхронная DRAM (SDRAM)
    - Использует синхронное управление вместо асинхронного
    - Повторно использует адрес строки (например: RAS, CAS, CAS, CAS)
  - Синхронная DRAM удвоенного быстродействия (DDR SDRAM)
    - Отсылка по одному проводу двух бит за цикл синхронизации
    - Различия в размере небольших буферов предвыборки:
      - DDR (2 бита), DDR2 (4 бита)…
    - Типично в 2010г. для для большинства серверов и настольных ПК
    - Intel Core i7 поддерживает только DDR3 SDRAM

# Тенденции развития памяти

SRAM

| Параметр | 1985 | 1990 | 1995 | 2000 | 2005 | 2010 | 2015 | *2015:1985* |
|---|---|---|---|---|---|---|---|---|
| $/МБ | 2,900 | 320 | 256 | 100 | 75 | 60 | *320* | *116* |
| доступ (нс) | 150 | 35 | 15 | 3 | 2 | 1.5 | *200* | *115* |

DRAM

| Параметр | 1985 | 1990 | 1995 | 2000 | 2005 | 2010 | 2015 | *2015:1985* |
|---|---|---|---|---|---|---|---|---|
| $/МБ | 880 | 100 | 30 | 1 | 0.1 | 0.06 | 0.02 | *44,000* |
| доступ (нс) | 200 | 100 | 70 | 60 | 50 | 40 | 20 | *10* |
| тип. ёмкость (ГБ) | 0.256 | 4 | 16 | 64 | 2,000 | 8,000 | 16.000 | *62,500* |

Диск

| Параметр | 1985 | 1990 | 1995 | 2000 | 2005 | 2010 | 2015 | *2015:1985* |
|---|---|---|---|---|---|---|---|---|
| $/ГБ | 100,000 | 8,000 | 300 | 10 | 5 | 0.3 | 0.03 | *3,333,333* |
| доступ (мс) | 75 | 28 | 10 | 8 | *5* | *3* | *3* | *25* |
| тип. ёмкость (ГБ) | 0.01 | 0.16 | 1 | 20 | 160 | 1,500 | 3,000 | *300,000* |

# Тактовая частота ЦП

**Переломная точка в компьютерной истории. Столкновение со “стеной энергопотребления”**

| | 1980 | 1990 | 1995 | 2000 | 2003 | 2005 | 2010 | *2010:1980* |
|---|---|---|---|---|---|---|---|---|
| ЦП | 8080 | 386 | Pentium | P-III | P-4 | Core 2 | Core i7 | --- |
| Тактовая ч-та (МГц) | 1 | 20 | 150 | 600 | 3300 | 2000 | 2500 | 2500 |
| Время такта (нс) | 1000 | 50 | 6 | 1.6 | 0.3 | 0.50 | 0.4 | 2500 |
| Ядер | 1 | 1 | 1 | 1 | 1 | 2 | 4 | 4 |
| Эффект. время цикла (нс) | 1000 | 50 | 6 | 1.6 | 0.3 | 0.25 | 0.1 | 10,000 |