© 2015 МГУ/ВМК/СП

Лекция 0x15

18 апреля

© 2015 МГУ/ВМК/СП

Примеры шин (3/3)

- AGP
  - Непосредственный доступ к оперативной памяти
- USB
  - Топология – звезда, до 127 устройств, разветвители, оконечные точки (15/15 + 1/1 управляющие).
  - 4 типа передач: управляющие, поточные, прерывания, изохронные
  - Open Host Controller Interface
    - Сам контроллер является PCI-устройством
- Serial ATA
  - 7 линий, 2 – прием, 2 – передача.
  - 1.5, 3, 6 GBit/s
  - Возможность горячей замены

© 2015 МГУ/ВМК/СП

Внутреннее устройство HDD

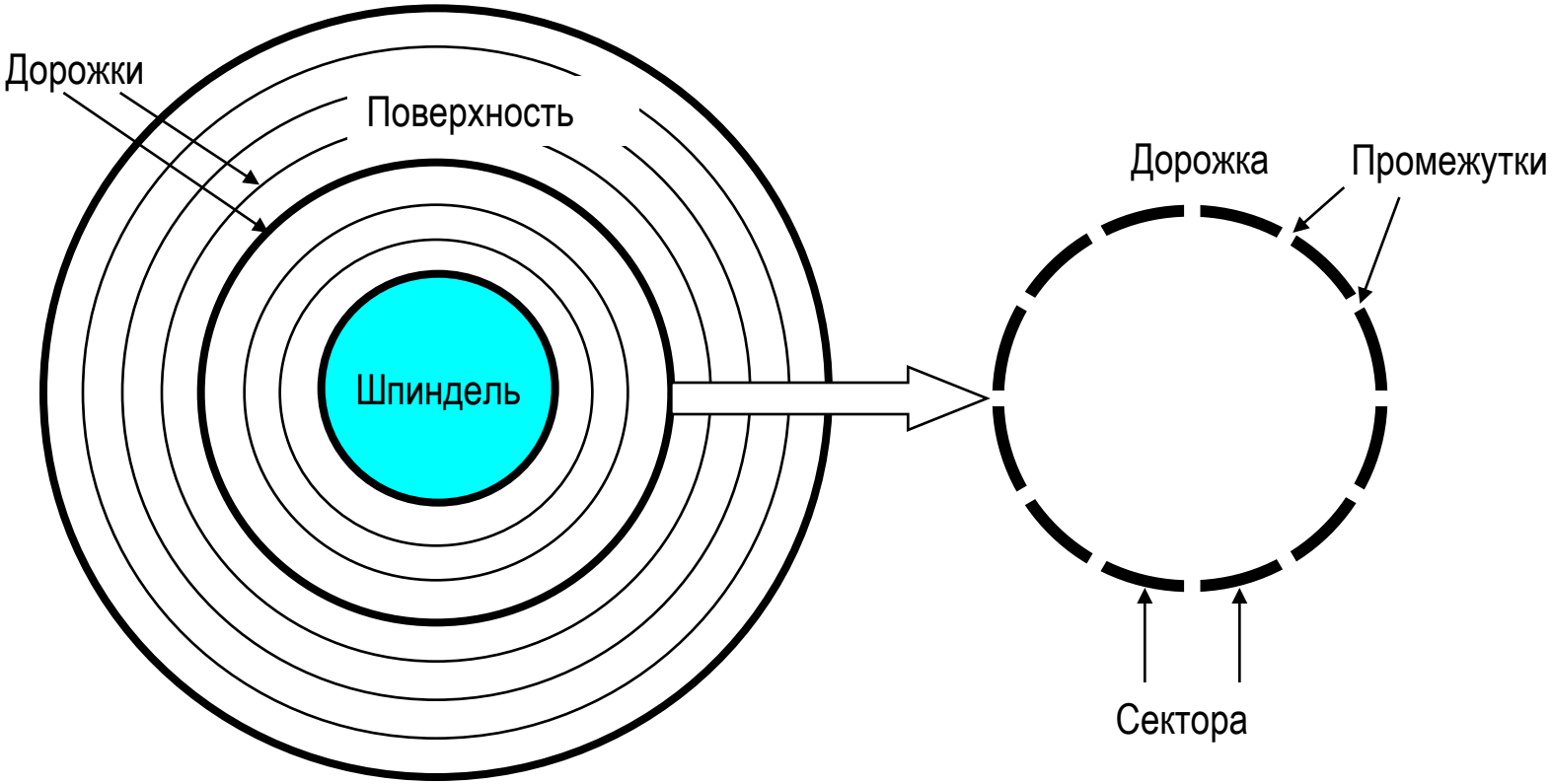
+ различная электроника, включая процессор и память

Изображение является собственностью Seagate Technology

© 2015 МГУ/ВМК/СП

Геометрия диска

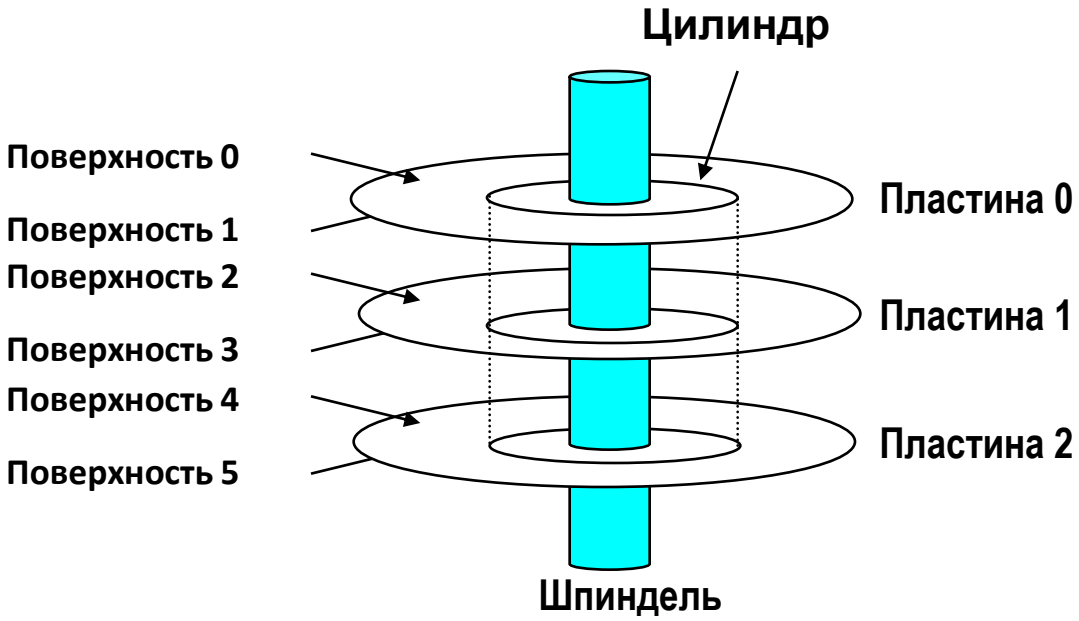
- Диск состоит из пластин, каждая обладает двумя поверхностями.
- Каждая поверхность состоит из концентрических кругов, называемых дорожки.
- Каждая дорожка состоит из секторов, разделенных промежутками.



© 2015 МГУ/ВМК/СП

Геометрия диска (несколько пластин)

- Равноудаленные от шпинделя дорожки образуют цилиндр.



© 2015 МГУ/ВМК/СП

Емкость диска

- Емкость: максимальное количество сохраняемых бит.
  - Производители выражают емкость в «необычных» гигабайтах, 1 ГБ = 10^9 байтам.
- Емкость определяется следующими технологическими факторами:
  - Плотность записи / линейная плотность (биты/дюймы – BPI): сколько битов может быть размещено на одном дюйме дорожки.
  - Трековая плотность (треки/дюйм – TPI): сколько треков может быть размещено на одном дюйме радиуса.
  - Поверхностная плотность (биты/дюймы^2): произведение линейной плотности на трековую плотность.
- Современные диски группируют дорожки в несколько зон записи
  - Каждая дорожка в зоне состоит из одного и того же количества секторов, определяемого длиной самой короткой дорожки.
  - У каждой зоны различное количество дорожек/секторов

© 2015 МГУ/ВМК/СП

Вычисление емкости диска

Емкость = (#байт/сектор) x (среднее # сектор/дорожка) x (# дорожка/поверхность) x (# поверхность/пластина) x (# пластина/диск)

Пример:

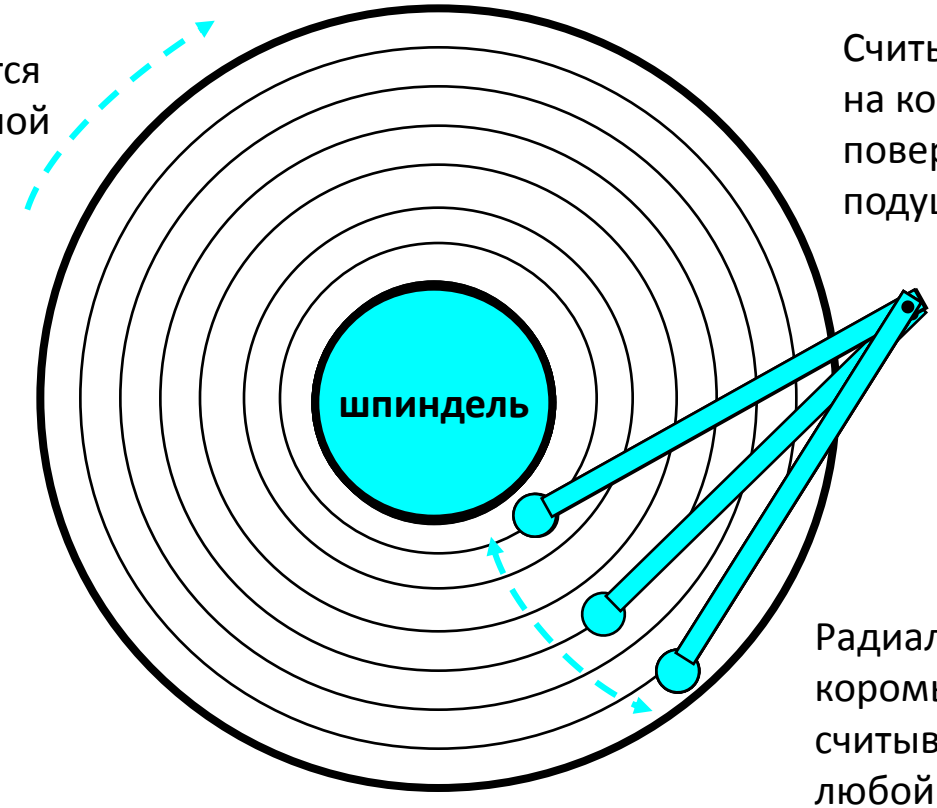
- 512 байт/сектор
- 300 сектор/дорожка (в среднем)
- 20,000 дорожка/поверхность
- 2 поверхность/пластина
- 5 пластина/диск

Емкость = 512 x 300 x 20000 x 2 x 5 = 30,720,000,000 = 30.72ГБ

© 2015 МГУ/ВМК/СП

Работа с диском (одна пластина)

Поверхность диска вращается с фиксированной скоростью.

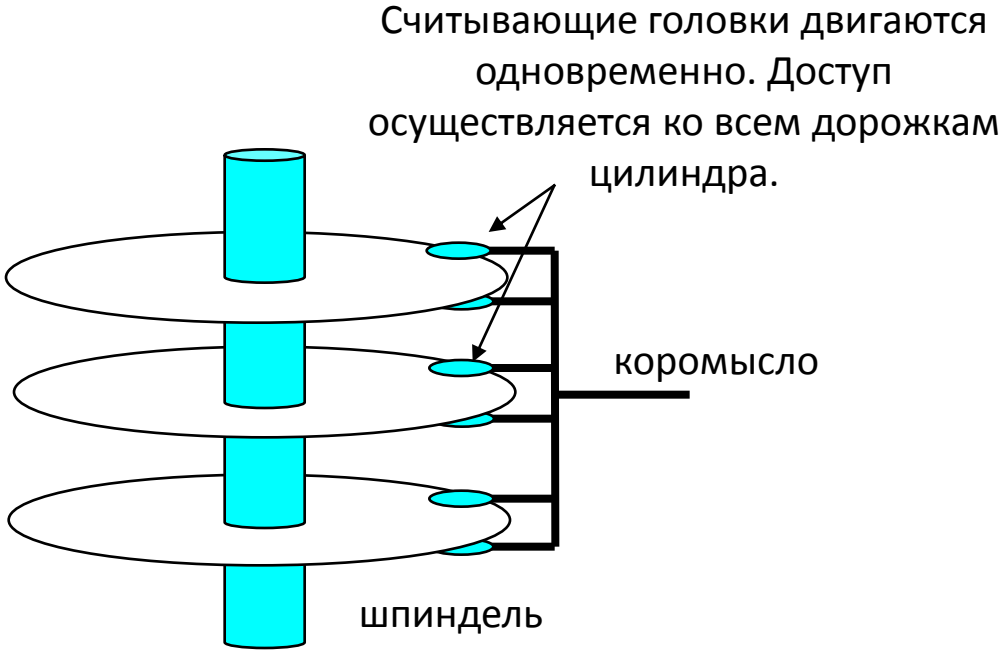


Считывающая головка закреплена на конце коромысла и парит над поверхностью на тонкой воздушной подушке.

Радиально перемещаясь, коромысло может выставить считывающую головку над любой дорожкой.

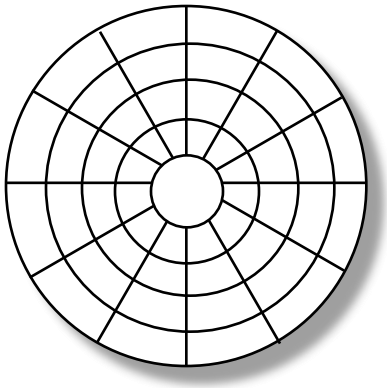
© 2015 МГУ/ВМК/СП

Работа с диском (несколько пластин)



© 2015 МГУ/ВМК/СП

Структура диска – вид сверху на одну пластину

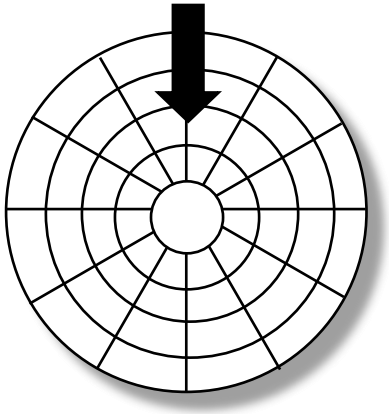


Поверхность разбита на дорожки

Дорожки разделены на сектора

© 2015 МГУ/ВМК/СП

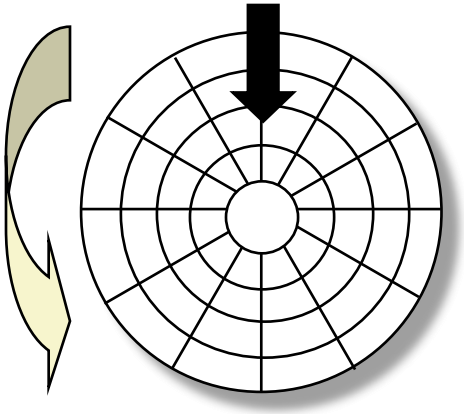
Доступ к диску



Считывающая головка в указанной позиции над диском

© 2015 МГУ/ВМК/СП

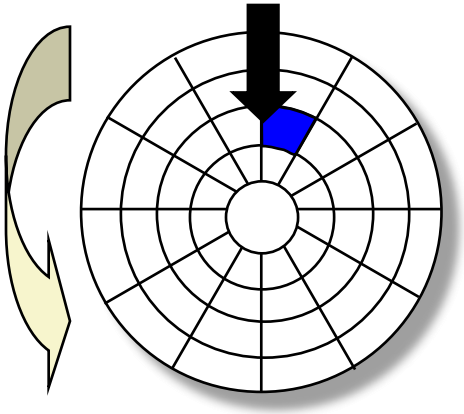
Доступ к диску



Направление вращения – против часовой стрелки

© 2015 МГУ/ВМК/СП

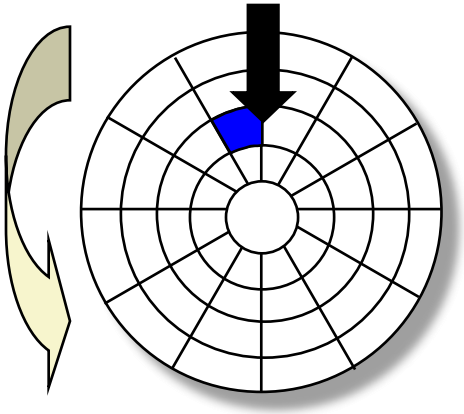
Доступ к диску – Чтение



Перед чтением синего сектора

© 2015 МГУ/ВМК/СП

Доступ к диску – Чтение

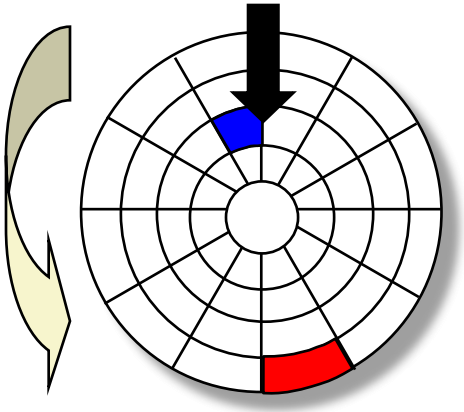


Синий сектор считан

После чтения синего сектора

© 2015 МГУ/ВМК/СП

Доступ к диску – Чтение

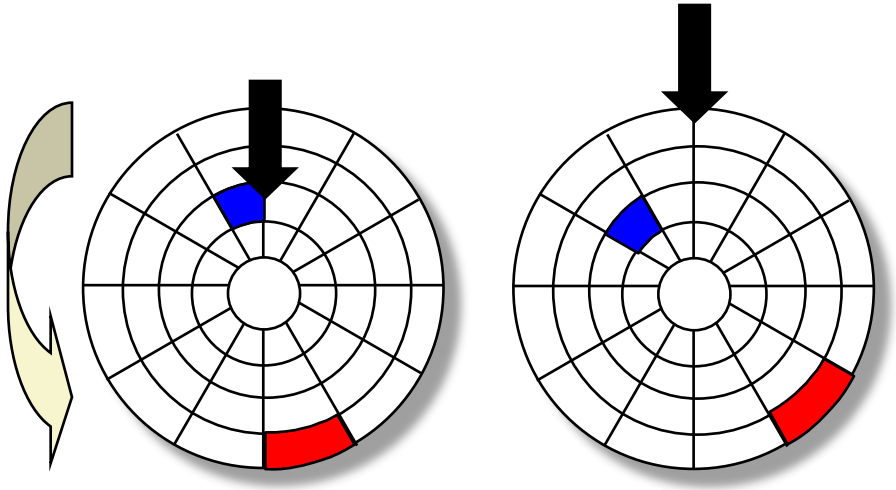


Синий сектор считан

Поступил запрос на чтение красного сектора

© 2015 МГУ/ВМК/СП

Доступ к диску – Поиск



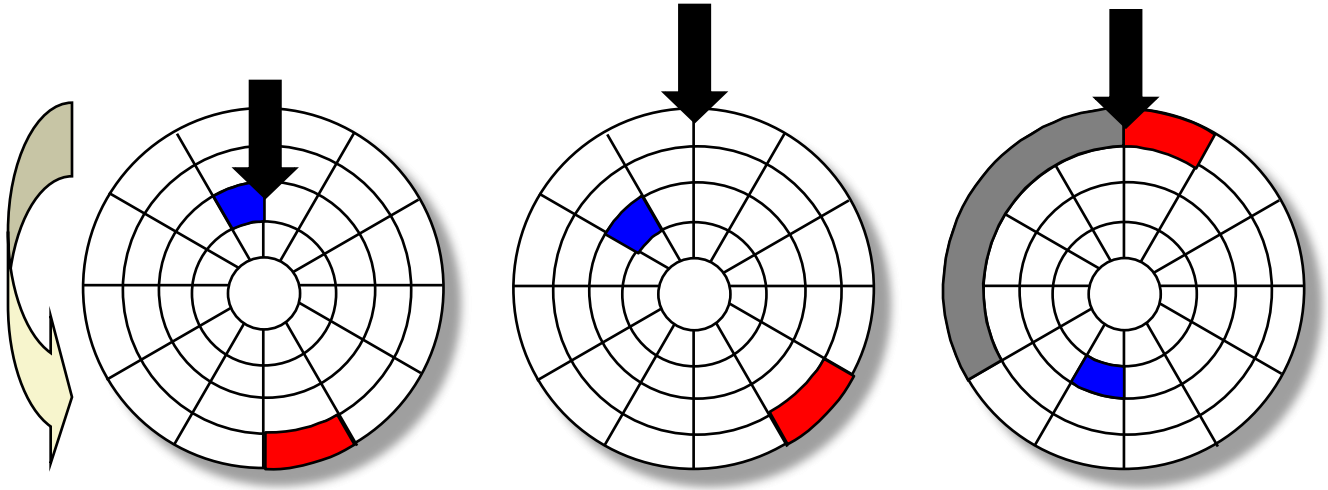
Синий сектор считан

Поиск Красного сектора

Ищем дорожку на которой расположен красный сектор

© 2015 МГУ/ВМК/СП

Доступ к диску – временная задержка из-за вращения

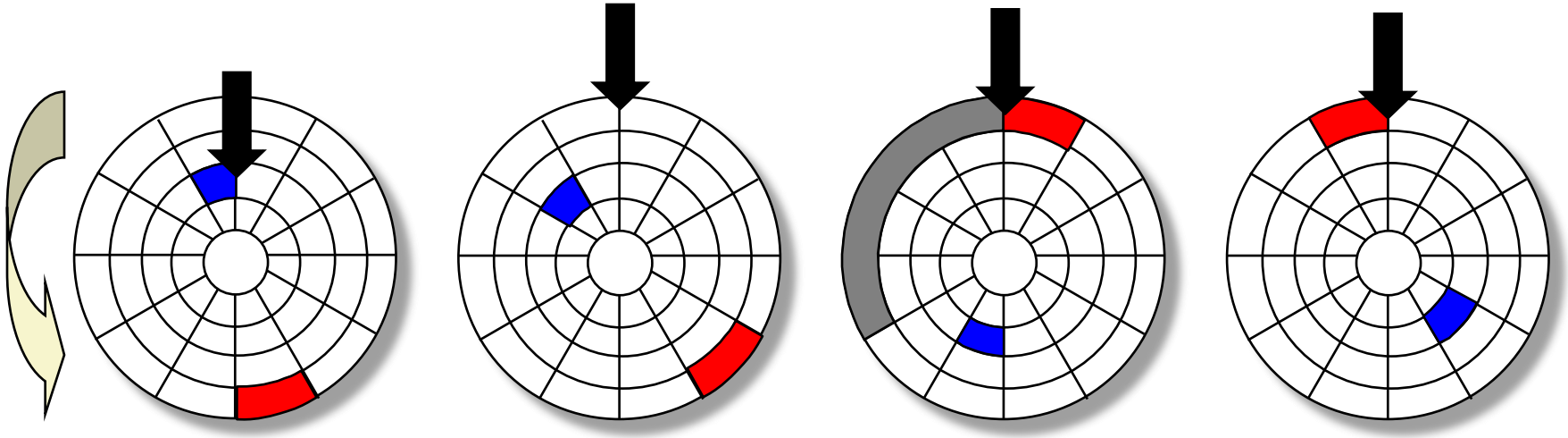


Синий сектор считан | **Поиск Красного сектора** | **Латентность вращения**

Вынужденное ожидание того момента, когда красный сектор достигнет считывающей головки

© 2015 МГУ/ВМК/СП

Доступ к диску – Чтение

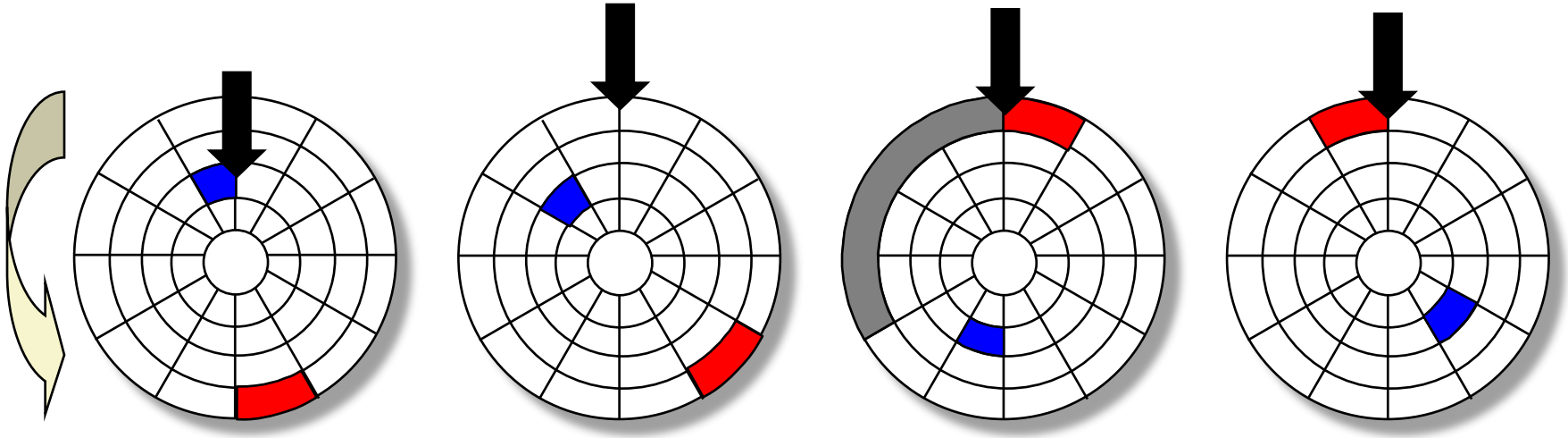


Синий сектор считан | **Поиск Красного сектора** | **Латентность вращения** | **Красный сектор считан**

Чтение красного сектора завершено

© 2015 МГУ/ВМК/СП

Доступ к диску – распределение времени



Синий сектор считан

Передача данных

Поиск Красного сектора

Поиск

Латентность вращения

Латентность вращения

Красный сектор считан

Передача данных

© 2015 МГУ/ВМК/СП

Время доступа к диску

- $T_{доступа} = T_{ср.\ поиск} + T_{ср.\ вращения} + T_{ср.\ передача}$
- Время поиска ($T_{ср.\ поиск}$)
  - Время, требуемое для перемещения считывающей головки в цилиндр, содержащий требуемый сектор.
  - Как правило $T_{ср.\ поиск}$ занимает 3—9 мс.
- Латентность вращения ($T_{ср.\ вращения}$)
  - Время ожидания момента, когда первый бит запрашиваемого сектора достигнет считывающей головки.
  - $T_{ср.\ вращения}$ = 1/2 x 1/RPM x 60 с / 1 мин
  - Типичная скорость вращения – 7200 RPM. $T_{ср.\ вращения} \approx$ 4 мс.
- Время передачи ($T_{ср.\ передача}$)
  - Время чтения содержимого сектора.
  - $T_{ср.\ передача}$ = 1/RPM x 1/(ср. # секторов на дорожке) x 60 с./1 мин.

© 2015 МГУ/ВМК/СП

Пример оценки времени доступа

- Исходные характеристики:
  - Скорость вращения = 7,200 RPM
  - Среднее время поиска = 9 мс.
  - Среднее # секторов на дорожке = 400.
- Оцениваем слагаемые и общую сумму:
  - $T_{ср.\ вращения}$ = 1/2 x (60 с/7200 RPM) x 1000 мс = 4 мс.
  - $T_{ср.\ передача}$ = 60/7200 RPM x 1/400 с/дорожка x 1000 мс = 0.02 мс
  - $T_{доступа}$ = 9 мс + 4 мс + 0.02 мс
- Выводы:
  - Время передачи существенно меньше остальных слагаемых.
  - Считать первый бит из сектора – «дорогая» операция, считывание остальных битов – «дешево».
  - Время доступа SRAM ≈ 4 нс для двойного слова, DRAM ≈ 60 нс
    - Диск медленнее SRAM в 40,000 раз, и …
    - … в 2,500 раз, чем DRAM.

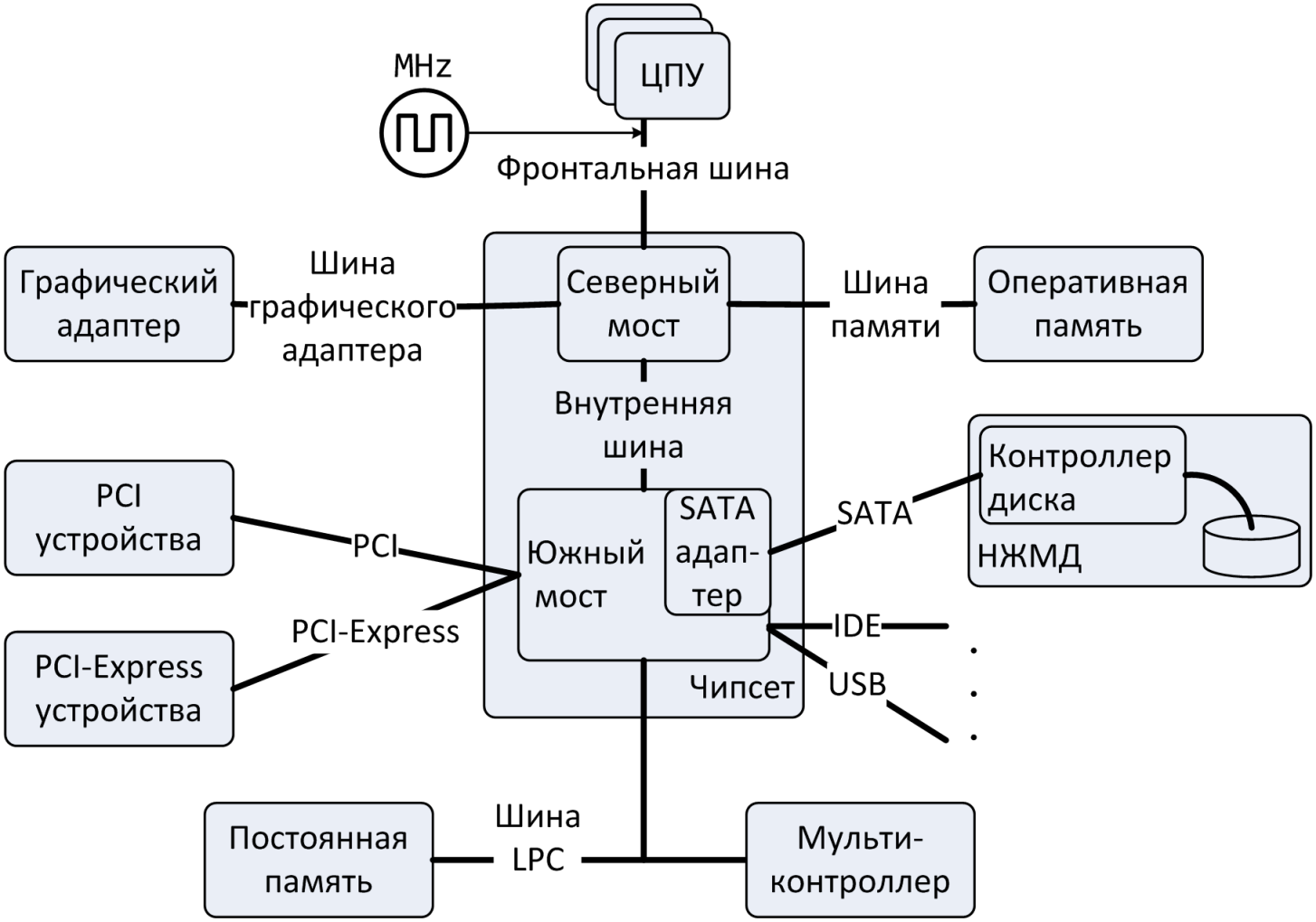
© 2015 МГУ/ВМК/СП

Логические блоки

- Более простой метод обращения к данным:
  - Сектора рассматриваются как последовательность логических блоков (0, 1, 2, ...)
- Соответствие между логическими блоками и (физическими) секторами
  - Отображение поддерживается аппаратурой + «прошивкой» – контроллером диска.
  - Номер логического блока → (поверхность, дорожка, сектор).
- Защита от выхода из строя отдельных цилиндров. Каждая зона записи содержит запасные цилиндры.
  - Размер диска после форматирования становится ощутимо меньше.

© 2015 МГУ/ВМК/СП

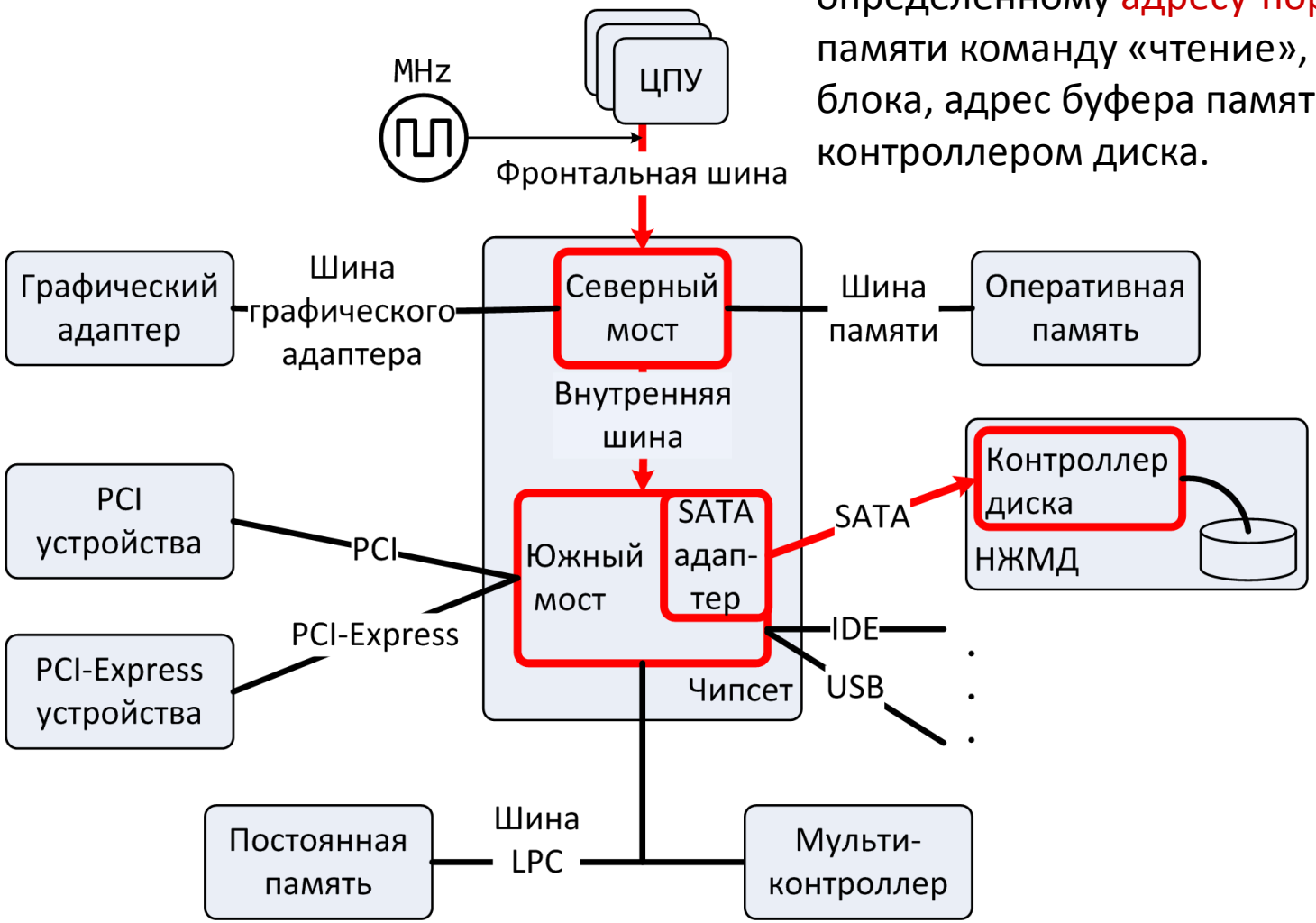
SATA: шина ввода/вывода



© 2015 МГУ/ВМК/СП

Чтение сектора (1)

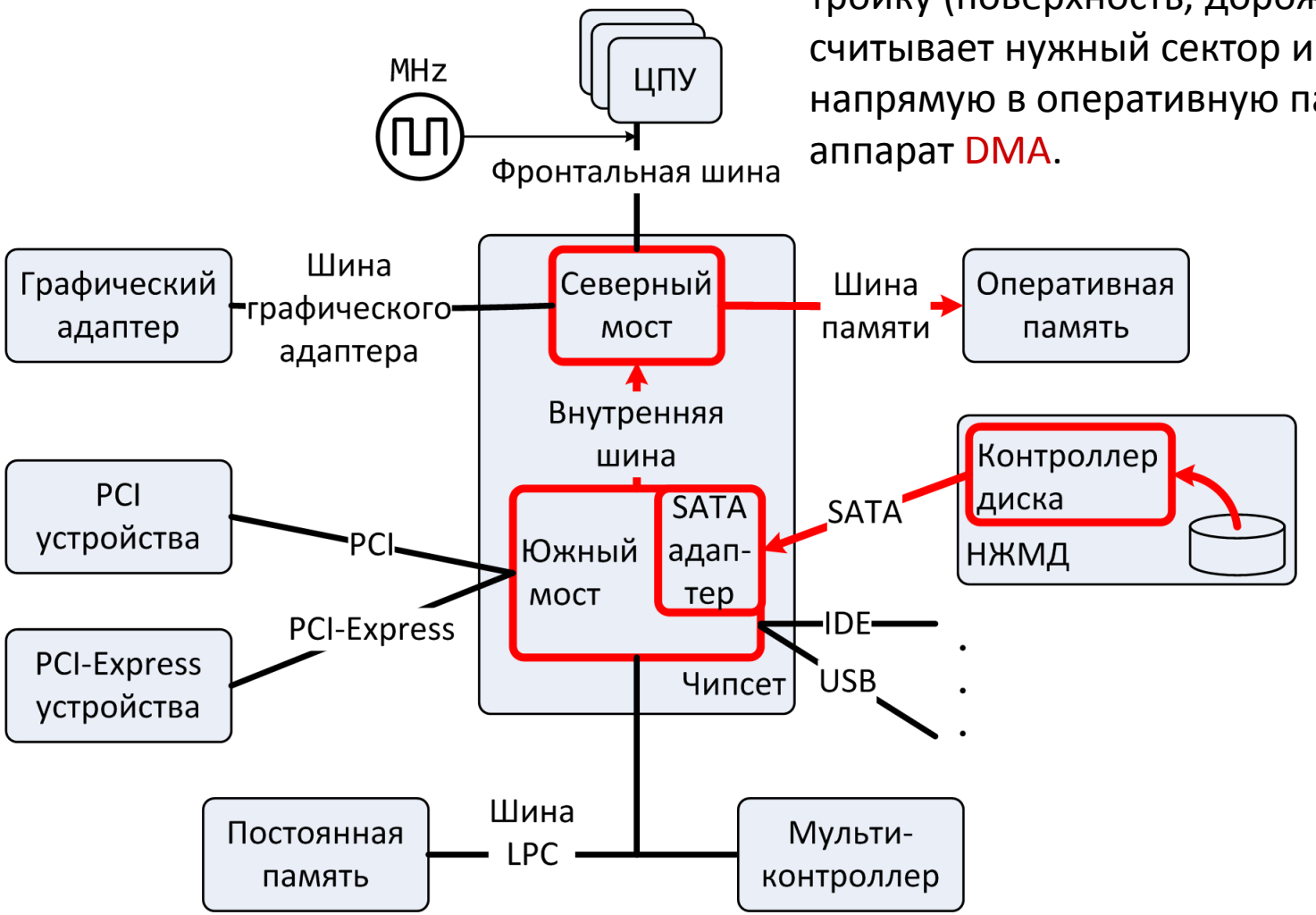
ЦПУ запускает чтение диска, записав по определенному адресу-порту оперативной памяти команду «чтение», номер логического блока, адрес буфера памяти. Порт связан с контроллером диска.



© 2015 МГУ/ВМК/СП

Чтение сектора (2)

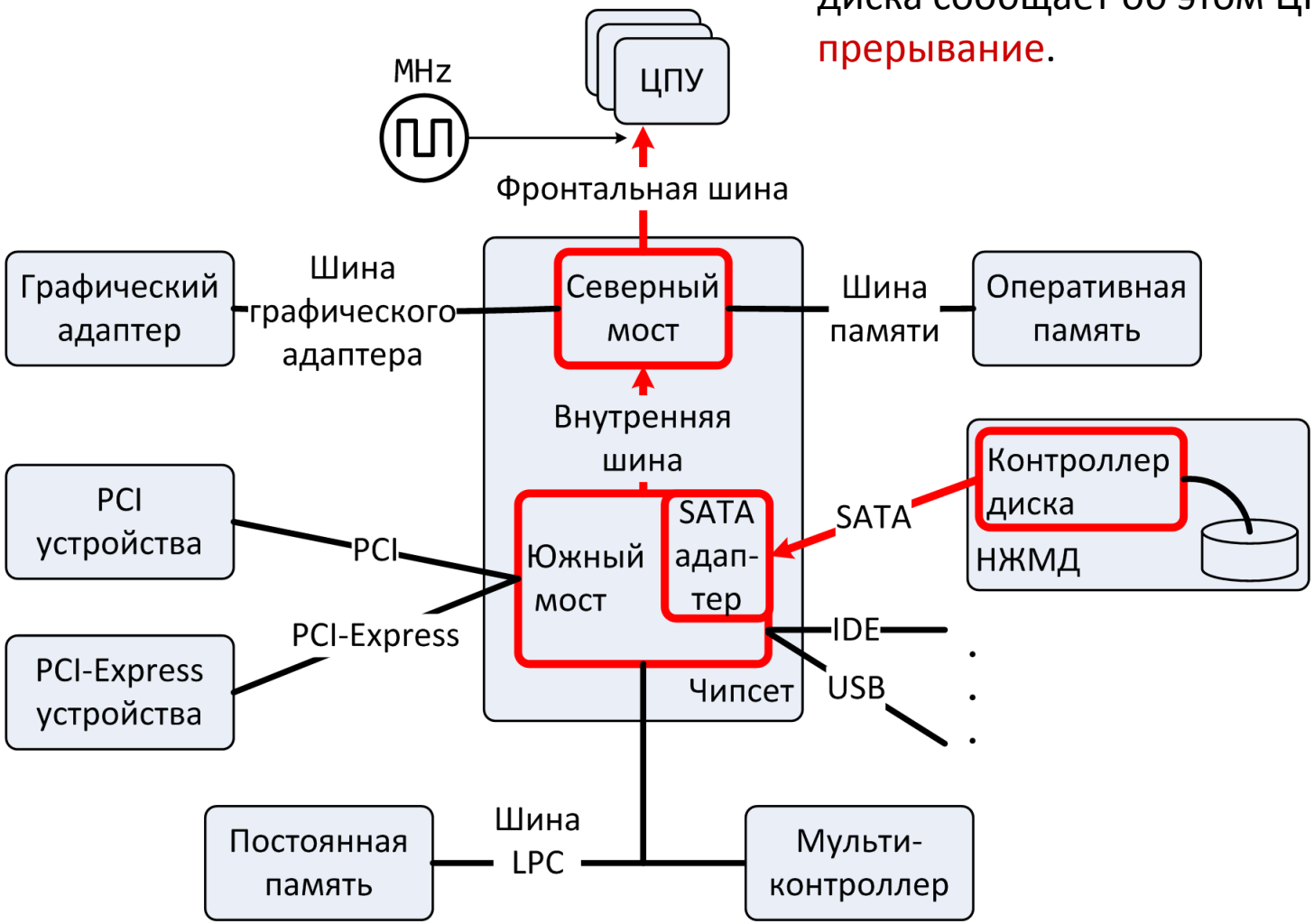
Контроллер диска преобразует номер блока в тройку (поверхность, дорожка, сектор), считывает нужный сектор и передает его напрямую в оперативную память, используя аппарат DMA.



© 2015 МГУ/ВМК/СП

Чтение сектора (3)

Когда DMA-трансферт закончен, контроллер диска сообщает об этом ЦПУ, вызывая прерывание.



© 2015 МГУ/ВМК/СП

Port IO vs. Memory Mapped IO

- Port IO: помимо пространства памяти вводится дополнительное пространство портов ввода/вывода
  - Работа с периферией: команды `in` и `out`
    - Все данные проходят через ЦПУ
  - Удобно при небольшом размере памяти

```
• IN EAX, DX ; AX, AL
    • EAX ← I/O[DX]
• OUT DX, EAX ; AX, AL
    • I/O[DX] ← EAX
```

- Memory Mapped IO: все управляющие регистры устройств отображаются на определенные адреса оперативной памяти
  - Требуется программировать контроллер памяти/северный мост
  - Перекрытая память не используется
  - Существенно более высокая производительность

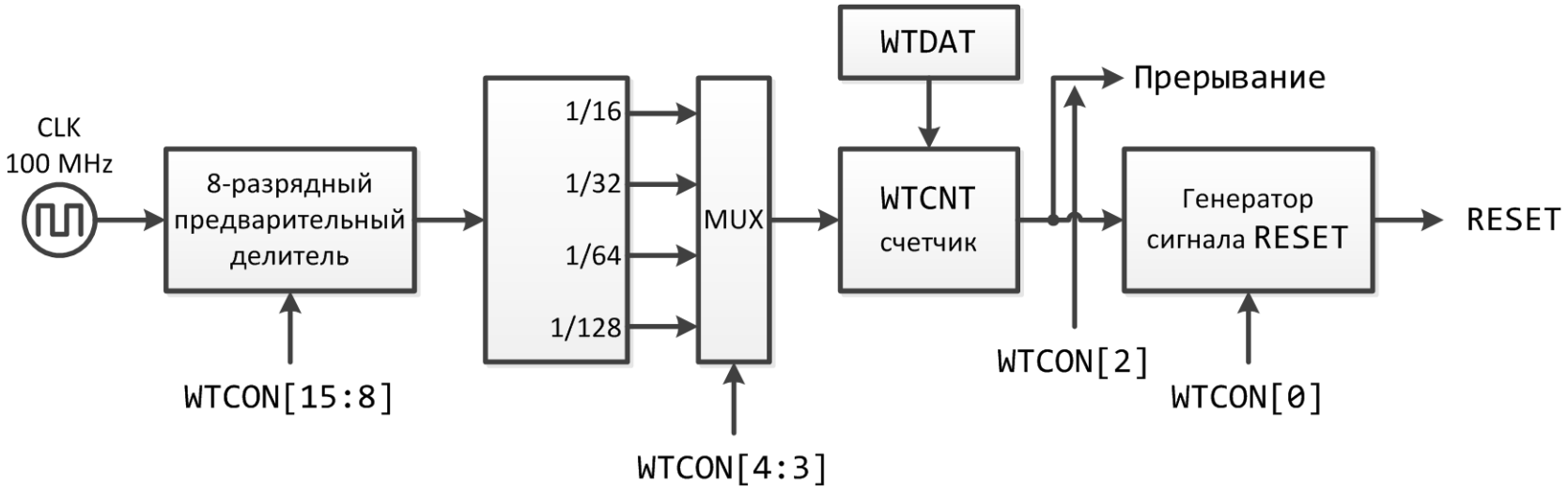
© 2015 МГУ/ВМК/СП

```
int device_driver_transmit_data(device *dev,
                                phys_addr_t *buffer,
                                size_t buf_len)
   {
           offset_t offset = device->offset;
           uint32_t status;
           time_t time = 0;
           outl(offset + BUFFER_REGISTER, buffer);
           outl(offset + BUFLEN_REGISTER, buf_len);
           outl(offset + COMMAND_REGISTER, COMMAND_TRANSMIT);
           do {
                   wait(WAIT_INTERVAL);
                   time += WAIT_INTERVAL;
                   inl(&status, STATUS_REGISTER);
                   if ((status & ERROR_MASK) || (time >= TIMEOUT))
                           goto error;
           } while (!(status & COMPLETE_MASK));
           return 0;
           error:
           return -1;
   }
```

```
void outl(int* m_reg, int val) {
   *m_reg = val;
}
```

© 2015 МГУ/ВМК/СП

Пример простейшего периферийного устройства: WatchDog @ SoC Exynos4210



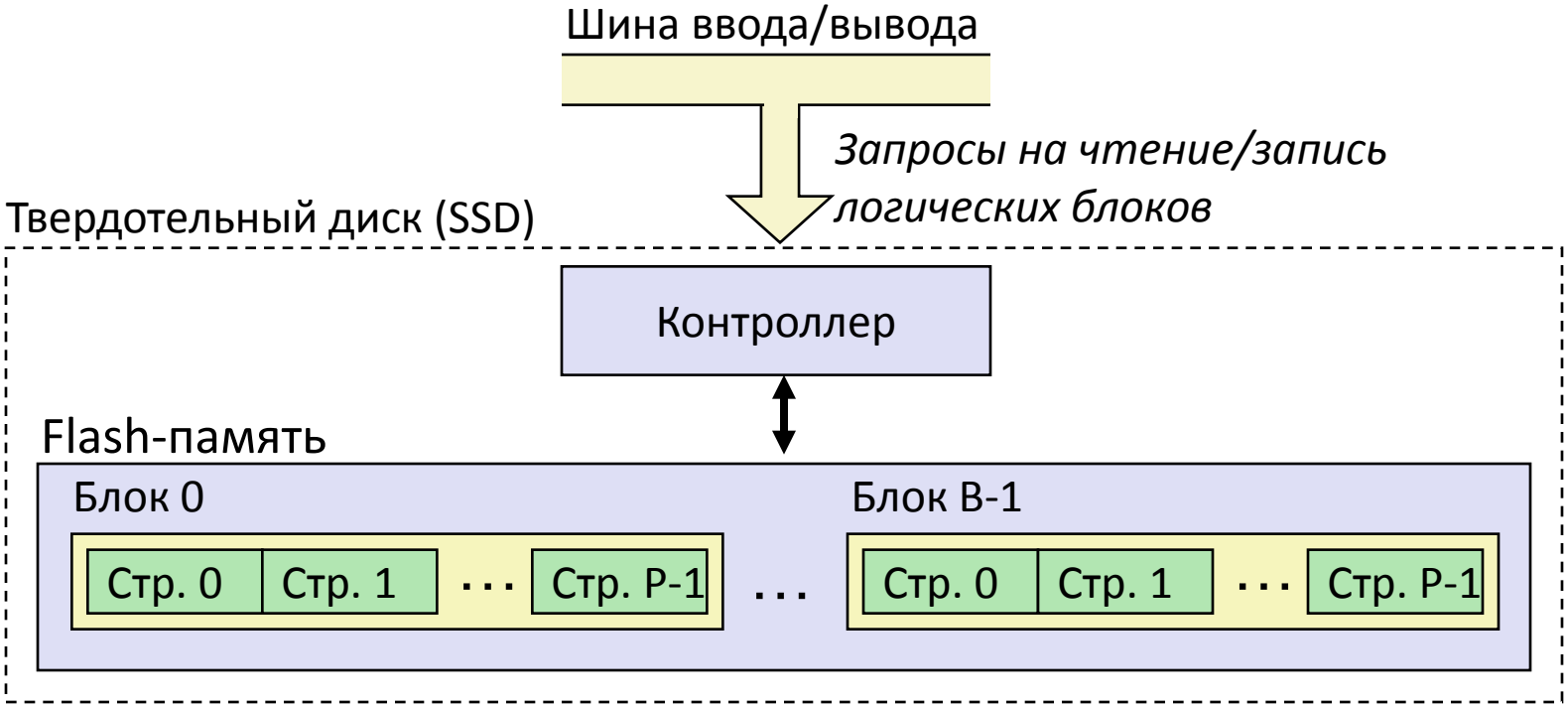
Базовый адрес `0x10060000`

Регистр	Смещение	Описание
WTCON	0x0	Управляющий регистр
WTDAT	0x4	Начальное значение счетчика
WTCNT	0x8	Счетчик таймера

Запуск устройства
```
WTCON[0] = 1
WTCON[5] = 1
```

© 2015 МГУ/ВМК/СП

Твердотельные диски (SSD)



- Размер
  - страницы: 512 – 4096 байт
  - блоки: 32 – 128 страниц
- Данные читаются/пишутся целыми страницами.
- Стирание данных – весь блок.
- Блок в некоторый момент вырабатывается
  - 100,000 перезаписей и более

© 2015 МГУ/ВМК/СП

Производительность SSD

Последовательное чтение	250 МБ/с	Последовательная запись	170 MB/s
Произвольное чтение	140 МБ/с	Произвольная запись	14 MB/s
Время доступа	30 мкс	Время старта записи 300 мкс	

- Причины малой скорости произвольной записи
  - Высокая длительность стирания блока (≈1 мс.)
  - Запись одной страницы вызывает копирование всех остальных страниц, расположенных в данном блоке
    - Выделить (найти) новый блок и стереть его содержимое
    - Записать станицу в новый блок
    - Скопировать остальные страницы из исходного блока

© 2015 МГУ/ВМК/СП

SSD vs. обычные диски

- Преимущества
  - Нет движущихся частей → более быстрые, меньшее энергопотребление, устойчивы к внешним воздействиям
- Недостатки
  - Ограниченное количество перезаписей
    - Контроллер стремится смягчить износ, распределяя перезаписи
    - Пример: гарантируют возможность произвольной записи 1 петабайта (10^{15} байт) данных до момента выработки
  - Высокая стоимость относительно HDD
    - В 2010 году, на два порядка
    - В 2015 году, на один порядок
- Применение
  - MP3 плееры, смартфоны, лэптопы
  - Появляются на персоналках и серверах

© 2015 МГУ/ВМК/СП

Тенденции в развитии запоминающих устройств

SRAM

Метрика	1980	1985	1990	1995	2000	2005	2010	*2010:1980*
$/МБ	19,200	2,900	320	256	100	75	60	*320*
t доступа (нс)	300	150	35	15	3	2	1.5	*200*

DRAM

Метрика	1980	1985	1990	1995	2000	2005	2010	*2010:1980*
$/МБ	8,000	880	100	30	1	0.1	0.06	*130,000*
t доступа (нс)	375	200	100	70	60	50	40	*9*
Размер (МБ)	0.064	0.256	4	16	64	2,000	8,000	*125,000*

HDD

Метрика	1980	1985	1990	1995	2000	2005	2010	*2010:1980*
$/МБ	500	100	8	0.30	0.01	0.005	0.0003	*1,600,000*
t доступа (мс)	87	75	28	10	8	*4*	*3*	*29*
Размер (МБ)	1	10	160	1,000	20,000	160,000	1,500,000	*1,500,000*

© 2015 МГУ/ВМК/СП

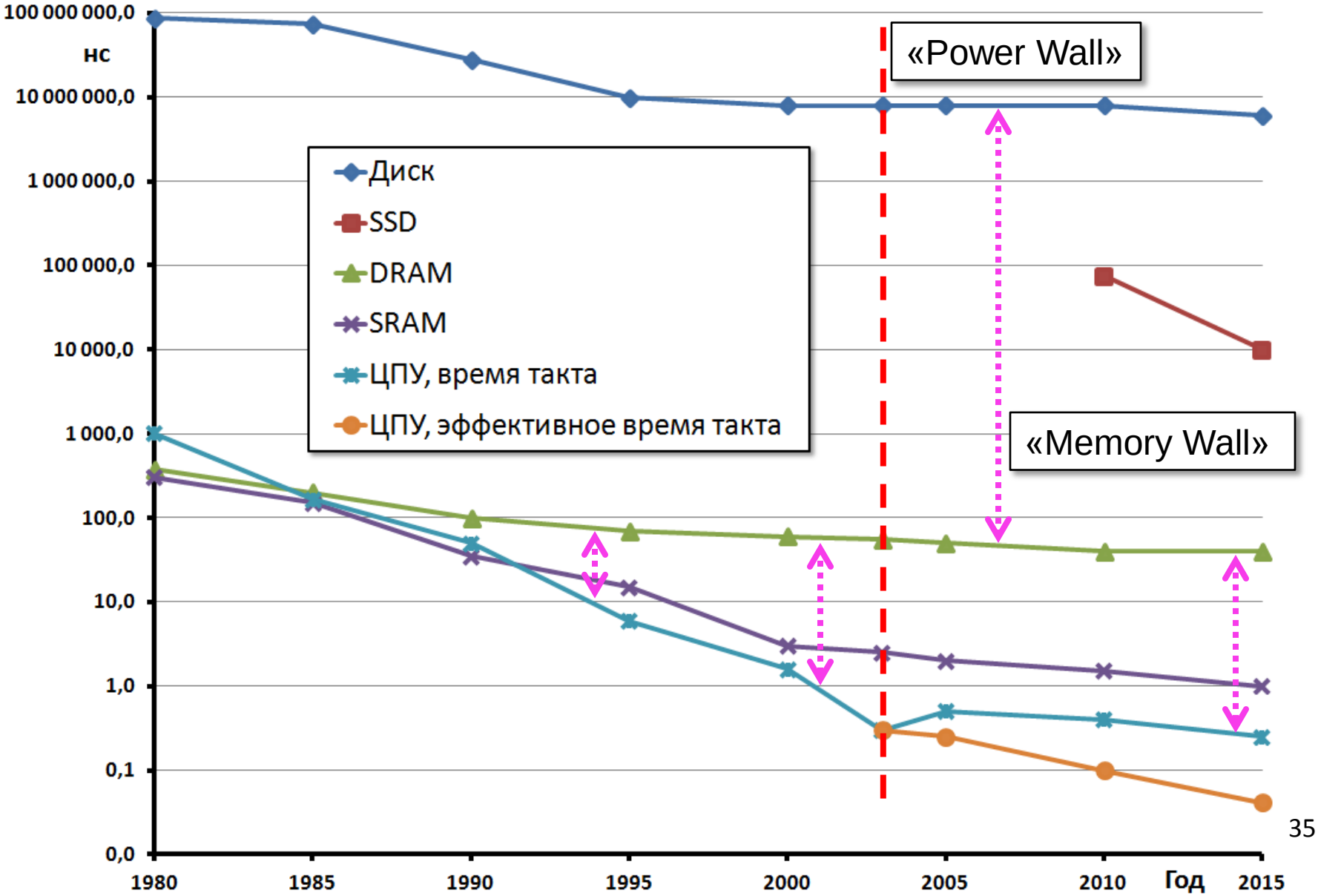
Частота ЦПУ

Разработчики аппаратуры столкнулись с “Power Wall”

	1980	1990	1995	2000	2003	2005	2010	2015	2015 ÷ 1980
ЦПУ	8086	80386	Pentium	P-III	P-4	Core 2	Core i7 Nehalem	Core i7 Haswell	
Частота (МГц)	1	20	150	600	3300	2000	2500	3000	3000
Длительность такта (нс.)	1000	50	6	1.6	0.3	0.5	0.4	0.25	3000
Количество ядер	1	1	1	1	1	2	4	8	8
Эффективная длительность такта (нс)	1000	50	6	1.6	0.3	0.25	0.1	0.04125	~24250

© 2015 МГУ/ВМК/СП

Разрыв между ЦПУ и памятью

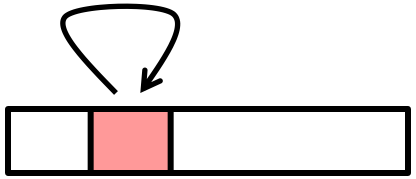


© 2015 МГУ/ВМК/СП

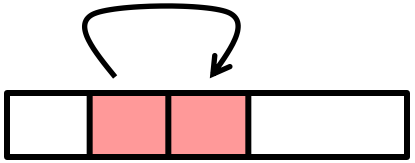
Локальность

- Основной принцип локальности: программа стремится использовать данные и инструкции с адресами близкими (либо точно такими же) к тем, которые использовались ранее.

- Временная локальность:
  - Повторные обращения



- Пространственная локальность:
  - В некоторый малый промежуток времени используются ячейки памяти с близкими адресами



© 2015 МГУ/ВМК/СП

Пример

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

- Выборка данных
  - Последовательные обращения к элементам массива. **Пространственная локальность**
  - Переменная `sum` используется на каждой итерации. **Временная локальность**
- Выборка инструкций
  - Последовательная выборка инструкции. **Пространственная локальность**
  - Повторное выполнение инструкций в цикле. **Временная локальность**

© 2015 МГУ/ВМК/СП

Оценка качества локальности

- Утверждение: способность беглым взглядом определить характер локальности кода является одним из необходимых навыков профессионального программиста.

- Вопрос: Достигается ли в функции `sum_array_rows` локальность обращений к массиву 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;
}
```

© 2015 МГУ/ВМК/СП

Еще один пример

- Вопрос: достигается ли в функции `sum_array_cols` локальность обращений к массиву 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;
}
```

© 2015 МГУ/ВМК/СП

Третий пример

- Вопрос: Как преобразовать гнездо циклов, что бы проход по 3-мерному массиву выполнялся с шагом 1 (и т.о. достичь пространственной локальности)?

```
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;
}
```

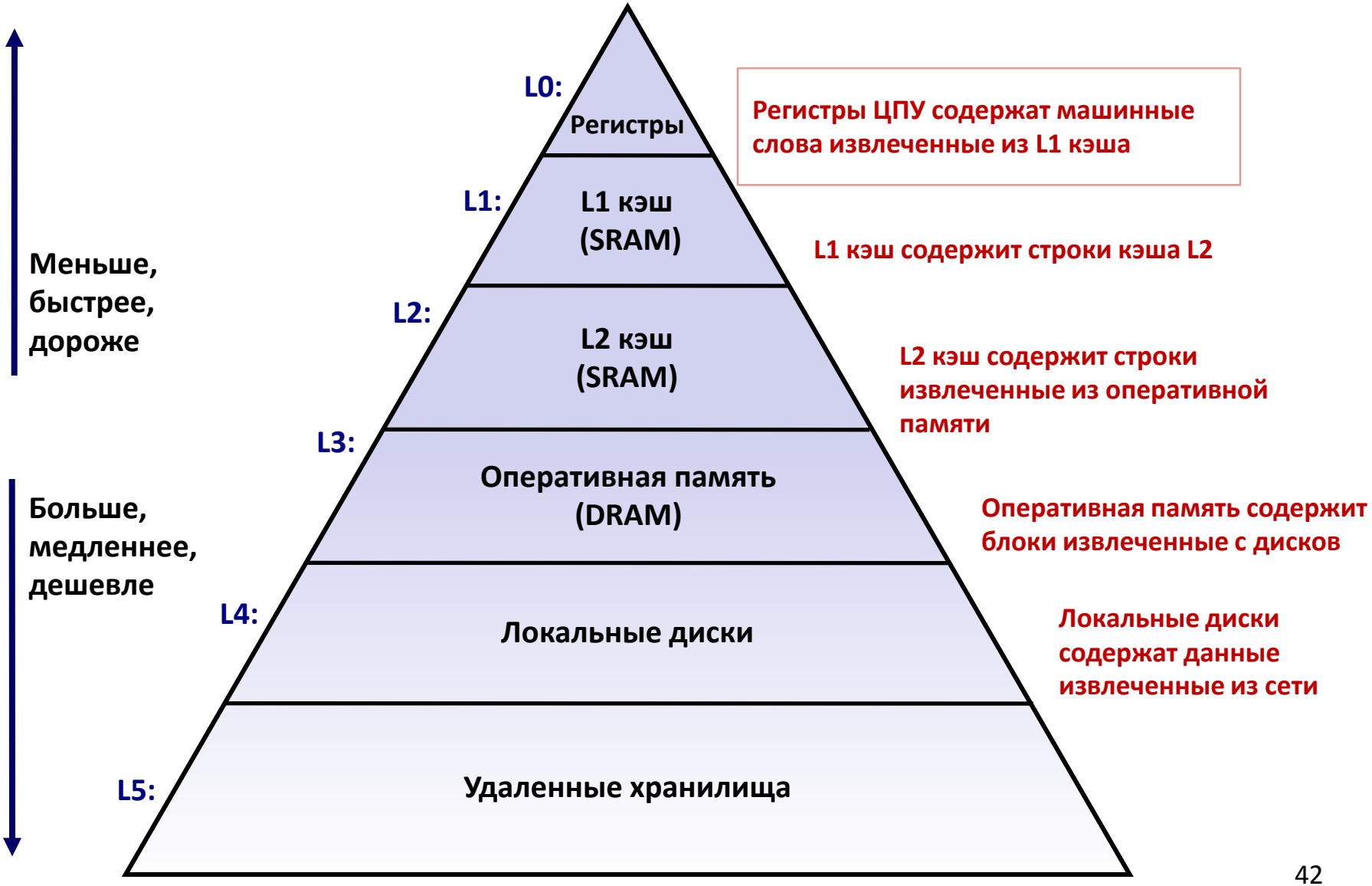
© 2015 МГУ/ВМК/СП

Иерархия памяти

- Ряд фундаментальных свойств аппаратуры и ПО:
  - Более быстрые устройства хранения стоят дороже, имеют меньший объем и потребляют больше энергии.
  - Разрыв в скорости работы между ЦПУ и оперативной памятью увеличивается.
  - В хорошо написанных программах демонстрируется хорошая локальность.
- Данные свойства дополняют друг друга и …
- … выводят на идею иерархической организации памяти.

© 2015 МГУ/ВМК/СП

Иерархия памяти



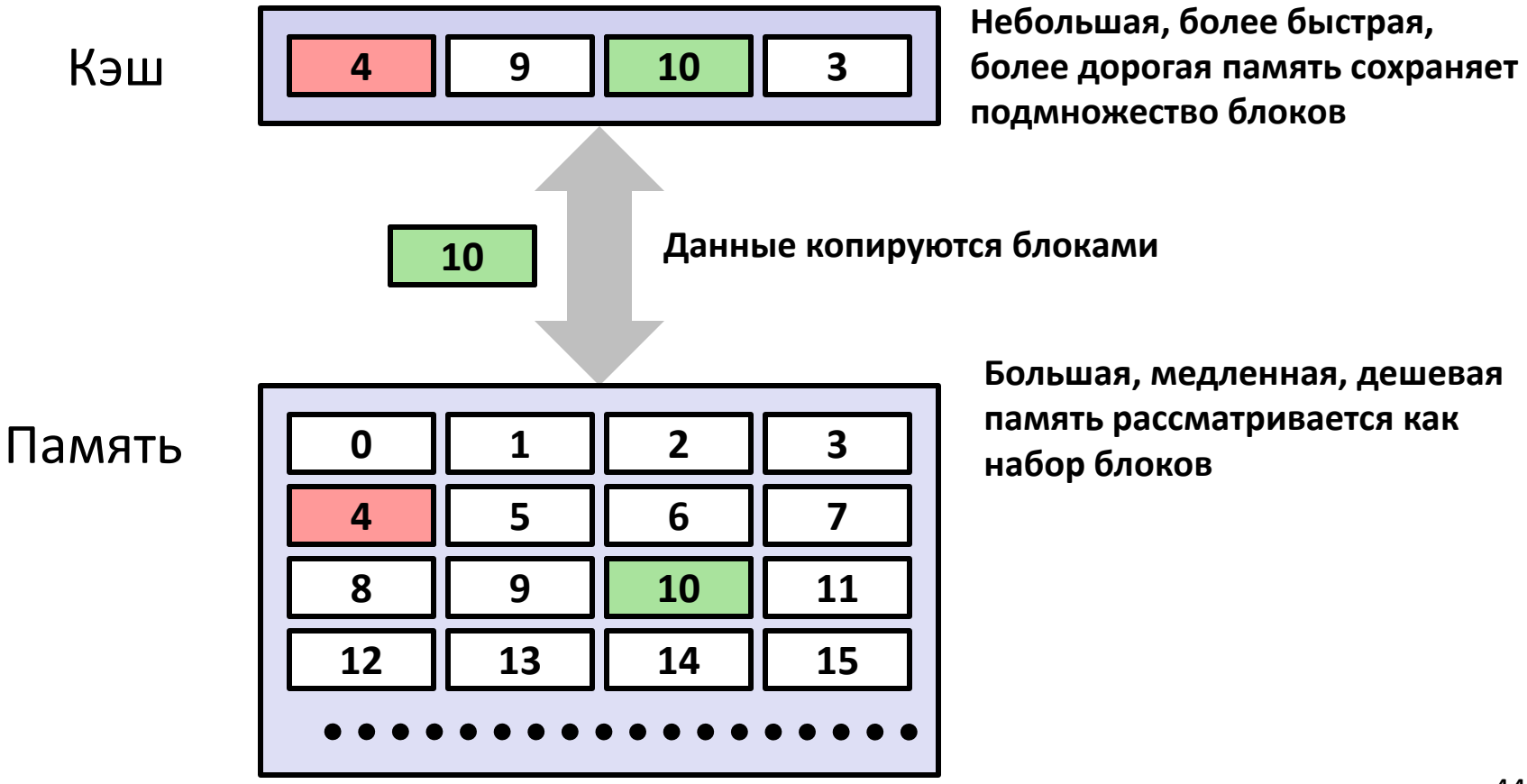
© 2015 МГУ/ВМК/СП

Кэш

- *Кэш:* меньшее объемом, но более быстрое устройство хранения, выступает в роли промежуточного хранилища для большего, но более медленного устройства.
- Основная идея иерархии памяти:
  - Для каждого k, более быстрое, меньшее устройство на уровне k выступает как кэш для большего, но более медленного устройства на уровне k+1.
- Почему это работает?
  - Из-за локальности программа обращается к данным на уровне k гораздо чаше чем к данным на уровне k+1.
  - Устройство уровня k+1 может быть более медленным → большего размера, более дешевым.
- *Результат:* Иерархическая память – большой объем данных, стоит сопоставимо с самой дешевой компонентой, работает с максимальной скоростью.

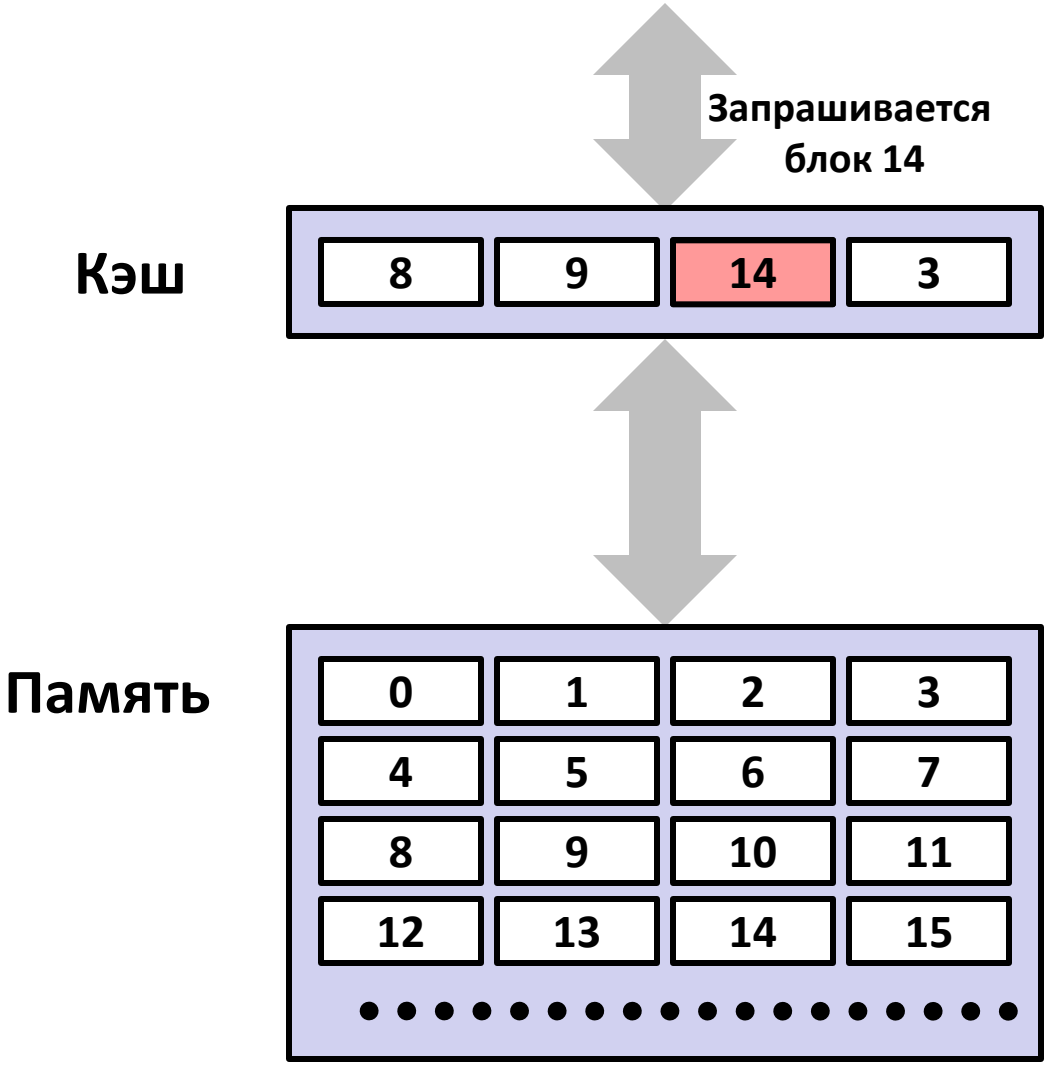
© 2015 МГУ/ВМК/СП

Основная идея кэша



© 2015 МГУ/ВМК/СП

Попадание в кэш



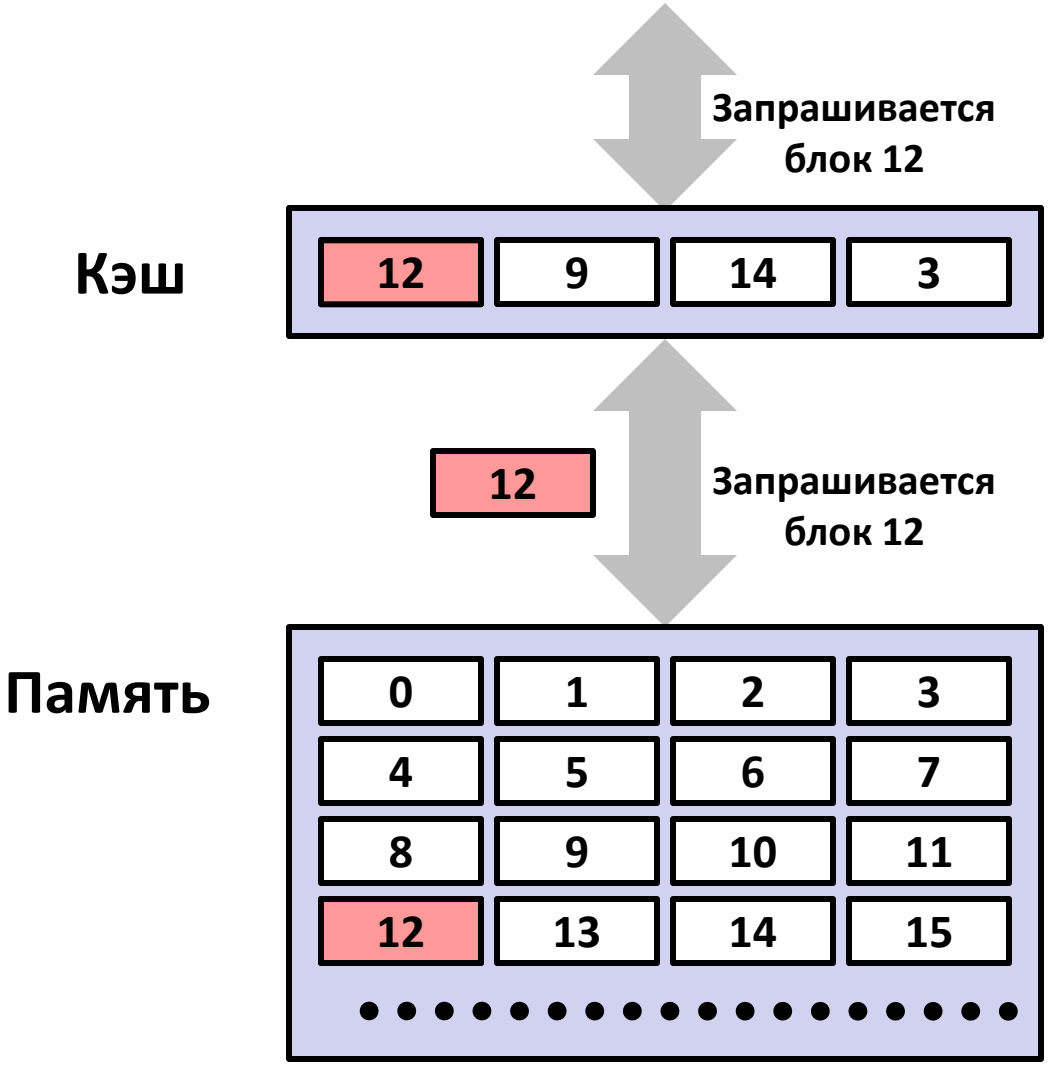
Запрашиваются данные из блока b

Блок b находится в кэше:

Попадание!

© 2015 МГУ/ВМК/СП

Промахи



Запрашиваются данные из блока b

Блока b нет в кэше: Промах!

Блок b извлекается из памяти

Блок b размещается в кэше

- Правила размещения определяют где будет находится блок b
- Правила замещения определяют какой блок будет исключен из кэша

© 2015 МГУ/ВМК/СП

Различные типы промахов

- Холодные (вынужденные) промахи
  - Причина – пустой кэш.
- Промахи из-за конфликтов
  - Количество мест размещения ограничено (может быть единственным)
    - Пример: блок i уровня k+1 размещается в блоке (i mod 4) на уровне k.
  - Промахи из-за конфликтов возникают когда несколько блоков размещаются на одном и том же месте.
    - Пример: запрос блоков 0, 8, 0, 8, 0, 8, ... Будет постоянно вызывать промахи.
- Промахи из-за нехватки емкости
  - Причина – используемых блоков (рабочее множество) больше, чем кэш может в себе вместить.

© 2015 МГУ/ВМК/СП

Примеры кэшей

Тип кэша	Что кэшируется?	Где кэшировано?	Задержка (такты)	Управление
Регистры	Слова по 4-8 байт	Ядро ЦПУ	0	Компилятор
L1 кэш	Блок 64 байта	На кристалле, L1	1	Аппаратура
L2 кэш	Блок 64 байта	На/вне кристалла, L2	10	Аппаратура
Буфер ввода/вывода	Части файлов	Оперативная память	100	ОС
Кэш диска	Сектора диска	Контроллер диска	100,000	Прошивка диска
Сетевой кэш	Части файлов	Локальный диск	10,000,000	AFS/NFS клиент
Кэш браузера	Веб-страницы	Локальный диск	10,000,000	Веб браузер
Веб-кэш	Веб-страницы	Диск на удаленном сервере	1,000,000,000	Веб-прокси сервер

© 2015 МГУ/ВМК/СП

Иерархическая память: промежуточные итоги

- Разрыв в скорости работы между ЦПУ и оперативной памятью продолжает увеличиваться.
- Хорошо написанные программы имеют хорошую локальность.
- Иерархическая организация памяти, основанная на кэшировании, позволяет эффективно бороться с разрывом в скорости с помощью локальности.