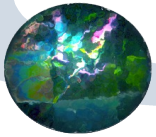
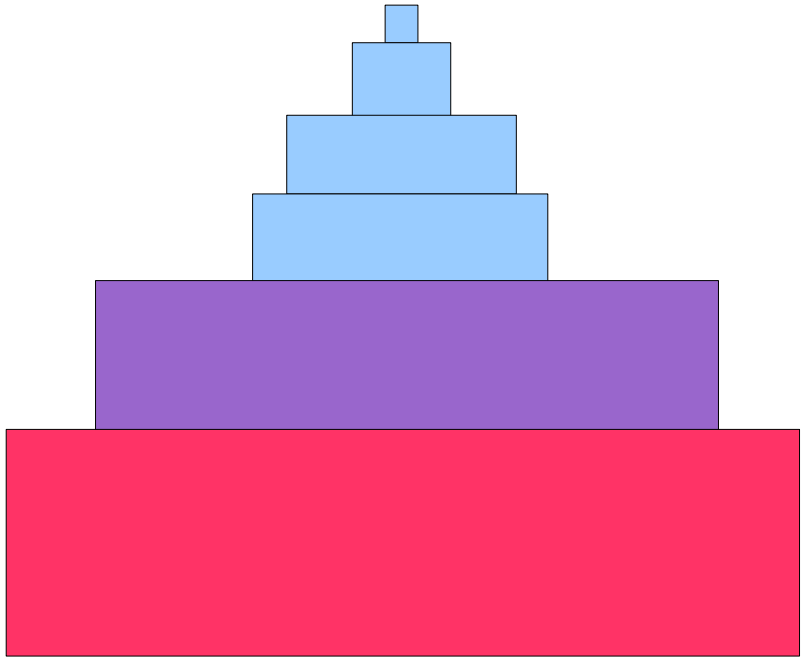


# *Иерархия памяти*

В.А.Савельев

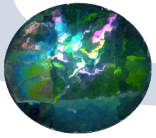
Факультет математики, механики и компьютерных наук

# *Что такое иерархия памяти?*



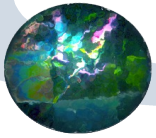
| | Скорость | Объем |
|---|---|---|
| Регистры | 0 | 1K |
| L1 Кэш | 2-3 | 16-32K |
| L2 Кэш | ~10 | 512K - 4M |
| L3 Кэш | ~10-25 | 2M-16M |
| Основная память | 30-100 | 64M-64ГБ |
| Вторичная память | ~$10^6$ | 200ГБ-2048ГБ |

Статическое ОЗУ  Динамическое ОЗУ  Магнитная запись



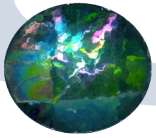
# *Статическая память*

- Значение бита сохраняется триггером — бистабильной схемой.
- Работает со скоростью переключения транзисторов в данной технологии СБИС
- Схема занимает довольно много места на кристалле — невозможна плотная упаковка
- Дорога
- Энергозависима



# *Динамическая память*

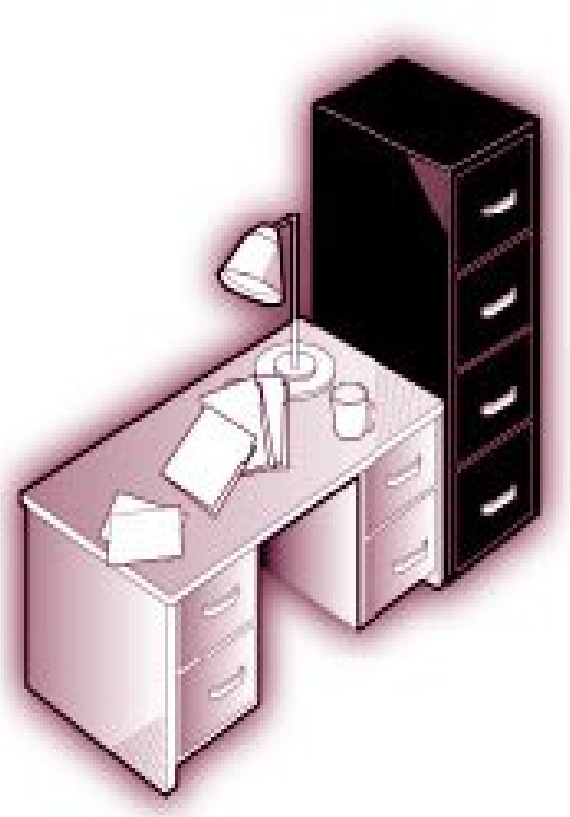
- Бит хранится в конденсаторе
- Конденсаторы могут быть эффективно упакованы на кристалле
- Требуются сложные схемы для доступа
- Требуется регулярная регенерация памяти (Время!)
- Относительно дешева
- Энергозависима



# *Магнитные и оптические диски*

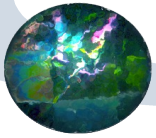
- Требует времени для перемагничивания
- Головка должна ***двигаться*** относительно магнитного носителя — то есть требуются механическая часть
- Очень большое время поиска информации
- Возможна очень плотная запись
- Дешевы
- Энергонезависима

# *Использование памяти*

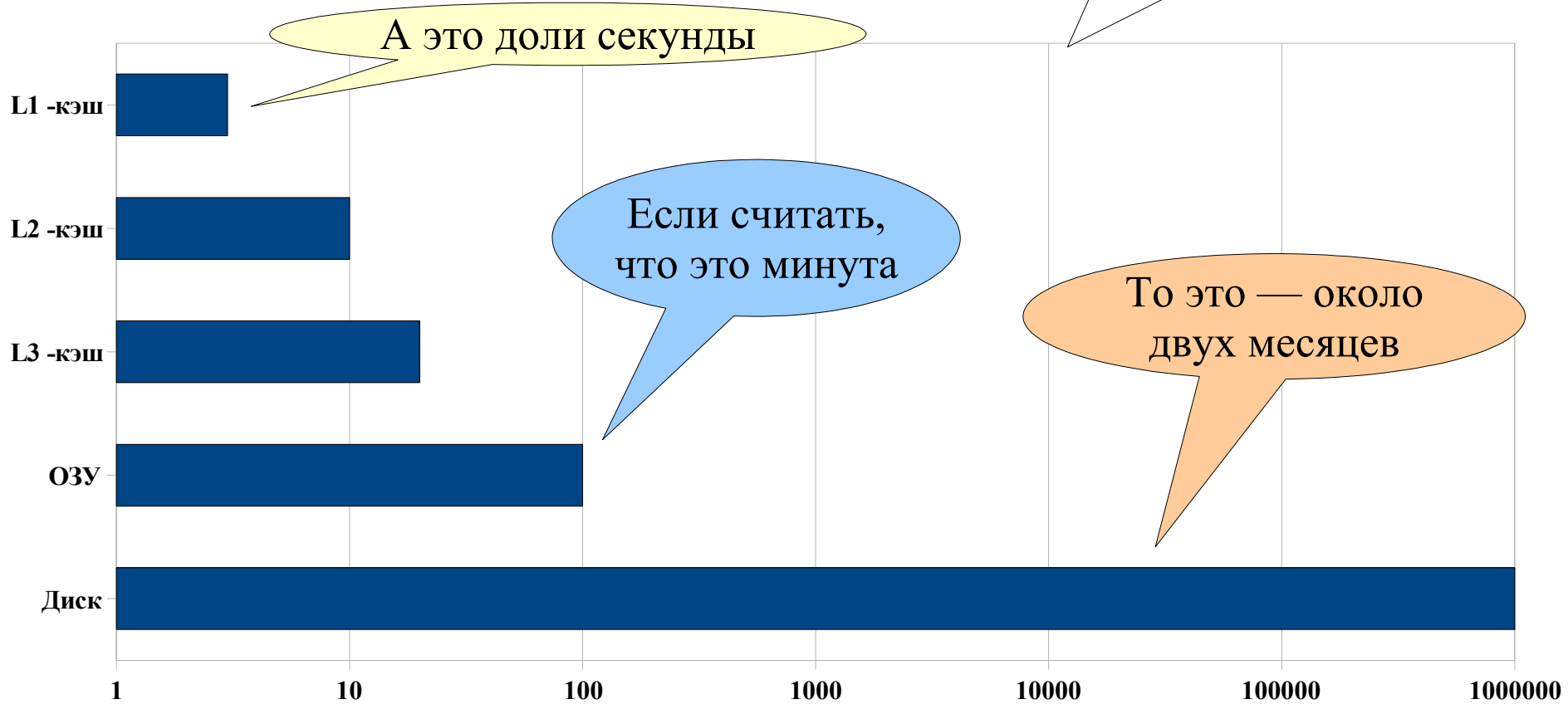


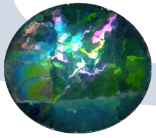
В современном компьютере оперативная или основная память — это рабочее пространство, как бы аналог рабочего стола.

А жёсткие диски — долговременное хранилище документов



# *Времена доступа*



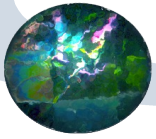


# *Времена доступа*

Да, соотношение времен именно таково.

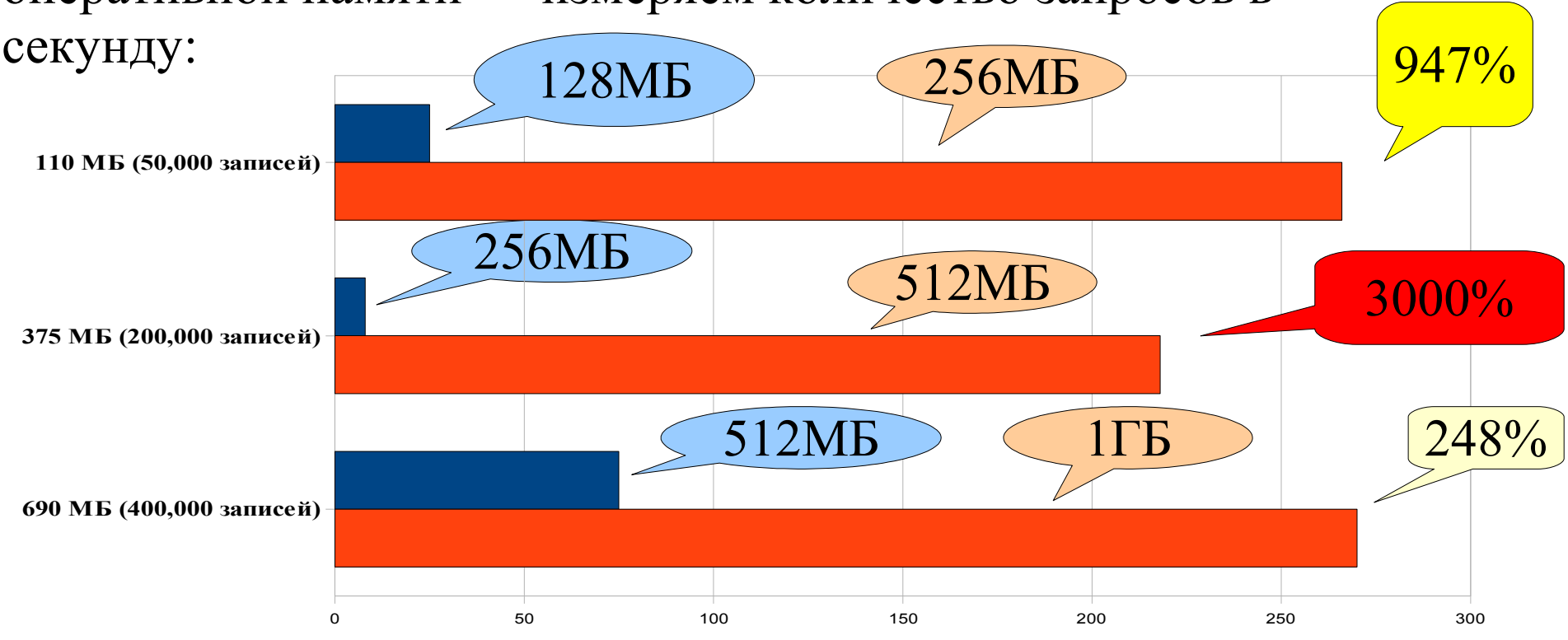
Жёсткий диск — это скорее далекая библиотека, а не книжный шкаф около стола. И этот факт необходимо учитывать.

Видимо еще не настало время отказываться от «медленной» оперативной памяти.

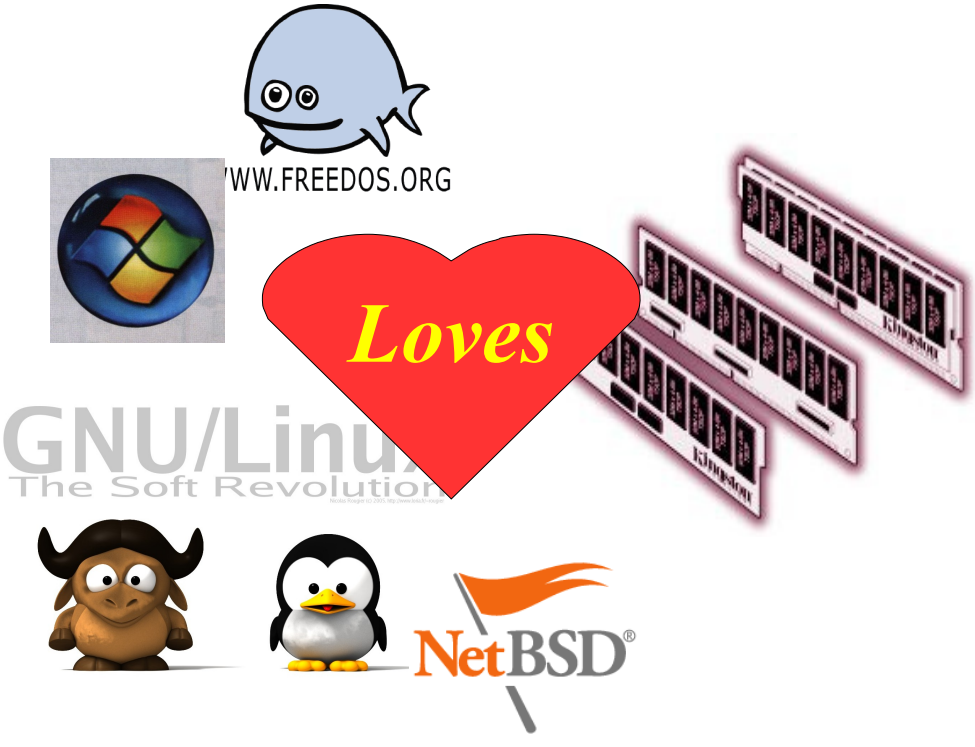


# *Увеличение оперативной памяти*

Посмотрим как СУБД PostgreSQL реагирует на удвоение оперативной памяти — измеряем количество запросов в секунду:

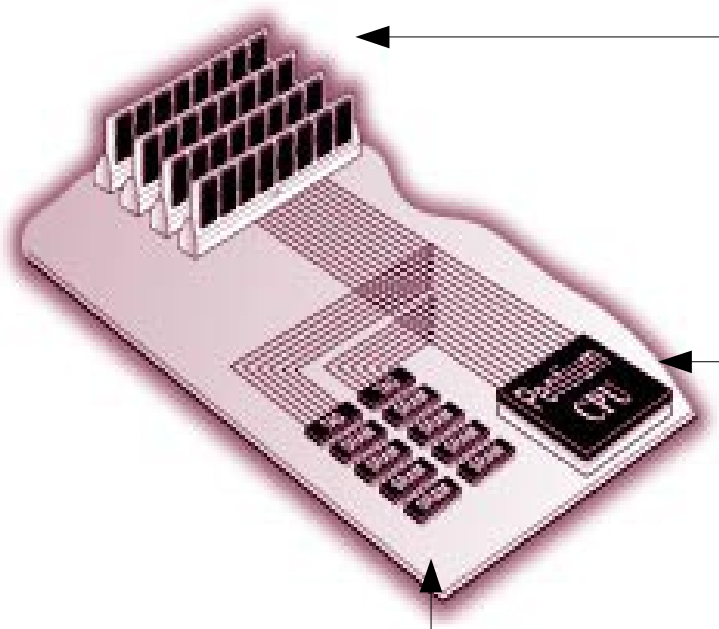


# *Почему необходима иерархия?*



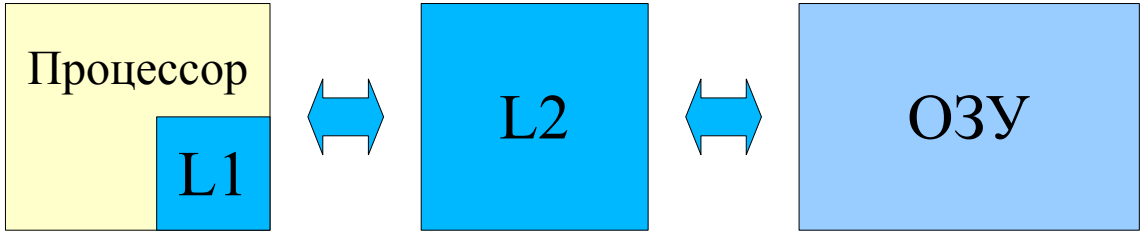

- Статической памяти не может быть много, как по инженерным, так и по экономическим основаниям
- Современные операционные системы и приложения не могут быть эффективны без большой оперативной памяти
- Мы вынуждены использовать динамическую память в больших количествах, чтобы получить эффективные системы

# *Начинаем кэшировать*

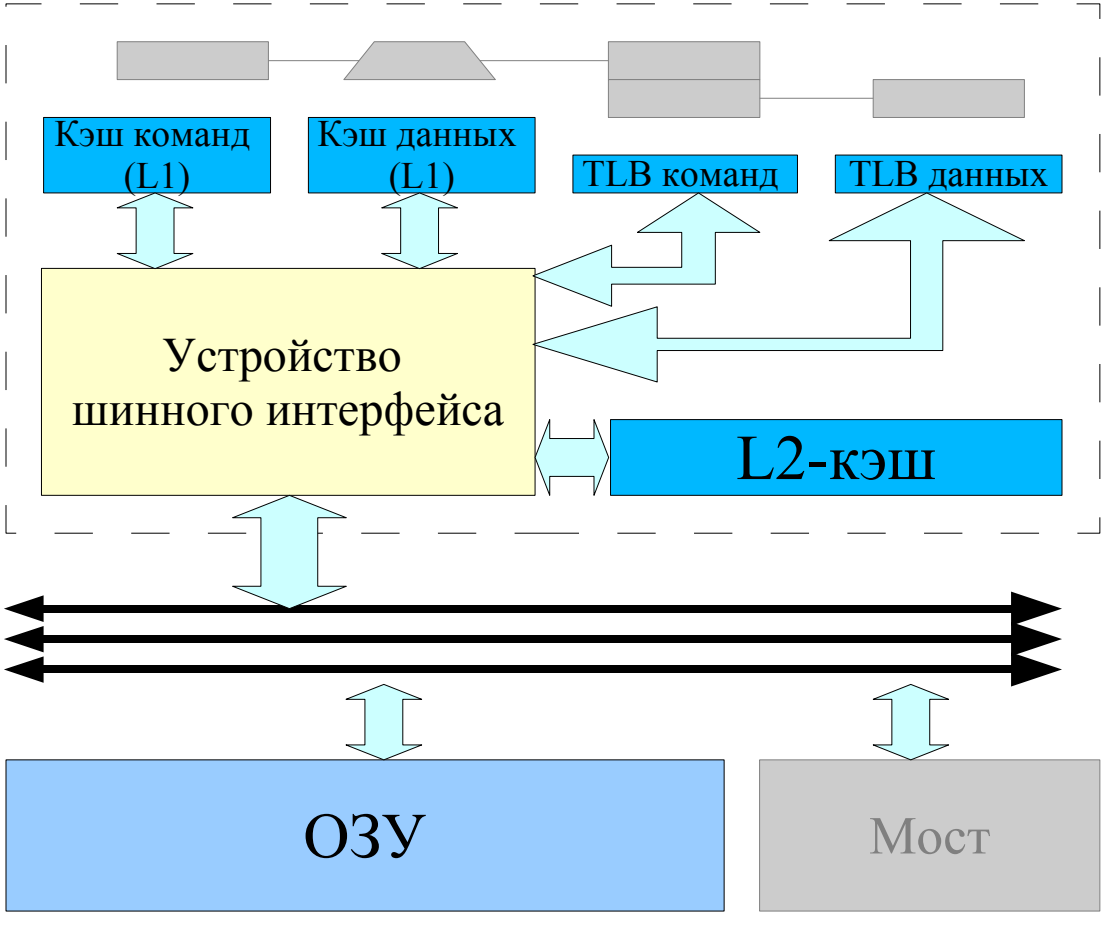



Так было в далекие времена 80486 и Pentium.

Кстати, многие дешевые машины обходились тогда без L2-кэша.



# *Как это делает Pentium 4*

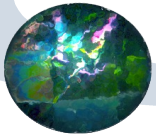



Внутри процессоров IA-32 **Pentium 4**, Xeon и Core2 размещены по крайней мере три кэша:

- L1 кэш команд (16K)
- L1 кэш данных (16-32K)
- L2 универсальный кэш (1МБ-8МБ)

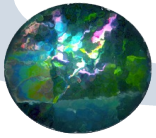
Для страничной памяти используется ещё два кэша:

- TLB команд
- TLB данных

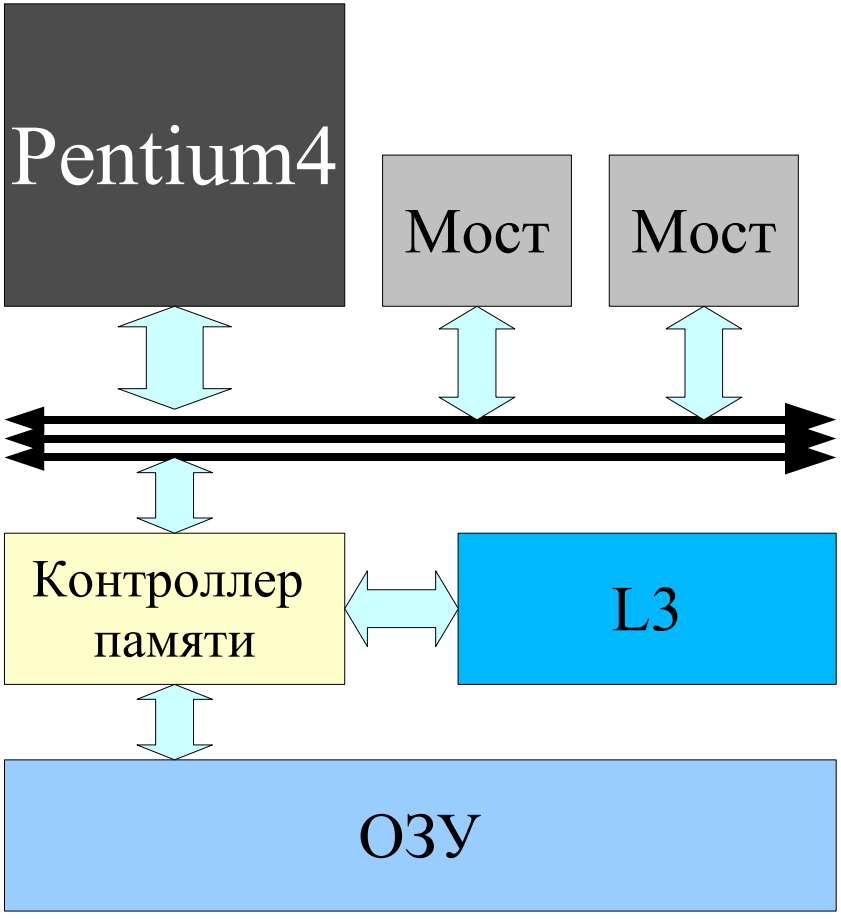


# *Как это делает Core2*

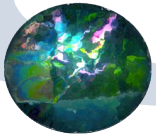
- На самом деле, L1 кэши конструктивно и логически являются частью ядра процессора, поэтому они повторяются в каждом ядре процессора Core2
- То есть
  - в Core2Duo — 4 × L1-кэша и 4 × TLB
  - в Core2Quad — 8 × L1-кэшей и 8 × TLB
- Универсальный L2-кэш единственный и совместно используется всеми ядрами процессора



# *А где L3?*



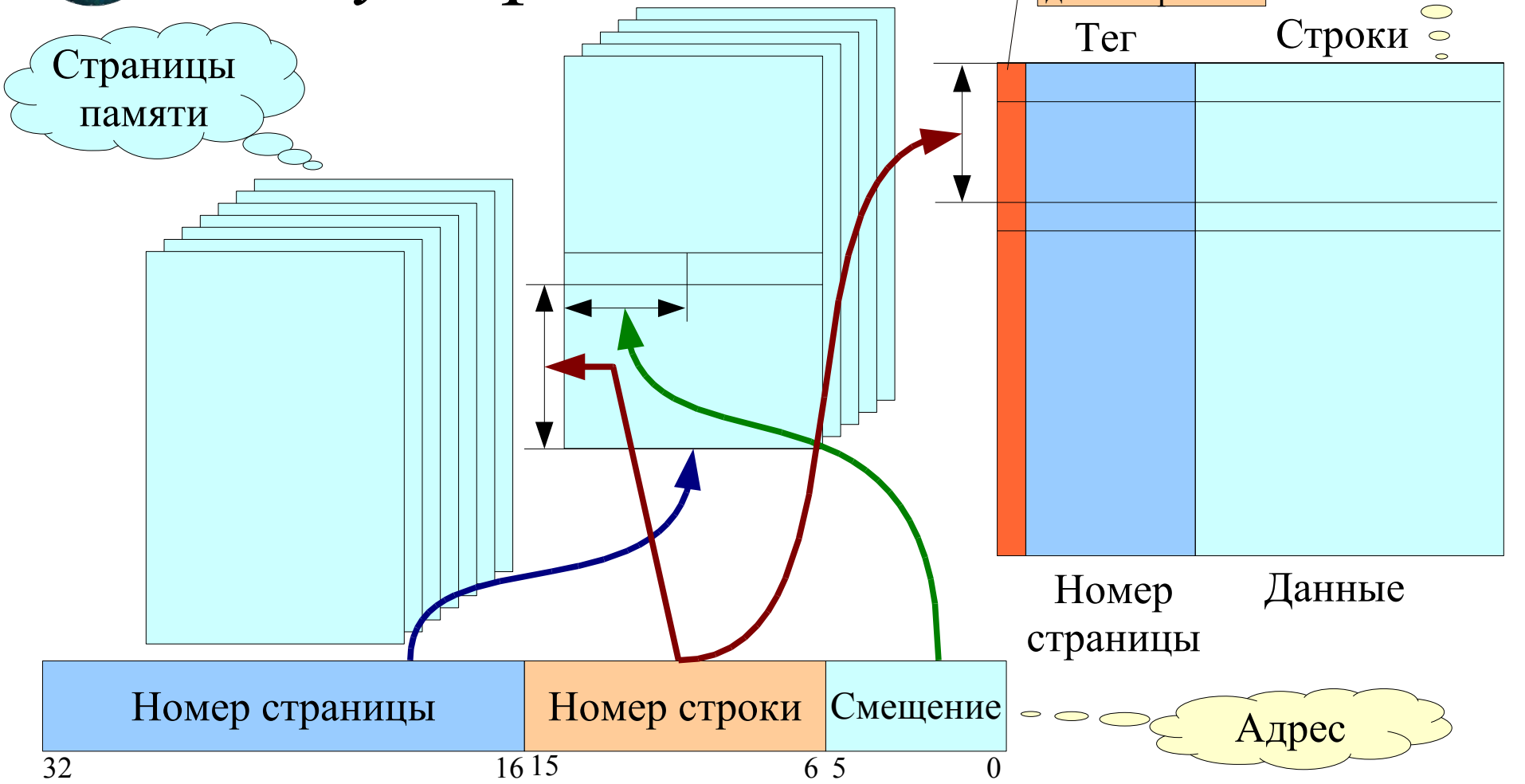

- Динамическая память имеет сравнительно низкое быстродействие и иногда бывает недоступна из-за регенерации
- Кэш позволяет увеличить производительность подсистемы памяти

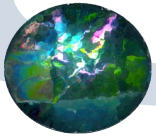


# *Как устроен кэш?*

- Кэши в процессорах IA-32 используют реальные (физические) адреса.
- В кэше хранятся не отдельные байты, а группы из 64 байтов
- Устройство шинного интерфейса всегда читает из оперативной памяти блоки по 64 байта
- Пустые строки и строки к которым обращались на запись другие устройства, имеют сброшенный бит достоверности

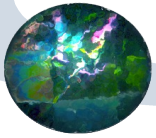
# *Как устроен кэш?*



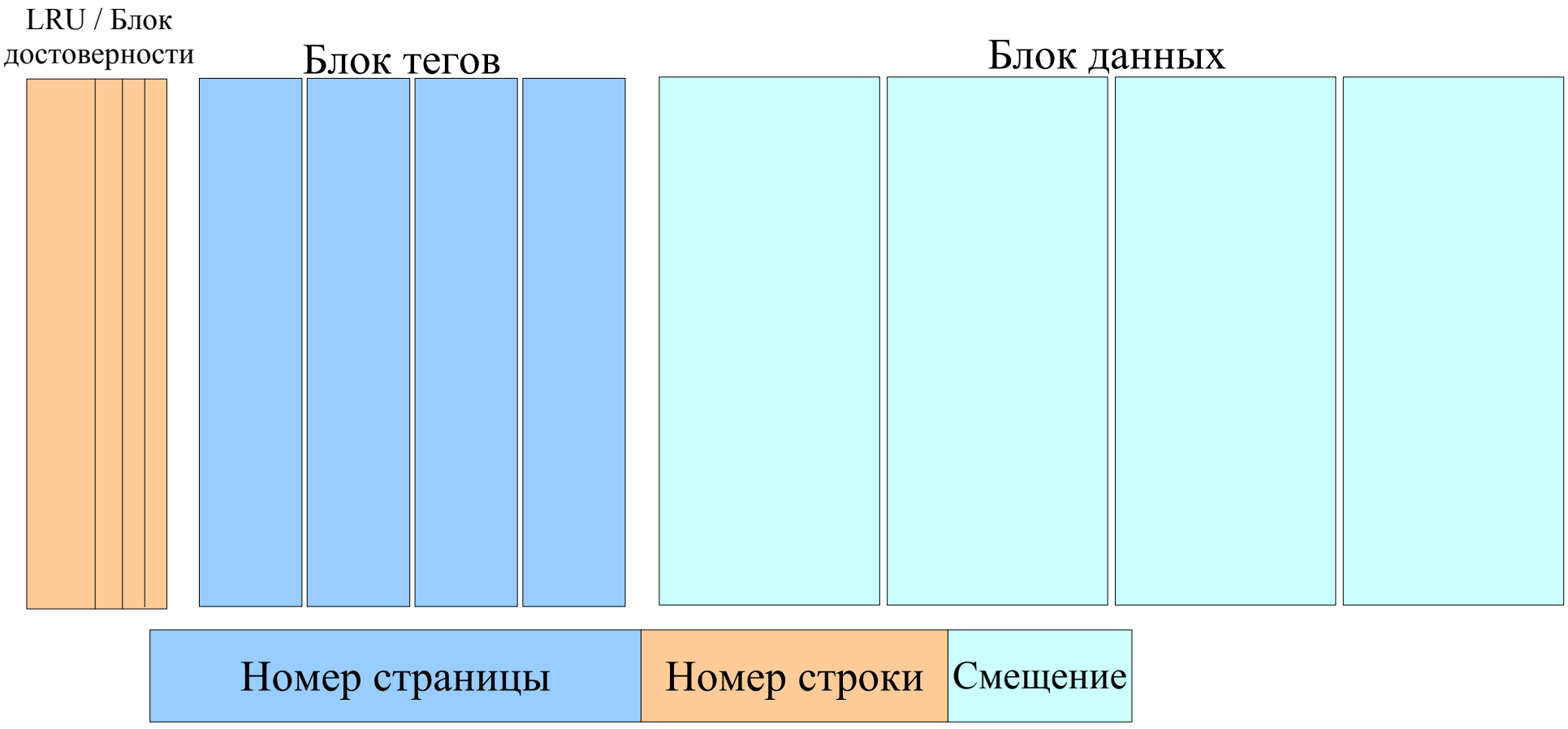


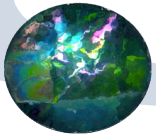
# *Такой кэш не используется!*

- В этом кэше мы сохраняем только одно строку данных для всех страниц. Поэтому вероятность коллизии кэша весьма высока.
- То есть, весьма вероятно, что обращение к данным с другой страницы даст тот же номер строки.
- Как избежать коллизий?
- Увеличим число каналов (входов) в кэше...



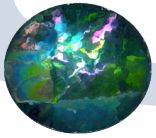
# *Как это делает Pentium 4*





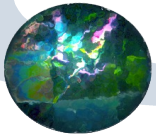
# *Как это делает Pentium 4*

- Блок LRU/битов достоверности содержит четыре бита достоверности (по одному на каждую строку) и три бита B0, B1, B2 для поддержки алгоритма псевдо-LRU
- Блок тегов сохраняет номера страниц для каждой из сохраненных строк
- Блок данных сохраняет сами строки



# *Как это делает Pentium 4*

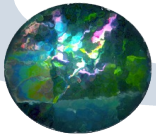
- Сначала перезаписываем недостоверные строки
- Если все строки достоверны, используем алгоритм псевдо-LRU (Least Recent Used):
  - Если обращались к L0-L1 установим B0 = 1, если же к L2-L3 то B0 = 0.
  - Если в паре L0-L1 обращались к L0 установим B1=1, иначе B1=0.
  - Если обращались к L2, то B2=1, иначе B2=0.



# *Как это делает Pentium 4*

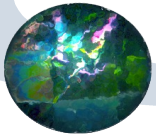
Теперь если нет недостоверных строк — выбираем замещаемую строке по правилам:

| B0 | B1 | B2 | |
|---|---|---|---|
| **0** | **0** | **x** | заменяется строка L0 |
| **0** | **1** | **x** | заменяется строка L1 |
| **1** | **x** | **0** | заменяется строка L2 |
| **1** | **x** | **1** | заменяется строка L3 |



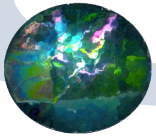
# *Как это делает Pentium 4*

- Все встроенные кэши процессоров IA-32 используют эту технику и являются четырехканальными прямыми кэшами.
- Отличие между кэшами — в количестве строк в кэшах



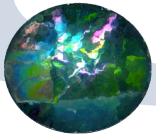
# *Translation Lookaside Buffer*

- При включенной страничной памяти (то есть всегда), процессор вынужден постоянно обращаться к таблицам страниц в оперативной памяти, чтобы транслировать виртуальные адреса в физические
- Translation Lookaside Buffer (TLB) кэширует для процессора последние 32 дескриптора кадров памяти



# *Translation Lookaside Buffer*

- TLB организован как четырехканальный кэш, использующий адрес страницы (часть виртуального адреса)



# *Библиография*


**1. Богачев К.Ю.** ***Архитектура процессоров*** **– Москва, 1999. – 127 с.**

***2. Intel® 64 and IA-32 Architectures Software Developer's Manual*** **– Vol. 1, 2a, 2b, 3a, 3b – Intel Corp., 2006. - 466 p.; 774 p.; 612 p.; 640 p.; 610 p.**

***3. Intel® 64 and IA-32 Architectures Optimization Reference Manual*** **– Intel Corp., 2006. – 490 p.**

***4. The ultimate memory guide*** **— Kingston Technology, 2000 - 110 p.**