Основные характеристики архитектуры фон Неймановского типа

- последовательно адресуемая единственная память линейного типа для хранения программ и данных;
- команды и данные различаются через идентификатор неявным способом лишь при выполнении операций. Принимаемые по умолчанию соглашения типа: операнды операции умножения – это данные, а объект, на который указывает команда перехода – это команда, позволяют обращаться с командой как с данными, например, для ее модификации;
- назначение данных определяется лишь логикой программы, так как в памяти машины набор бит может представлять собой как десятичное число с фиксированной точкой, так и строку символов.

Требования ЯВУ к архитектуре ЭВМ

- память состоит из набора дискретных именуемых переменных.
- ЯВУ наряду с линейными данными оперируют и с многомерными: массивами, структурами, списками;
- в ЯВУ четко разграничены операции и данные;
- данные определяют и операции над ними.

Методы адресации

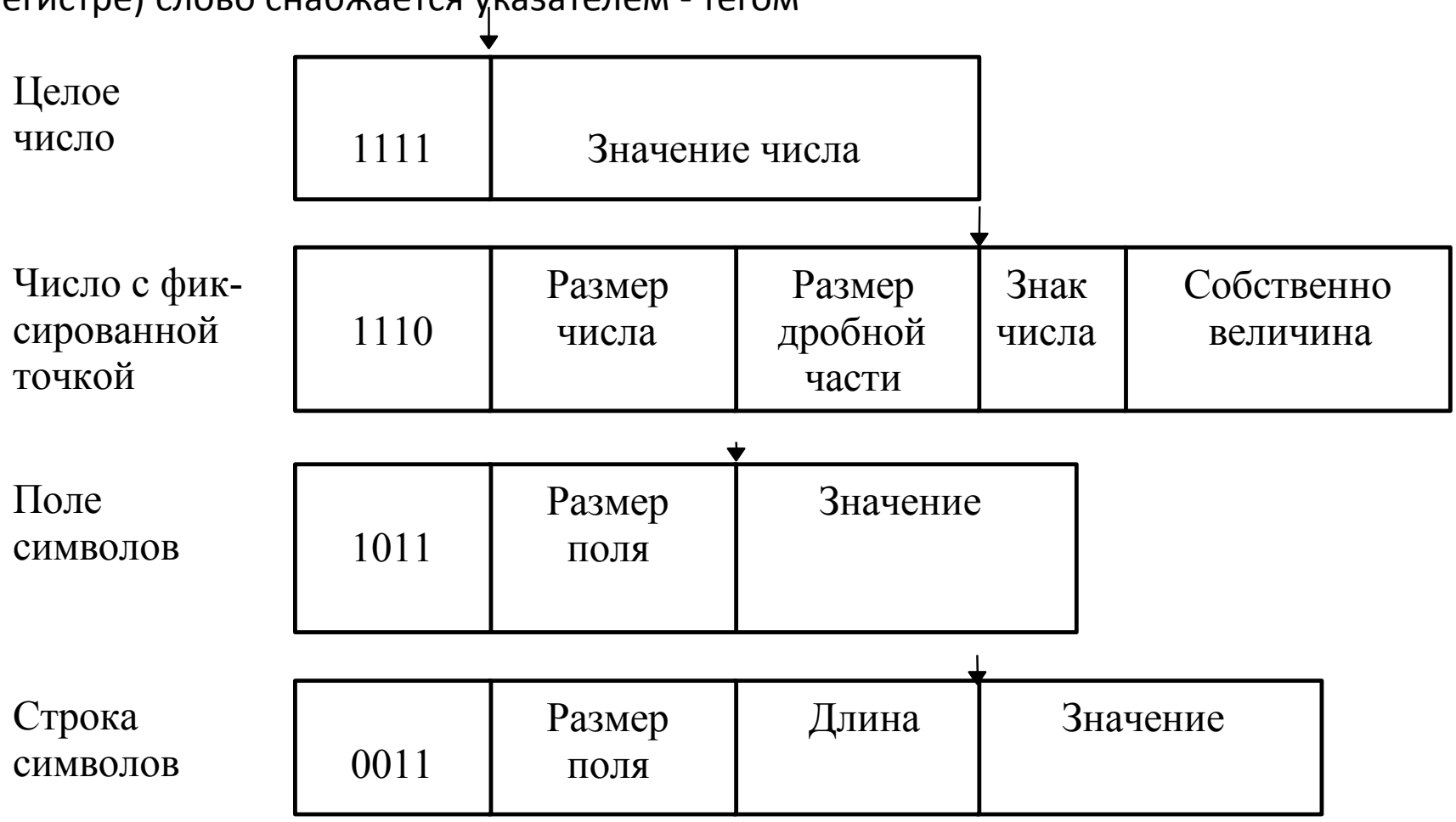
Метод адресации	Пример команды	Смысл команды	Использование команды
Прямая или абсолютная	Add R1, (1000)	R1=R1+M(1000)	Полезна для обращения к статическим данным
Косвенная	Add R1, @(R3)	R1 = R1+M(M(R3))	Если R3 – адрес указателя p, то выбирается значение по этому указателю
Автоинкрементная	Add R1, (R2)+	R1 = R1+M(R2) R2 = R2+d	Полезна для прохода в цикле по массиву с шагом: R2 – начало массива. В каждом цикле R2 получает приращение d
Автодекрементная	Add R1, (R2)–	R2 = R2–d R1 = R1+M(R2)	Аналогична предыдущей. Обе могут использоваться для реализации стека
Базовая индексная со смещением и масштабированием	Add R1, 100(R2)(R3)	R1=R1+M(100)+R2 +R3*d	Для индексации массивов

Методы адресации

Метод адресации	Пример команды	Смысл команды	Использование команды
Регистровая	Add R4, R3	R4 = R4+R3	Для записи требуемого значения в регистр
Непосредственная или литерная	Add R4, #3	R4 = R4+3	Для задания констант
Базовая со смещением	Add R4, 100(R1)	R4= R4+M(100+R1)	Для обращения к локальным переменным
Косвенная регистровая	Add R4, (R1)	R4 = R4+M(R1)	Для обращения по указателю к вычисленному адресу
Индексная	Add R3, (R1+R2)	R3 = R3+M(R1+R2)	Полезна при работе с массивами: R1 – база, R3 – индекс

Примеры типов ячеек при теговой организации

Одним из эффективных средств совершенствования архитектуры совре-менных ЭВМ является теговая организация памяти, при которой каждое хранящееся в памяти (или регистре) слово снабжается указателем - тегом



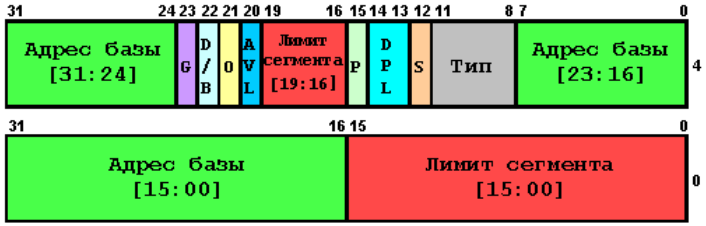
Пример дескриптора

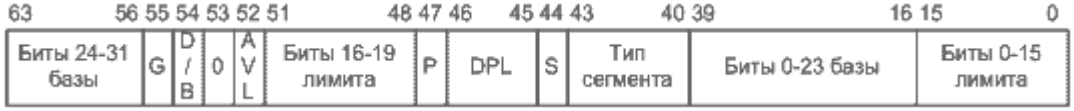
В архитектуре некоторых ЭВМ используются дескрипторы — служебные слова, содержащие описание массивов данных и команд, причем дескрипторы могут употребляться как в машинах с теговой организацией памяти, так и без тегов.
Дескриптор содержит сведения о размере массива данных, его местоположении (в ОП или внешней памяти), адресе начала массива, типе данных, режиме защиты данных (например, запрет записи в ячейки массива) и некоторых других параметрах данных. Отметим, что задание в дескрипторе размера массива позволяет контролировать выход за границу массива при индексации его элементов

- Основное отличие тегов и дескрипторов состоит в следующем: дескрипторы создают дополнительный уровень адресации, что требует увеличения затрат на формирование адреса

101	P	I	R	W	Длина	Адрес

Здесь первые три бита содержат тег. Если значение его 101, то данное слово дескриптор. Бит P указывает, находятся данные в основной памяти или во вспомогательной; I указывает, одиночный ли элемент описывает данный дескриптор или весь массив; R идентифицирует непрерывную или разрывную область памяти; W означает, что разрешено только чтение данных.





Достоинства виртуальной памяти

- Однородность области адресов

  каждый процесс может выполняться в памяти начиная с фиксированной (обычно нулевой) ячейки, имеющей необходимые размеры области ЗУ. Каждое обращение к виртуальной памяти во время выполнения посредством АПА преобразуется в реальное обращение.
- Защита памяти

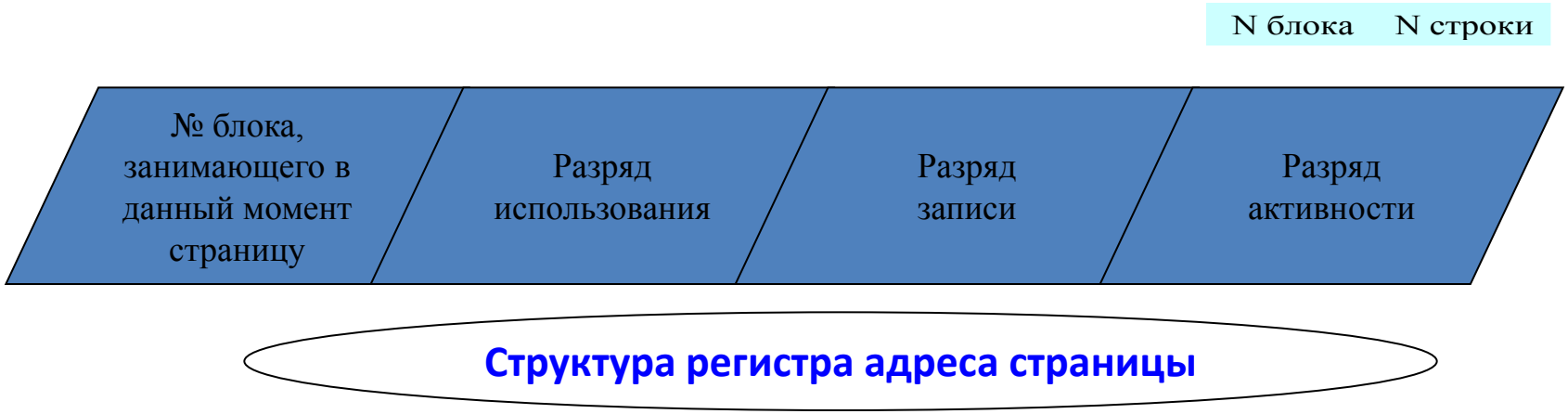
  при каждой ссылке процессом на память проверяется, принадлежит ли она к области виртуальных адресов, отведенных для данного процесса.
- Изменение структуры памяти

  Применение виртуальной адресации позволяет преобразовать память на разных ступенях иерархии в "одноуровневую память" с одинаковым доступом ко всем элементам.

Виртуальная память

Виртуальную память пользователя можно разделить на три типа:

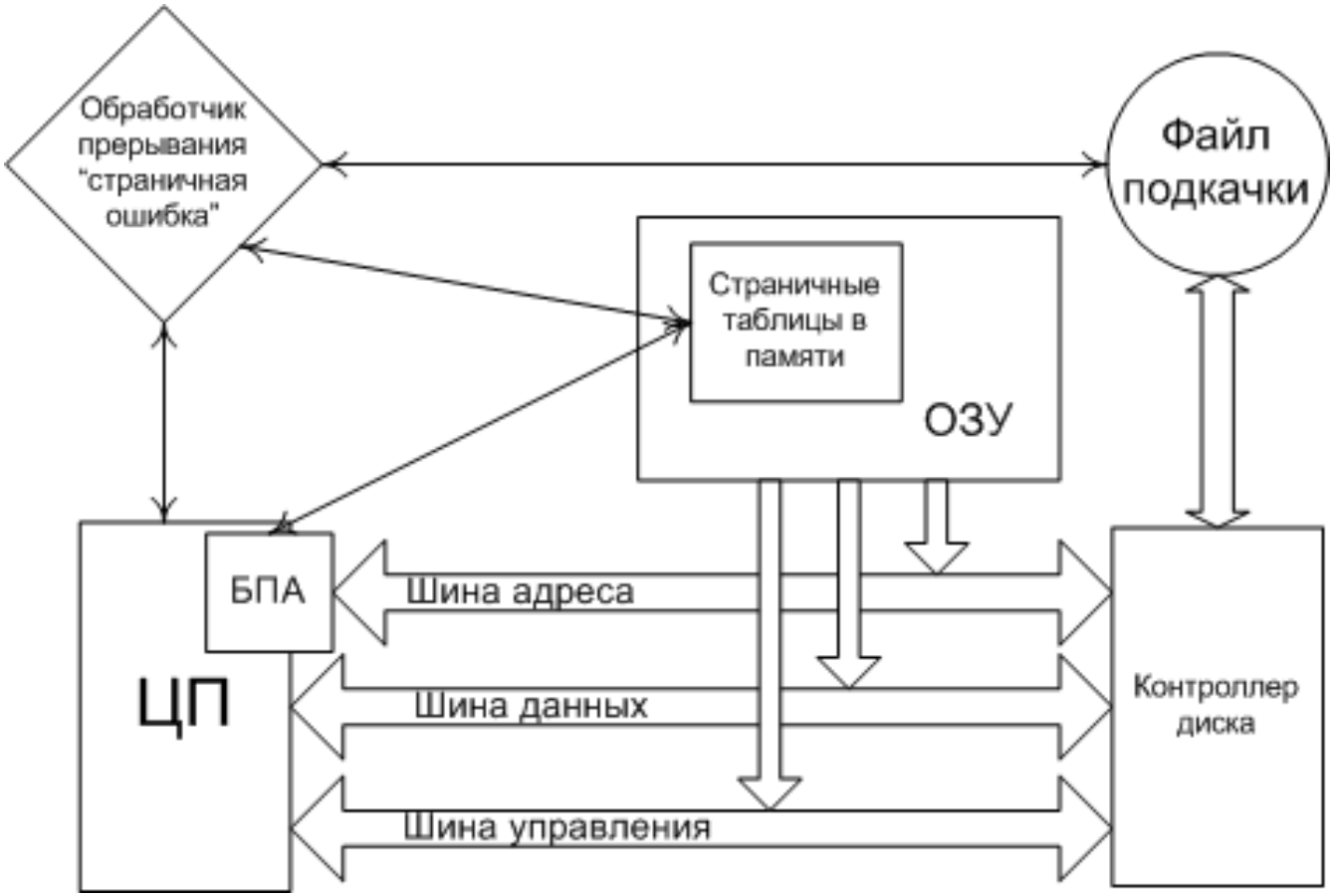
- "активные" блоки, которые содержат программу и данные, используемые в текущий момент;
- "пассивные" блоки, содержащие программу и данные, которые будут использоваться при выполнении программы;
- "мнимые" блоки, к которым не обращаются на протяжении выполнения программы.



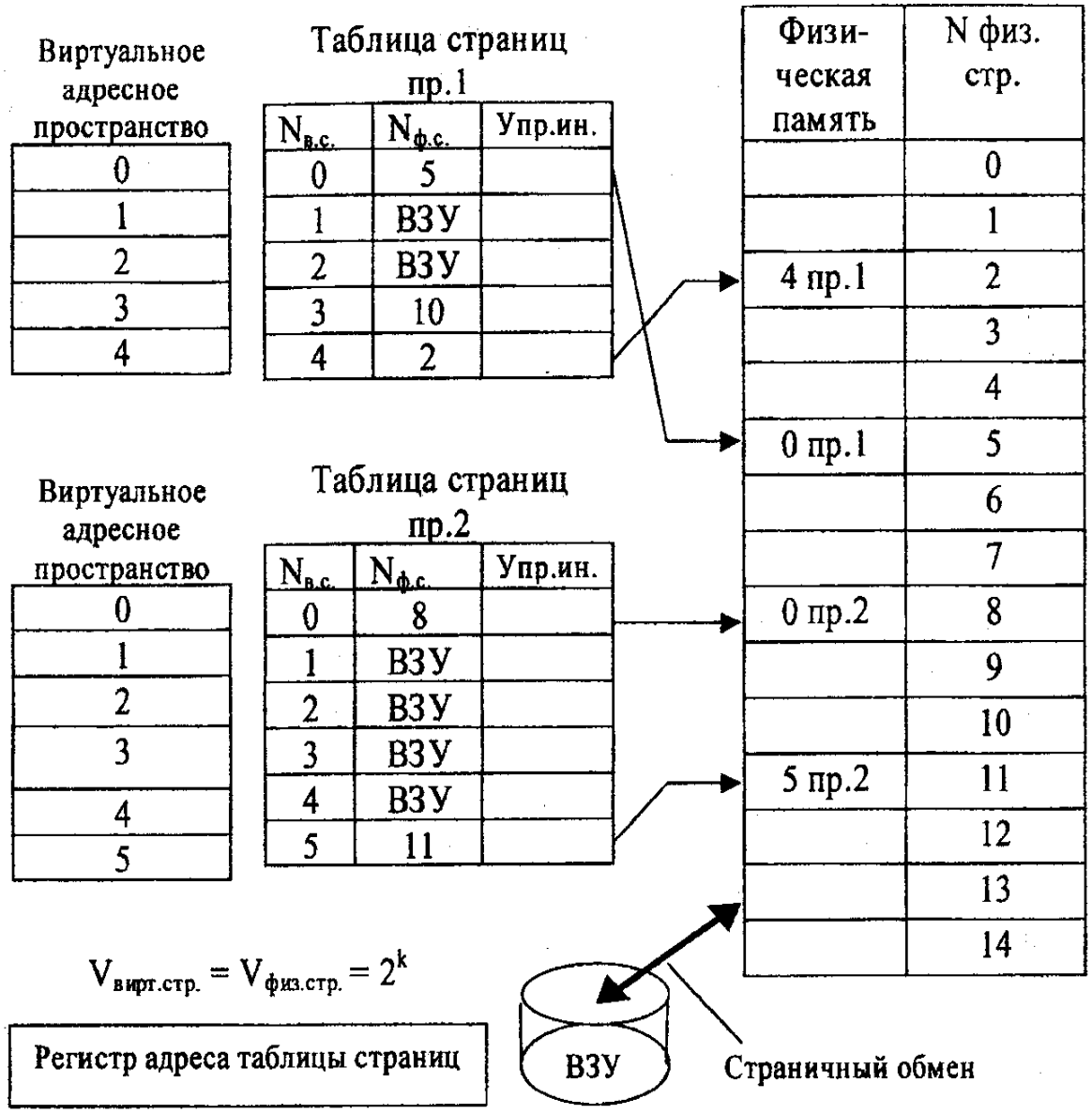
Структура регистра адреса страницы

виртуальная память — это совокупность программно-аппаратных средств, позволяющих пользователям писать программы, размер которых превосходит имеющуюся ОП.

Функционирование виртуальной памяти



Страничное распределение памяти



Виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые виртуальными страницами. В общем случае размер виртуального адресного пространства не является кратным размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью.

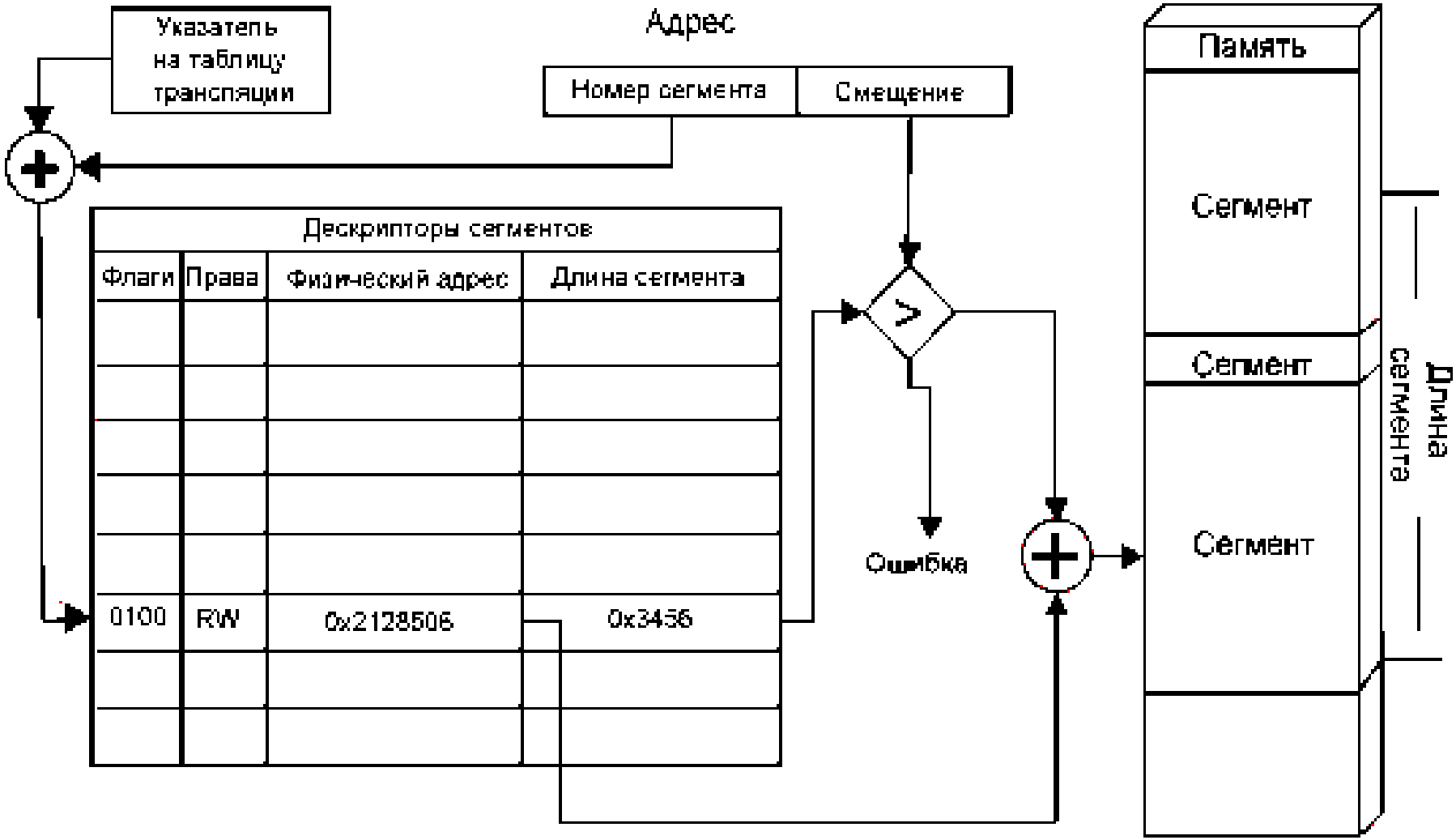
Вся оперативная память машины также делится на части такого же размера, называемые физическими страницами (или блоками).

Размер страницы обычно выбирается равным степени двойки: 512, 1024 и т.д., это позволяет упростить механизм преобразования адресов.

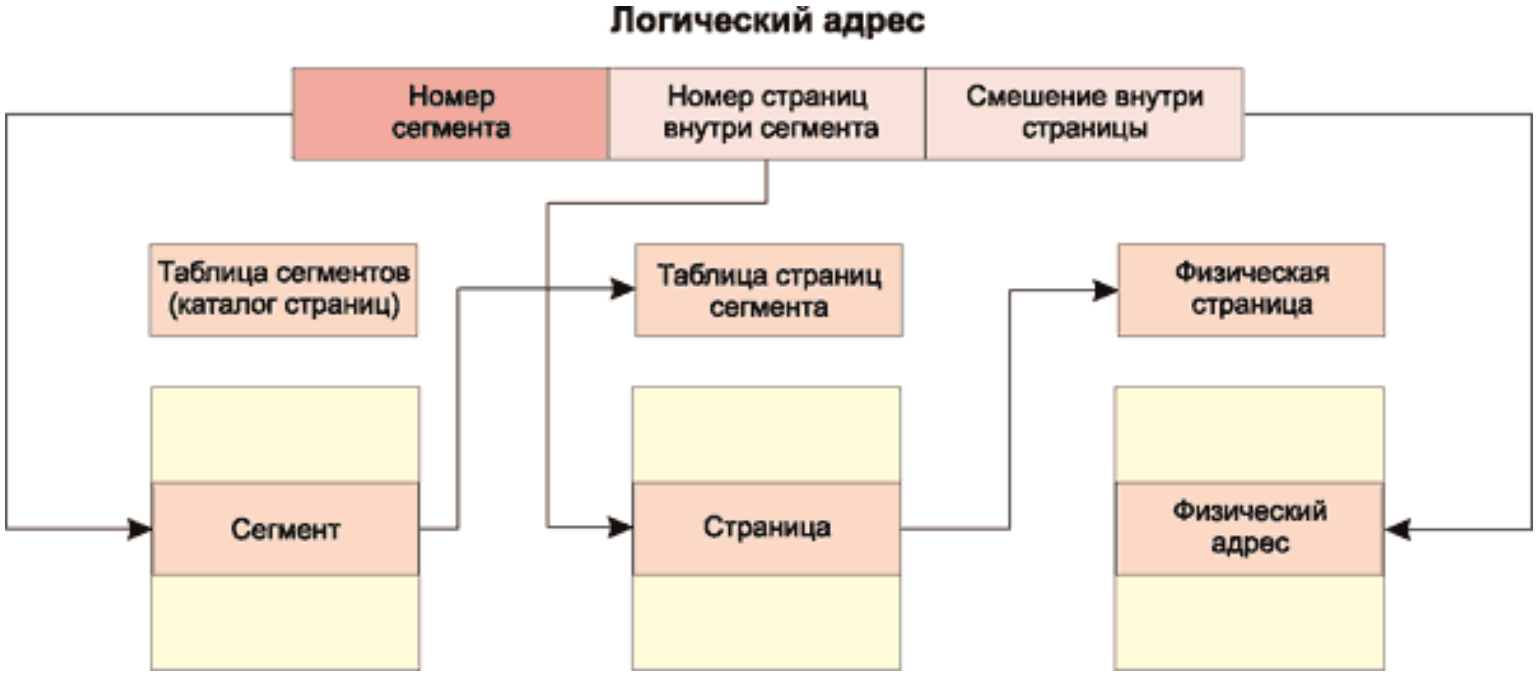
Механизм преобразования виртуального адреса в физический

1. На основании начального адреса таблицы страниц (содержимое регистра адреса таблицы страниц), номера виртуальной страницы (старшие разряды виртуального адреса) и длины записи в таблице страниц (системная константа) определяется адрес нужной записи в таблице.
2. Из записи извлекается номер физической страницы.
3. К номеру физической страницы присоединяется смещение (младшие разряды виртуального адреса.

Сегментное распределение

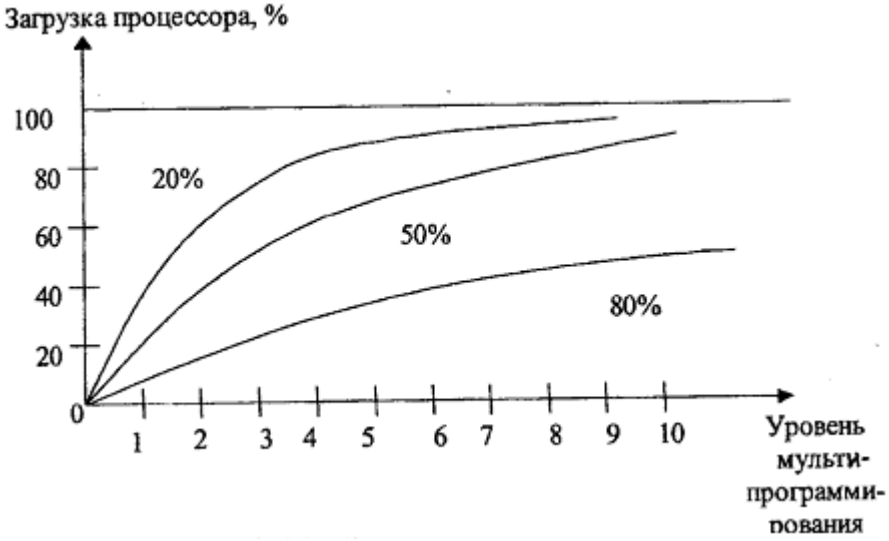


Формирование реального адреса



Виртуальный адрес при сегментной организации памяти может быть представлен парой (g, s), где g — номер сегмента, as — смещение в сегменте. Физический адрес получается путем сложения начального физического адреса сегмента, найденного в таблице сегментов по номеру g, и смещения s.

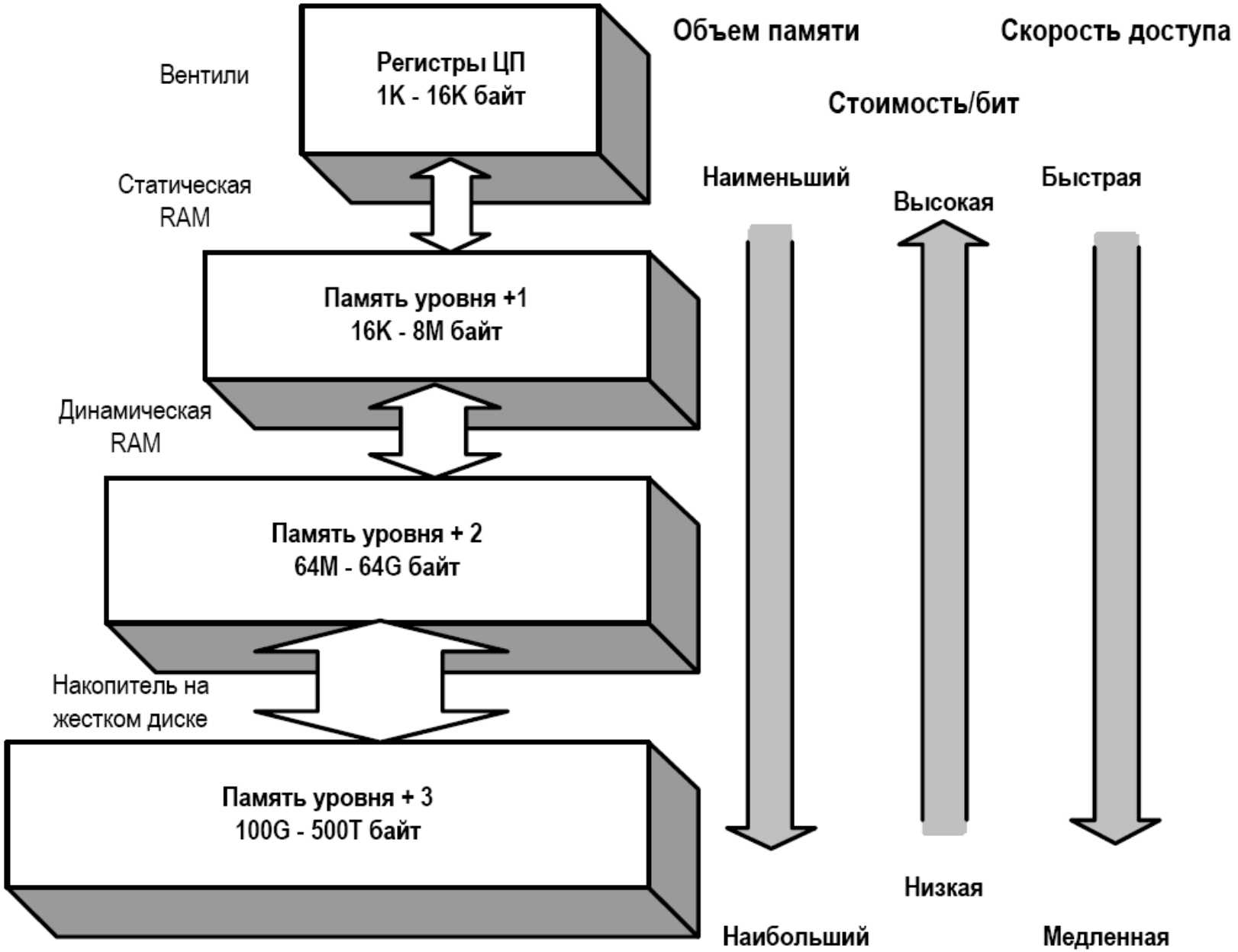
Свопинг



Зависимость загрузки процессора от числа задач и интенсивности ввода-вывода

В соответствии с
этим методом некоторые процессы (обычно находящиеся в состоянии ожидания) временно выгружаются на диск. Планировщик операционной системы не исключает их из своего рассмотрения, и при наступлении условий активизации некоторого процесса, находящегося в области свопинга на диске, этот процесс перемещается в оперативную память. Если свободного места в оперативной памяти не хватает, то выгружается другой процесс.

Иерархия памяти в компьютере



Параметры различных типов памяти.

Технология памяти	**Типичное время доступа**	**Стоимость за Мегабайт**
Register RAM Регистровая память	250 – 1000 пикосекунд	Очень дорого, на порядок величины по сравнению со статической памятью
SRAM Статическая память	0.5 – 5 наносекунд	$4 - $10 , достаточно дорого
DRAM Динамическая память	50 – 70 наносекунд	$0.1 - $0.2 вполне доступно
Магнитные диски	5 – 20 миллисекунд	$0.0005-$0.002 очень недорого
Магнитные ленты	Секунды и минуты	Совсем дешево

Характеристики различных типов памяти

Тип запоминающего устройства	Категория	Стирание информации	Изменение информации по байтам	Необходимость питания	Применение
SRAM	Чтение и запись	Электрическое	Да	Да	Кэш-память второго уровня
DRAM	Чтение и запись	Электрическое	Да	Да	Основная память (старые модели)
SDRAM	Чтение и запись	Электрическое	Да	Да	Основная память (новые модели)
ROM	Только чтение	Невозможно	Нет	Нет	Устройства большого объема
PROM	Только чтение	Невозможно	Нет	Нет	Устройства небольшого объема
EPROM	Преимущественно чтение	Ультрафиолетовый свет	Нет	Нет	Моделирование устройств
EEPROM	Преимущественно чтение	Электрическое	Да	Нет	Моделирование устройств
Флэш-память	Чтение и запись	Электрическое	Нет	Нет	Цифровые камеры

Кэш-память

Процессоры работают быстрее чем память! При обращении процессора к памяти из-за разницы в скорости работы процессору приходится ждать. У этой проблемы существует два способа решения:

1) начать считывать команды заранее. Проблема: если какая-либо команда захочет использовать это слово, то процессор должен простаивать.

2) сконструировать машину, которая не приостанавливает работу, но следит, чтобы компиляторы не использовали слова, до того как они считаются из памяти. Проблема – слишком сложно!

Быструю память можно сделать! Но для это она должна быть в одной микросхеме с процессором (это не всегда так, но наиболее близко к истине). Следствие – очень дорого!

Промежуточное решение - это сочетание технологий с использованием маленькой и быстрой памяти с большой и медленной. Маленькая память с большой скоростью работы называется кэш-памятью.

Что хранится в кэш-памяти?

В ней находятся слова, которые чаще всего используются.

Если процессору нужно какое-нибудь слово он сначала обращается к кэш-памяти и только, если и там его нет, он обращается к основной памяти.

Если значительная часть слов лежит в кэш-памяти, то скорость значительно возрастает!

Принцип локальности

Большинство программ вызывают команды из последовательных участков памяти, кроме того в циклах также обычно используется один и тот же кусок памяти.

Свойство программ, что при последовательных отсылках к памяти в течении некоторого промежутка времени они используют только ее небольшой кусок, называется принципом локальности. Он и составляет основу всех систем кэш-памяти.

Ситуация, когда при последовательных обращениях к памяти в течение некоторого промежутка времени используется только небольшая ее область, называется **принципом локальности**. Этот принцип составляет основу всех систем кэш-памяти. Идея состоит в том, что когда определенное слово вызывается из памяти, оно вместе с соседними словами переносится в кэш-память, что позволяет при очередном запросе быстро обращаться к следующим словам. Общее устройство процессора, кэш-памяти и основной памяти иллюстрирует рис. 2.13. Если слово считывается или записывается k раз, компьютеру требуется сделать 1 обращение к медленной основной памяти и $k - 1$ обращений к быстрой кэш-памяти. Чем больше k, тем выше общая производительность.

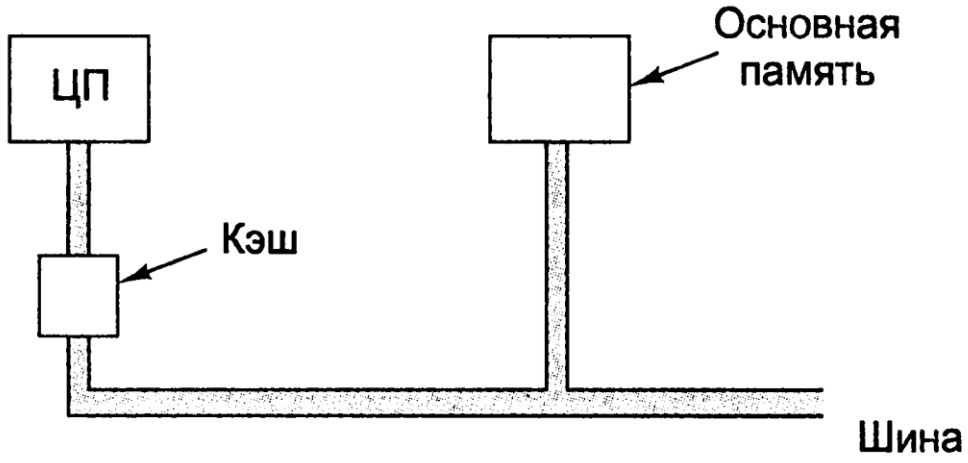


Рис. 2.13. Кэш-память по логике вещей должна находиться между процессором и основной памятью. В действительности существует три возможных варианта размещения кэш-памяти

Подсчитаем выигрыш

Среднее время доступа при к запросах к одному и тому же слову будет:

$$t = c + (1 - h)*m, \text{ где:}$$

c - время доступа к кэш-памяти
m - время доступа к основной памяти
h - коэффициент совпадения

Если слово считывается или записывается k раз, то

$$h = (k – 1)/k$$

При h -> 1 получаем выигрыш
При h -> 0 - проигрыш

В некоторых системах процесс обращения к основной памяти стартует вместе с процессом обращения к кэш-памяти и, если в кэш-памяти слова нет, то обращение к основной памяти будет уже начато.

Немного об организации кэш-памяти

Загрузка в кэш-память идет сразу строками, т.е. вся кэш память поделена на некоторые куски, большие машинного слова. При необходимости загрузить в кэш память слово из основной памяти загружается весь участок в котором это слово находится, по размеру совпадающий со строчкой кэш-памяти.

В современных машинах могут разделять кэш-память для данных и для команд. Также могут быть предусмотрены системы с несколькими уровнями кэш-памяти.

Кэш-память очень важна для высокопроизводительных процессоров. Однако здесь возникает ряд вопросов. Первый вопрос — объем кэш-памяти. Чем больше объем, тем лучше работает память, но тем дороже она стоит. Второй вопрос — размер строки кэша. Кэш-память объемом 16 Кбайт можно разделить на 1024 строки по 16 байт, 2048 строк по 8 байт и т. д. Третий вопрос — механизм организации кэш-памяти, то есть то, как она определяет, какие именно слова находятся в ней в данный момент. Устройство кэш-памяти мы рассмотрим подробно в главе 4.

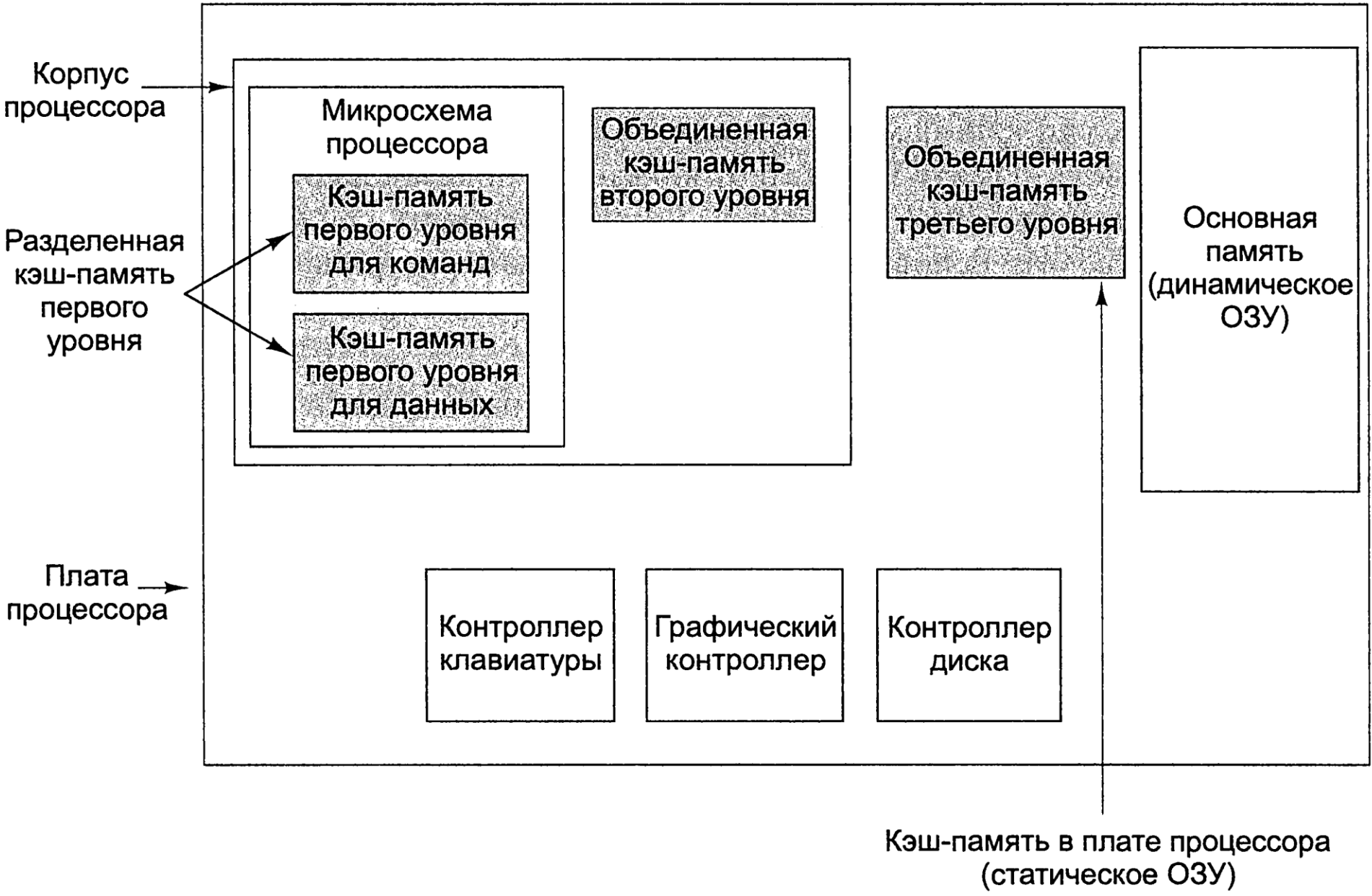
Четвертый вопрос — должны ли команды и данные находиться вместе в общей кэш-памяти. Проще всего разработать **объединенную кэш-память** (unified cache), в которой будут храниться и данные и команды. В этом случае вызов команд и данных автоматически уравновешивается. Однако в настоящее время существует тенденция к использованию **разделенной кэш-памяти** (split cache), когда команды хранятся в одной кэш-памяти, а данные — в другой. Такая архитектура также называется **гарвардской** (Harvard architecture), поскольку идея использования отдельной памяти для команд и отдельной памяти для данных впервые воплотилась в компьютере Marc III, который был создан Говардом Айкеном (Howard Aiken) в Гарварде. Современные разработчики пошли по этому пути, поскольку сейчас широко распространены конвейерные архитектуры, а при конвейерной организации должна быть возможность одновременного доступа и к командам, и к данным (операндам). Разделенная кэш-память позволяет осуществлять параллельный доступ, а общая — нет. К тому же, поскольку команды обычно не меняются во время выполнения программы, содержание кэша команд не приходится записывать обратно в основную память.

Кэш центрального процессора разделён на несколько уровней. Для универсальных процессоров — до 3.

Самой быстрой памятью является кэш первого уровня — **L1**-cache. **L1** кэш работает на частоте процессора, и, в общем случае, обращение к нему может производиться каждый такт (зачастую является возможным выполнять даже несколько чтений/записей одновременно). Латентность доступа обычно равна 2–4 тактам ядра. Объём обычно невелик — не более 128 КБ.

Вторым по быстродействию является **L2**-cache — кэш второго уровня. В современных многоядерных процессорах кэш второго уровня, находясь на том же кристалле, является памятью раздельного пользования — при общем объёме кэша в 8 МБ на каждое ядро приходится по 2 Мб. Обычно латентность **L2** кэша, расположенного на кристалле ядра, составляет от 8 до 20 тактов ядра.

Кэш третьего уровня наименее быстродействующий и обычно расположен отдельно от ядра ЦП, но он может быть очень внушительного размера — более 32 МБ. **L3** кэш медленнее предыдущих кэшей, но всё равно значительно быстрее, чем оперативная память. В многопроцессорных системах находится в общем пользовании.


Корпус процессора
Микросхема процессора
Объединенная кэш-память второго уровня
Объединенная кэш-память третьего уровня
Основная память (динамическое ОЗУ)
Разделенная кэш-память первого уровня
Кэш-память первого уровня для команд
Кэш-память первого уровня для данных
Плата процессора
Контроллер клавиатуры
Графический контроллер
Контроллер диска
Кэш-память в плате процессора (статическое ОЗУ)

Полный адрес данных в упрощенной памяти на 64 байта

Старшие биты адреса	Индекс (адрес блока данных)	Смещение (Адрес байта в блоке)
1 старший бит	3 средних бита	2 младших бита

Кэш прямого отображения на 32 байта

Номер строки кэша	000	001	010	011	100	101	110	111
Данные строки кэша	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит
Адресный тэг строки кэша	1 бит	1 бит	1 бит	1 бит	1 бит	1 бит	1 бит	1 бит

Основная память на 64 байта

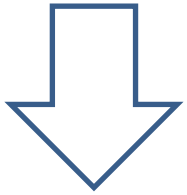
Прямое отображение блока основной памяти в строку кэша

Адрес	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Данные	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит	32 бит

Рассмотренный модуль памяти имеет 8 ячеек, называемых строками кэша и в каждой из которых хранится 32 бита данных и 2 бита использованы для хранения служебной информации, необходимой для обеспечения унифицированного доступа к данным со стороны ЦП. Попробуем посчитать процент дополнительной памяти: ((2 бита * 8 ячеек) / (8 бит * 4 байта * 8 ячеек)) * 100% = 6.25%.

Табл. 2. Структура модуля памяти упрощенного кэша прямого отображения.

Номер строки кэша	**Флаг действи-тельности данных**	**Адресный тэг блока данных (старший бит адреса)**	**Данные Байт 3**	**Данные Байт 2**	**Данные Байт 1**	**Данные Байт 0**
0	< 1 бит >	< 1 бит > < 27 бит >	<8 бит>	<8 бит>	<8 бит>	<8 бит>
1	< 1 бит >	< 1 бит > < 27 бит >	<8 бит>	<8 бит>	<8 бит>	<8 бит>
...	....	.....	...	..	..	...
7	< 1 бит >	< 1 бит > < 27	<8 бит>	<8 бит>	<8 бит>	<8 бит>

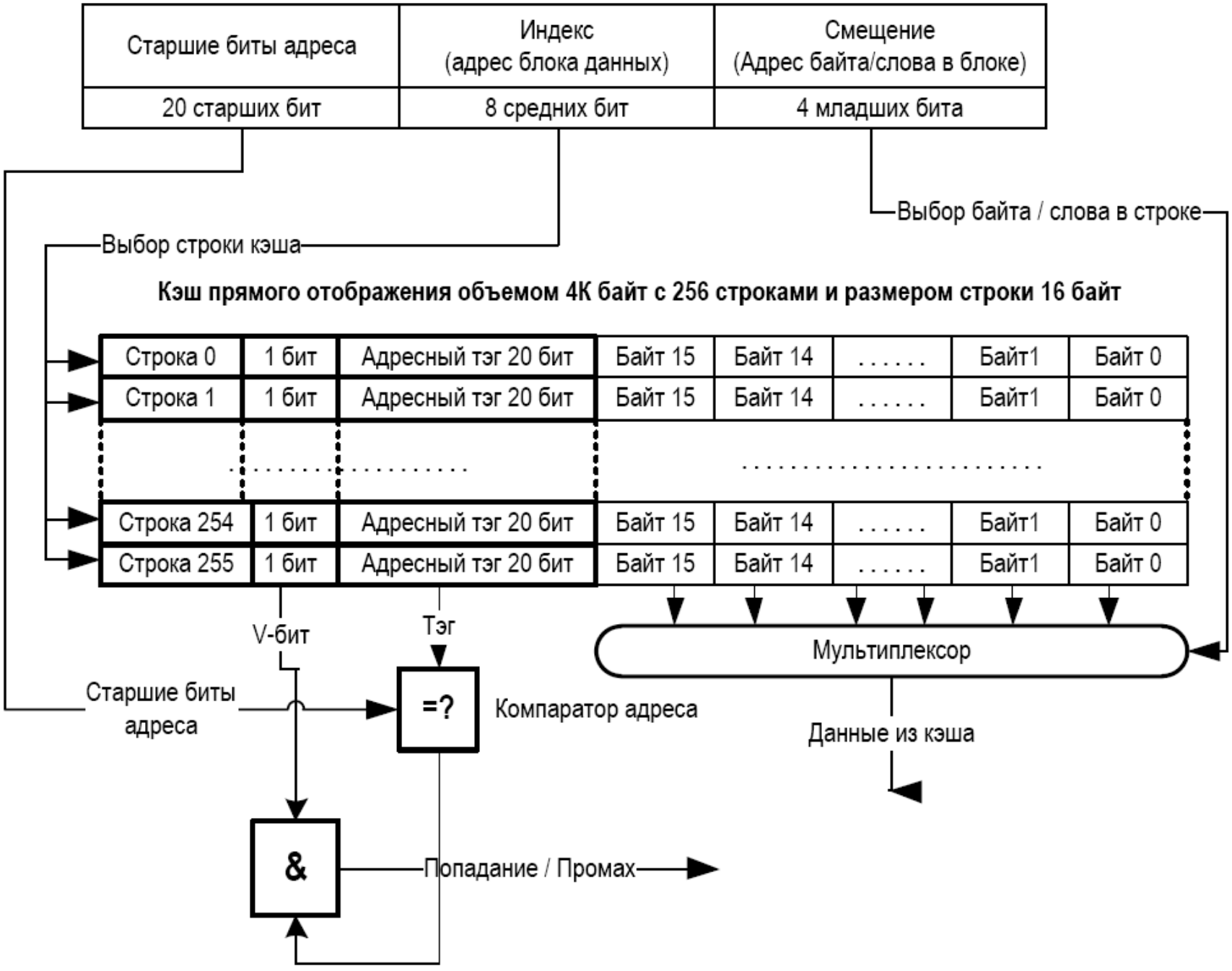


Старшие биты адреса	**Индекс (адрес блока данных)**	**Смещение (Адрес байта в блоке)**
Биты адреса [31:5]	Биты адреса [4:2]	Биты адреса [1:0]

1. ЦП генерирует полный 32-разрядный адрес необходимых данных и посылает этот адрес кэш-контроллеру;
2. Кэш-контроллер выбирает биты индекса и читает содержимое строки кэш-памяти (включая адресный тэг и данные), при этом данные не передаются ЦП.
3. Адресный тэг из кэш-памяти сравнивается со старшими 25 битами адреса, сгенерированного ЦП. Если значения совпадают, то произошло попадание и считанные данные должны быть переданы ЦП с использованием значения поля смещения для выбора нужного байта.
4. Если значение тэга и старших битов адреса от ЦП не совпадают, то произошел промах и данные должны быть загружены из основной памяти в данную строку кэш-памяти. Считанные при промахе данные из кэш-памяти аннулируются и ЦП должен остановиться и ждать, пока требуемые данные будут загружены в кэш из основной памяти. Кэш-контроллер получив запрос, должен считать требуемый блок данных из основной памяти и послать их назад в кэш (при этом одновременно данные могут быть посланы ожидающему их ЦП).
5. Затем ЦП должен считать данные, на доступе к которым произошел промах и завершив исполнение этой команды, продолжить нормальную последовательность работы.

Полный адрес данных в основной памяти на 4 G байта с 32-бит адресным пространством

Старшие биты адреса	Индекс (адрес блока данных)	Смещение (Адрес байта/слова в блоке)
20 старших бит	8 средних бит	4 младших бита



Нам необходимо хранить адресный тэг 20 бит, 1 бит для флага достоверности и еще 1 бит флага изменения данных. Флаг изменения необходимо устанавливать каждый раз, когда ЦП производит запись данных в кэш-память для последующей синхронизации содержания кэш-памяти и основной памяти (напоминание о том, что оба уровня памяти должны содержать одни и те же данные). Таким образом имеется 22 бита для каждой строки. Процент дополнительного объема для служебных бит в модуле памяти определяется:

((22 бит * 256 строк) / (8 бит * 16 * 256 строк)) * 100% = 17.1%

Как видно из расчета, расходы на дополнительную память возрастают практически втрое по сравнению с предыдущим расчетом, сделанным для простейшей структуры кэш-памяти и в основном за счет достаточно большого адресного тэга.

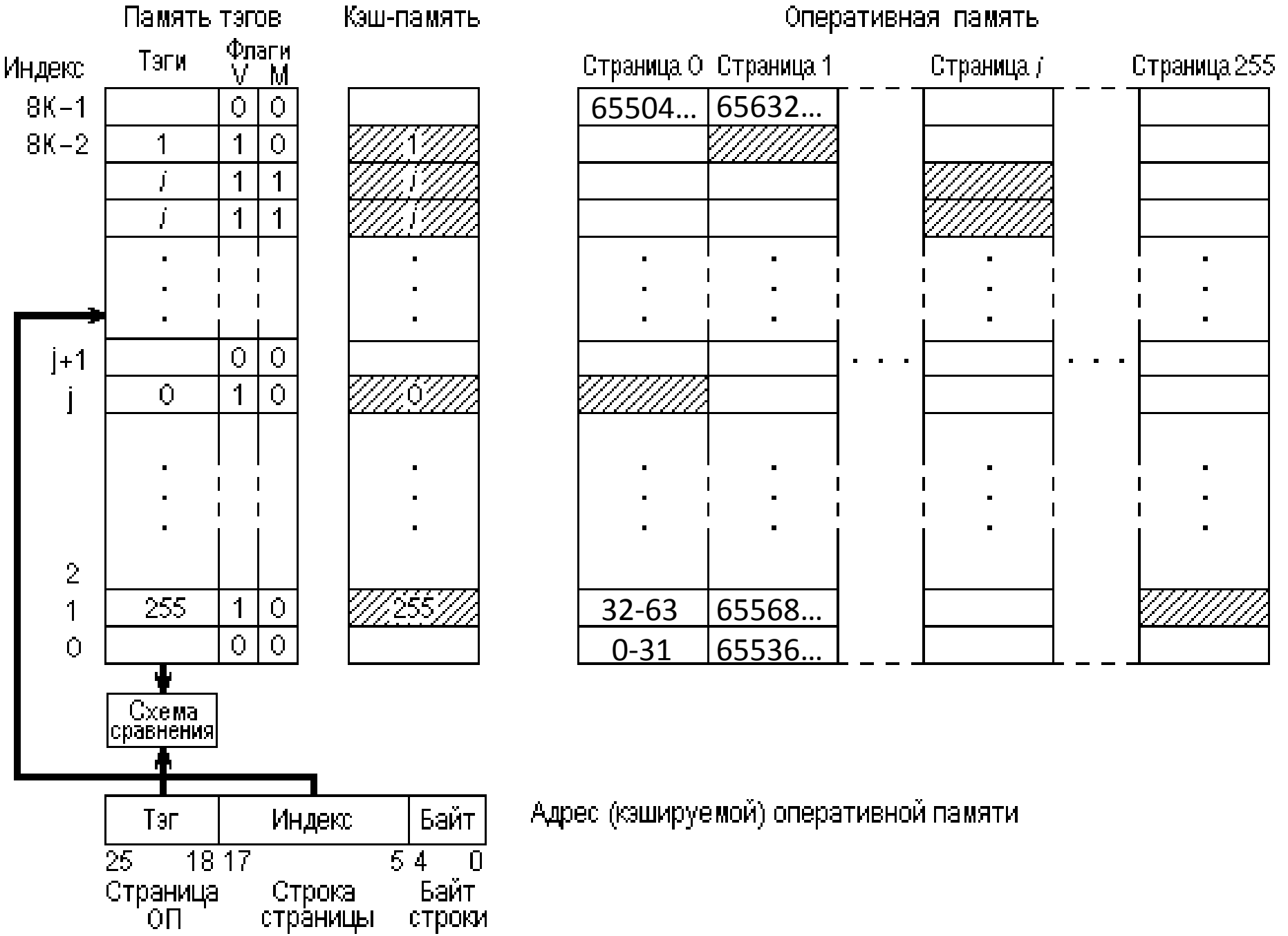
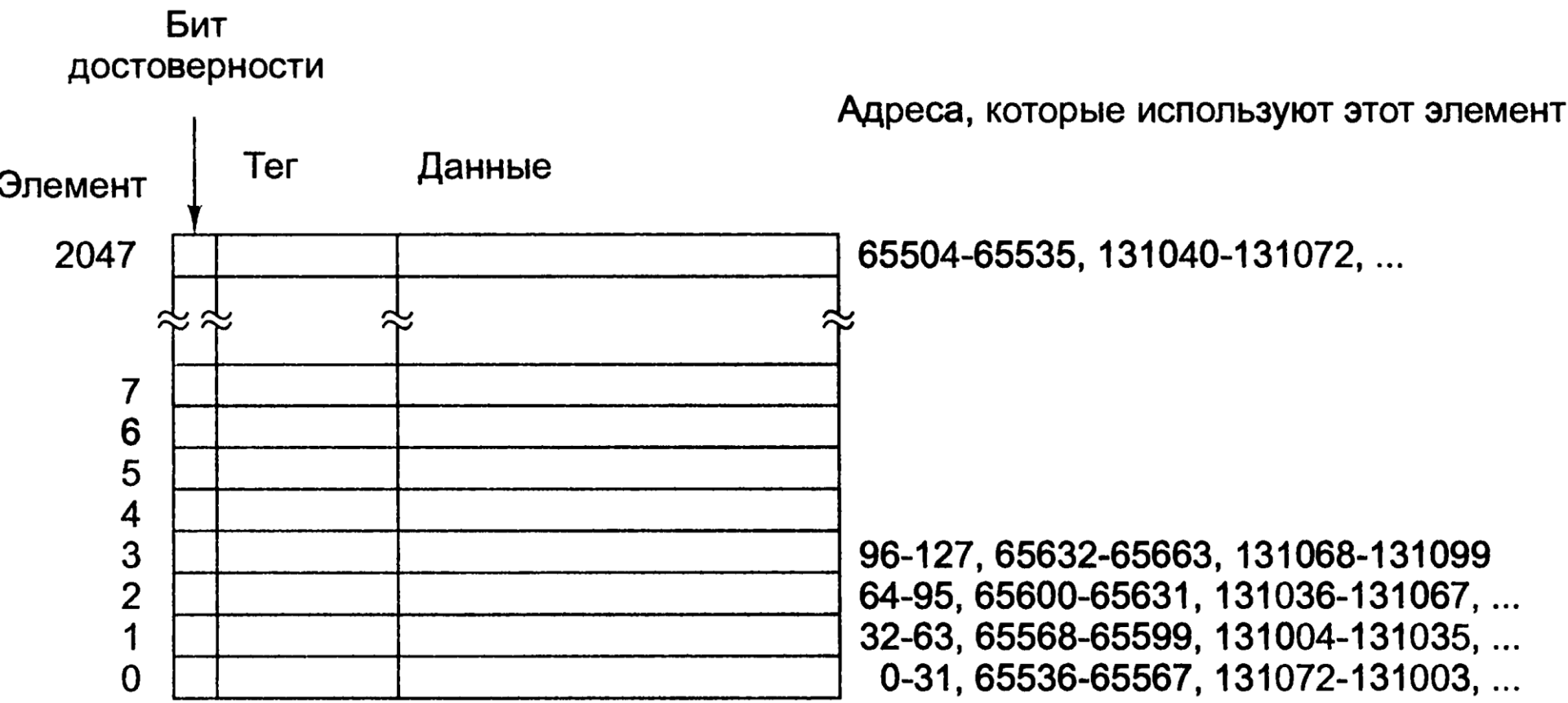
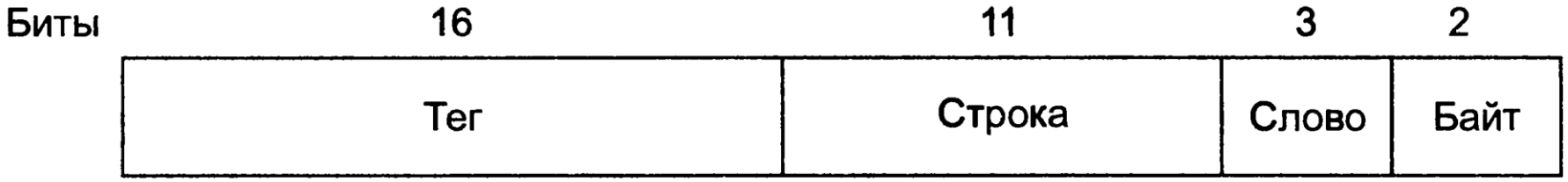


Рис. 40. Кэш-прямое отображение


Бит достоверности
Адреса, которые используют этот элемент
Элемент
Тег
Данные
2047
65504-65535, 131040-131072, ...
7
6
5
4
3
96-127, 65632-65663, 131068-131099
2
64-95, 65600-65631, 131036-131067, ...
1
32-63, 65568-65599, 131004-131035, ...
0
0-31, 65536-65567, 131072-131003, ...

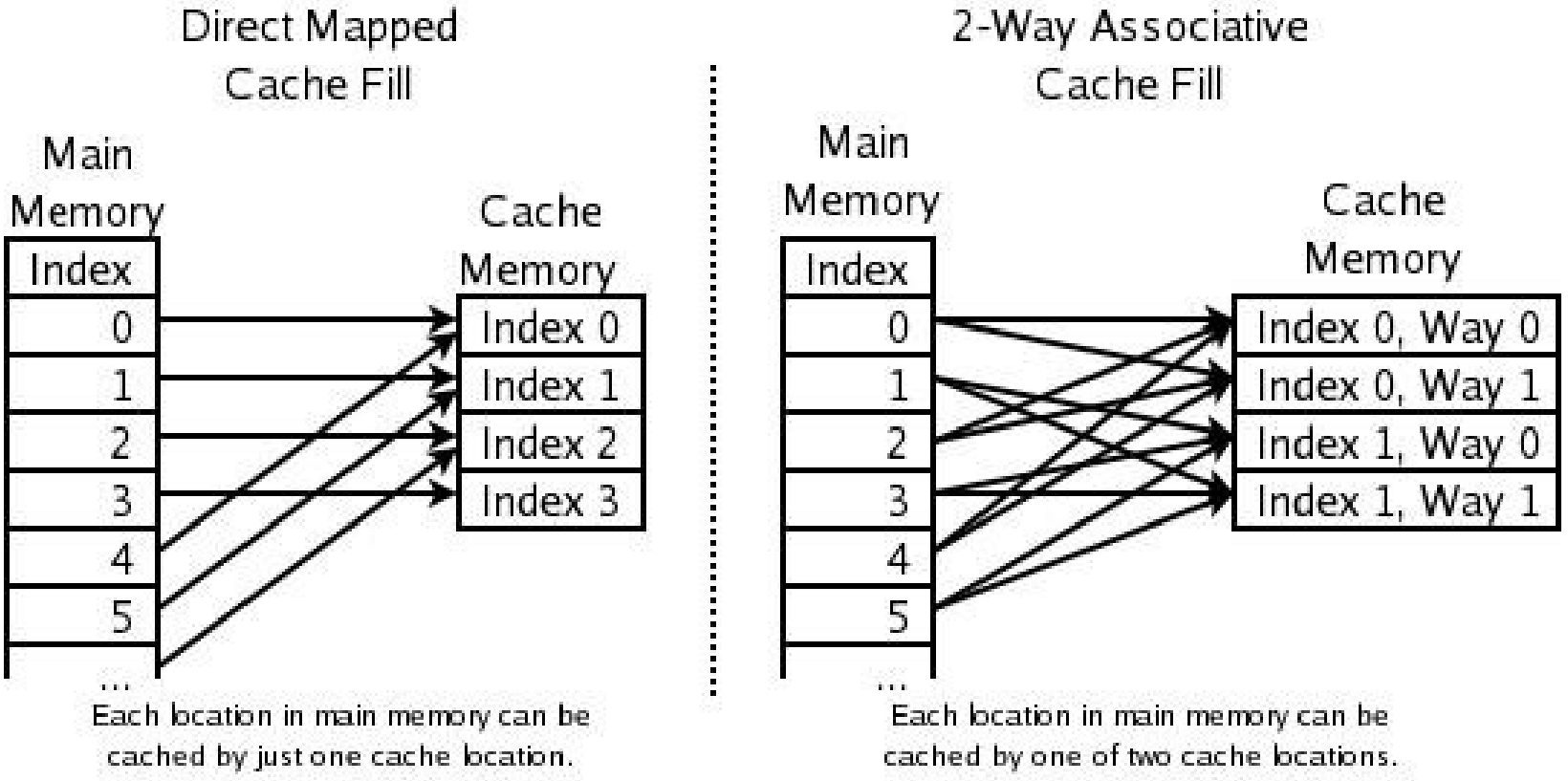
а


Биты
16
11
3
2
Тег
Строка
Слово
Байт

б

Виды кэш-памяти

- Кэш с прямым отображением
- Полностью ассоциативный кэш
- Частично ассоциативный кэш



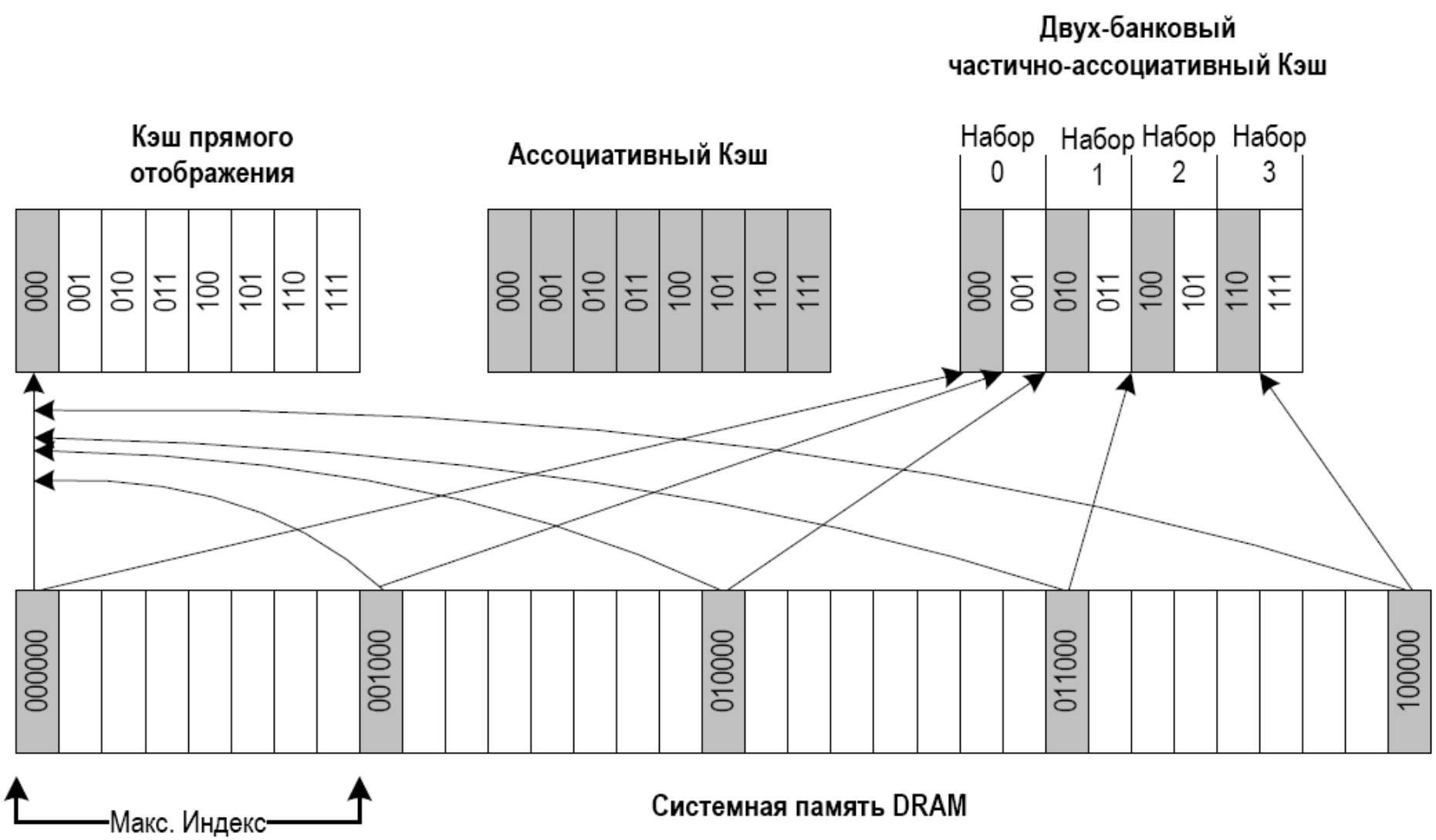


Рис. 4. Отображение блоков данных из основной памяти в кэш-память прямого отображения, ассоциативную кэш-память и частично-ассоциативную кэш-память.

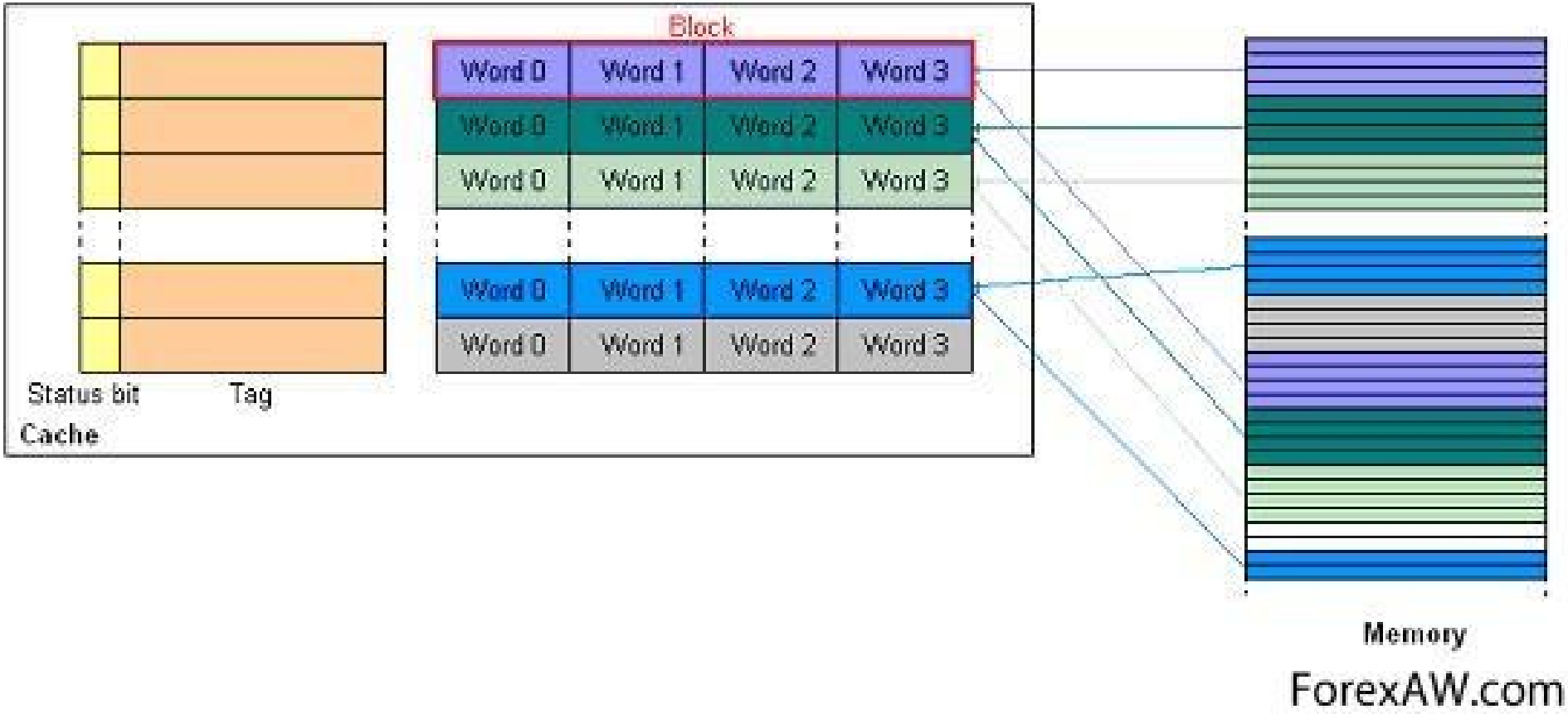
Таким образом, можно перечислить все три способа построения кэша с краткой характеристикой каждого из них:

- Кэш прямого отображения (**Direct mapped Cache**), где каждый блок данных из памяти может быть отображен только в одну строку кэша с определенным номером;
- Полно-ассоциативный кэш (**Fully Associative Cache**), где каждый блок данных из памяти может быть отображен в любую из из строк кэша;
- Частично-ассоциативный кэш (**Set Associative Cache**), где каждый блок данных может быть отображен в строку на выбор из нескольких параллельных наборов строк кэша (обычно 2, 4, 8 или 16). Частично-ассоциативный кэш может рассматриваться как набор из нескольких (2, 4, 8 или 16) параллельных кэшей прямого отображения, объединенных общей логикой управления.

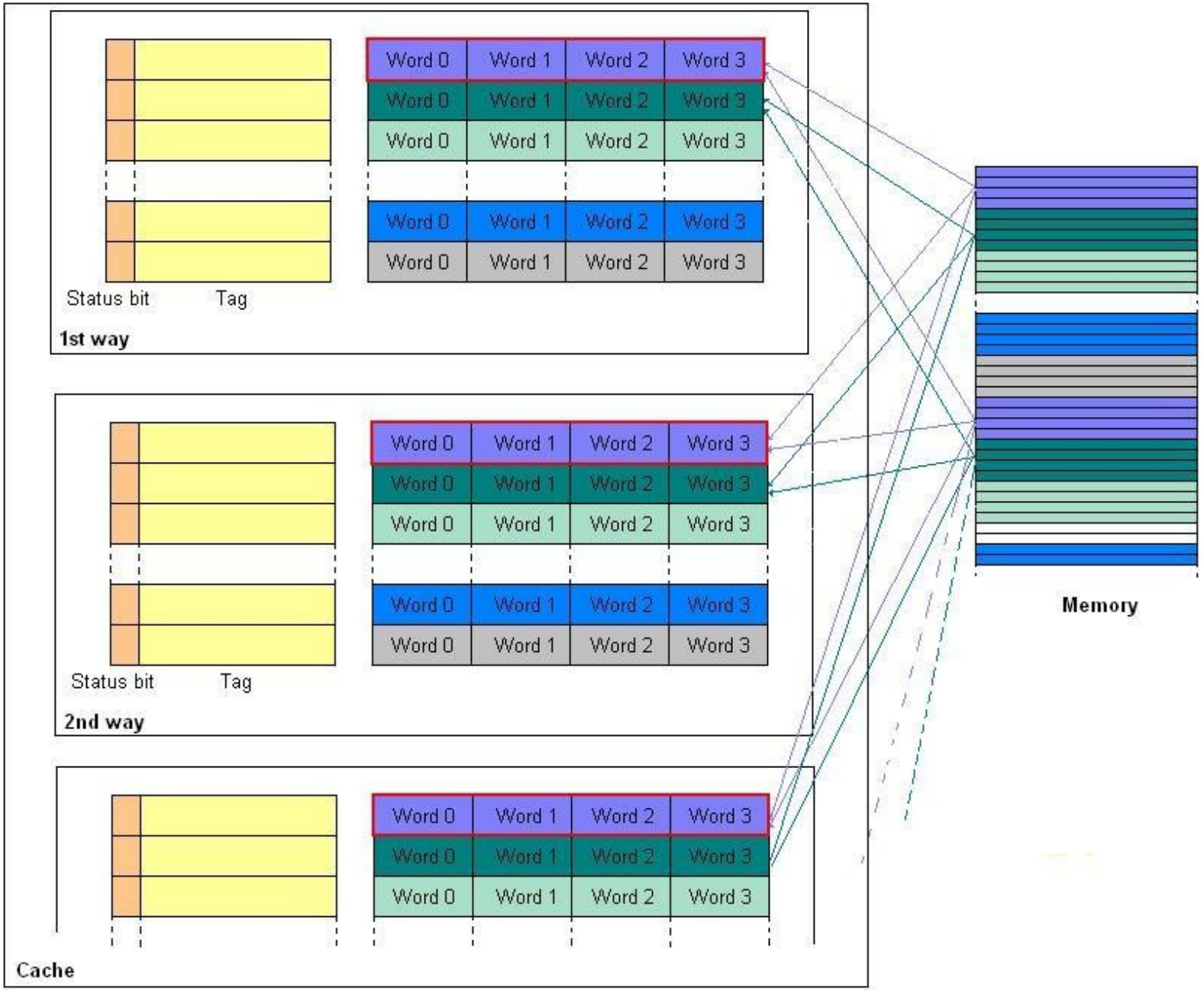
Принципы размещения блоков в кэш-памяти определяют три основных типа их организации:

- Если каждый блок основной памяти имеет только одно фиксированное место, на котором он может появиться в кэш-памяти, то такая кэш-память называется кэшем с прямым отображением (**direct mapped**). Это наиболее простая организация кэш-памяти, при которой для отображение адресов блоков основной памяти на адреса кэш-памяти просто используются младшие разряды адреса блока. Таким образом, все блоки основной памяти, имеющие одинаковые младшие разряды в своем адресе, попадают в один блок кэш-памяти, т.е. (адрес блока кэш-памяти) =(адрес блока основной памяти) mod (число блоков в кэш-памяти)
- Если некоторый блок основной памяти может располагаться на ограниченном множестве мест в кэш-памяти, то кэш называется множественно-ассоциативным (**set associative**). Обычно множество представляет собой группу из двух или большего числа блоков в кэше. Если множество состоит из **n** блоков, то такое размещение называется множественно-ассоциативным с **n** каналами (**n-way set associative**). Для размещения блока, прежде всего, необходимо определить множество. Множество определяется младшими разрядами адреса блока памяти (индексом): (адрес множества кэш-памяти) =(адрес блока основной памяти) mod (число множеств в кэш-памяти)
- Если некоторый блок основной памяти может располагаться на любом месте кэш-памяти, то кэш называется полностью ассоциативным (**fully associative**).

Кэш прямого отображения



N-канальный ассоциативный кэш

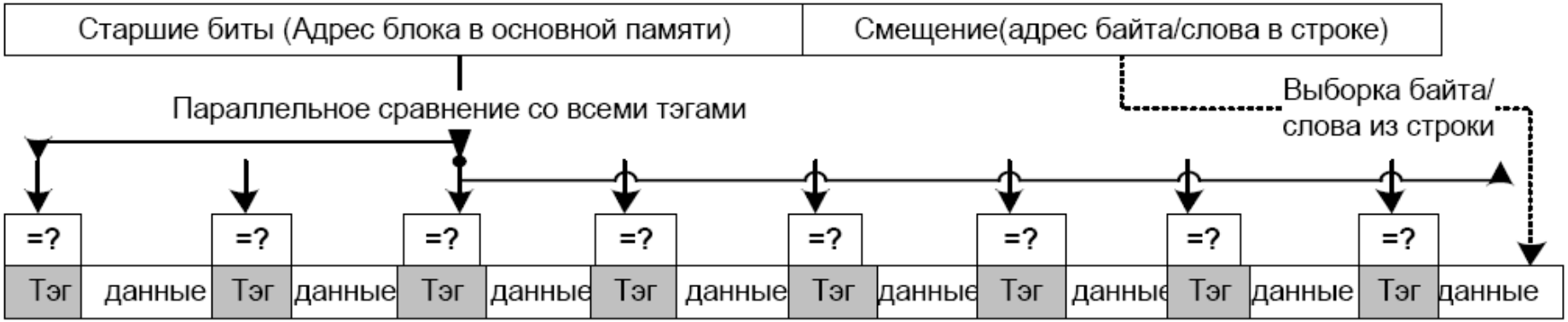


Сегмент тэгов				**Сегмент данных**				
Компаратор Адресного тэга с адресом ЦПУ (отдельный для каждой строки)	Адресный тэг блока данных (старшие биты адреса)	Флаги досто-верности и изменения данных V-флаг и D-флаг		Байт 3	Байт 2	Байт 1	Байт 0	Номер строки сегмента данных кэша
ЦПУ ?=Тэг 0	<30 бит>	<2 бит>		< 32 бит >				0
.....	.....	.....		.....				....
ЦПУ ?=Тэг 6	<30 бит>	<2 бит>		< 32 бит >				6
ЦПУ ?=Тэг 7	<30 бит>	<2 бит>		< 32 бит >				7

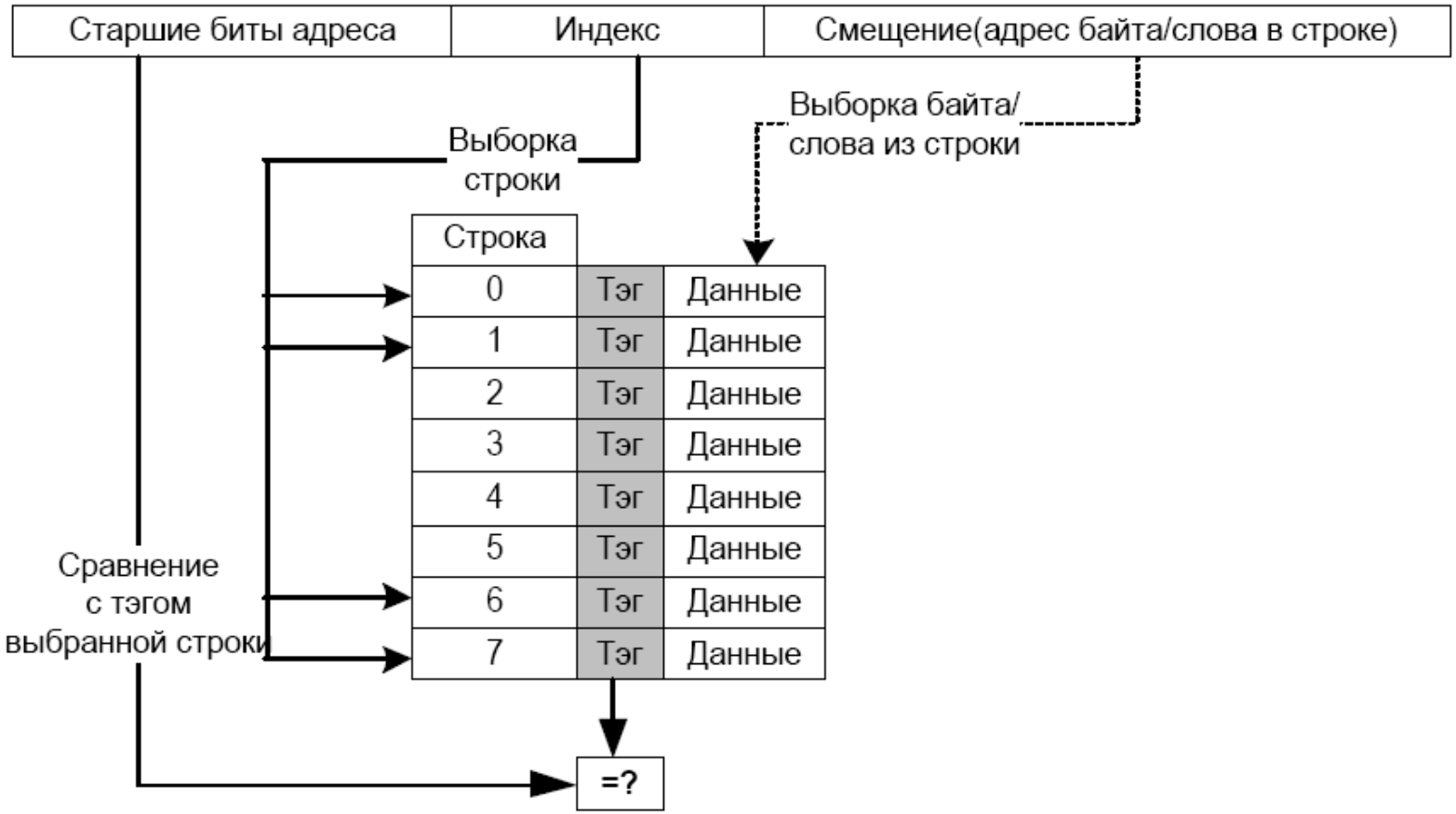
Старшие биты адреса <30 бит>	Смещение (Адрес байта в блоке) <2 бит>
Биты адреса [31:2]	*Биты адреса [1:0]*

Рис. 5. Структура ассоциативной кэш-памяти.

Формат адреса и доступ к полностью ассоциативному кэшу



Формат адреса и доступ к кэшу прямого отображения



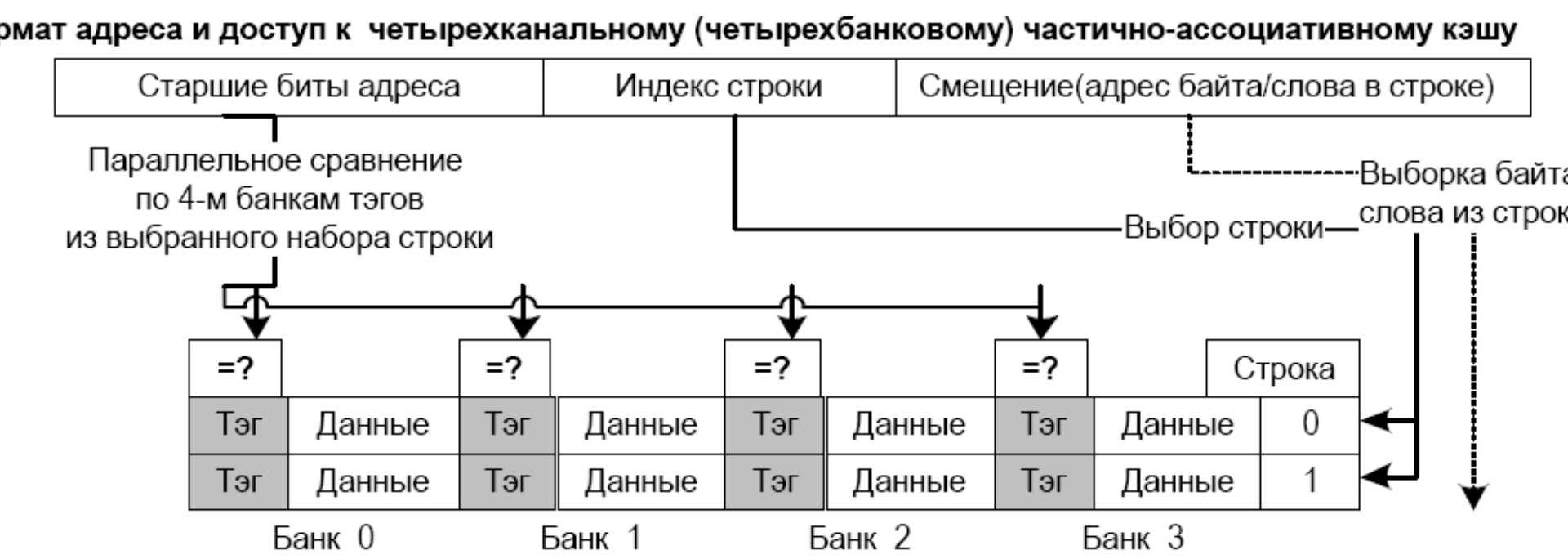
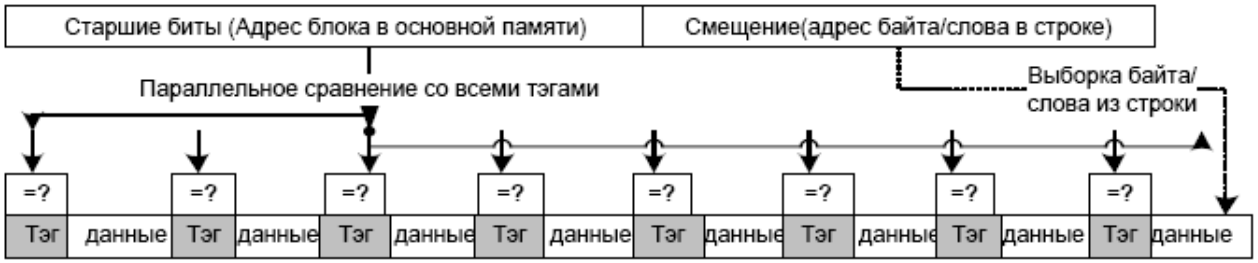
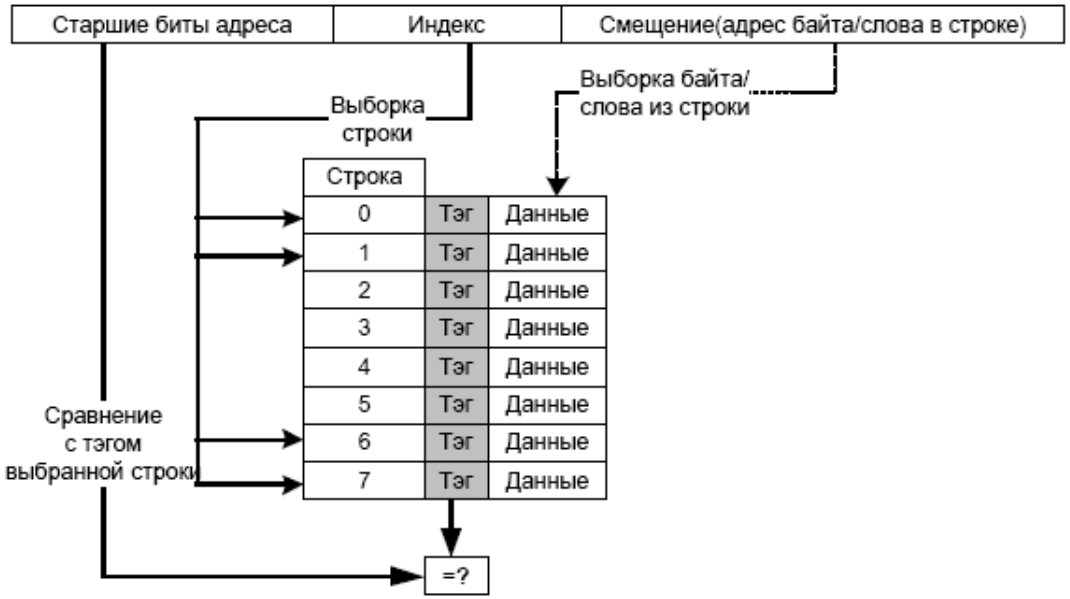


Рис. 6 проиллюстрированы отличия в доступе к ассоциативному кэшу в отличие от кэша прямого отображения. Если представить кэш-память как двумерный массив, то ассоциативный кэш выглядит как одна строка блоков, доступ к которым определяется сравнением полных адресных тэгов всех блоков. А кэш прямого отображения выглядит как столбец блоков, где доступ к отдельному блоку определяется индексом из поля адреса и не требует сравнения всех тэгов, а только одного тэга из блока, определенного значением индекса.

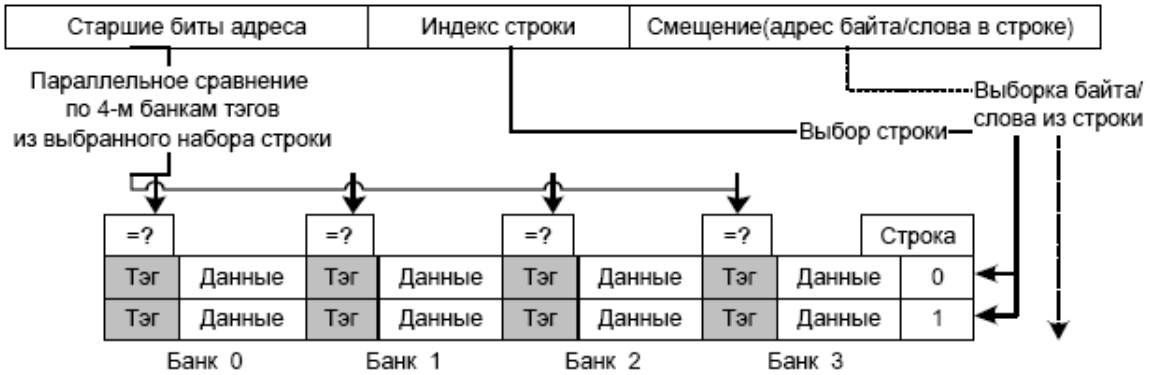
Формат адреса и доступ к полностью ассоциативному кэшу



Формат адреса и доступ к кэшу прямого отображения



Формат адреса и доступ к четырехканальному (четырехбанковому) частично-ассоциативному кэшу



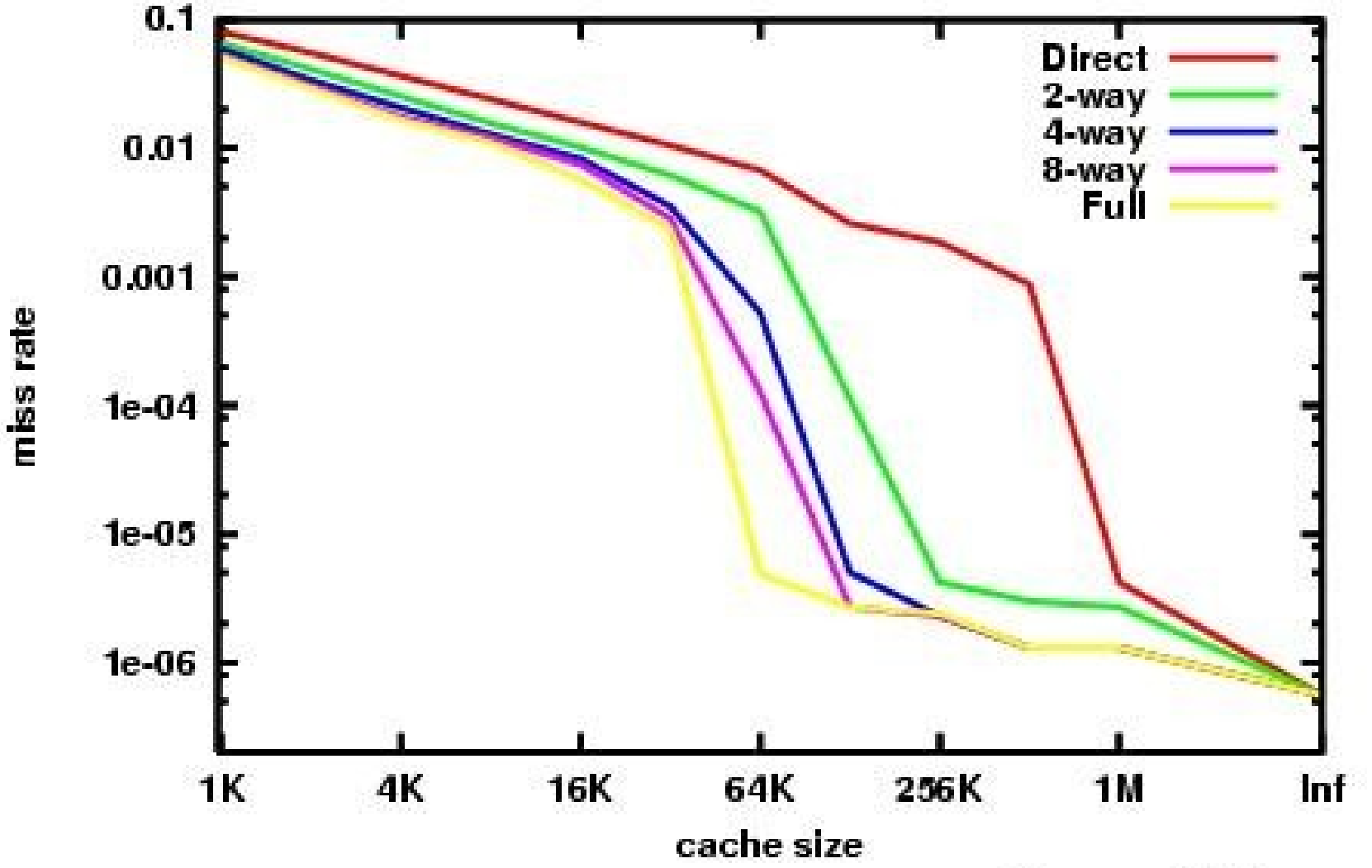

0.1
0.01
0.001
1e-04
1e-05
1e-06
miss rate
1K
4K
16K
64K
256K
1M
Inf
cache size
Direct
2-way
4-way
8-way
Full

Табл. 3. Зависимость скорости доступа в кэш-память от объема и степени ассоциативности

Степень ассоциативности и объем кэш-памяти	**Кэш прямого отображ-я**	**2-банковый частично ассоциатив. кэш**	**4-банковый частично ассоциатив. кэш**	**Полностью ассоциативн. кэш**
4 КВ	3	5.2	5.8	5.8
8 КВ	3	5.4	6	6
16 КВ	4	6	6.3	7
64 КВ	6	7	8.2	8.5
256 КВ	9	11.5	11.8	14

КРАТКО О ТИПАХ КЭША

1) Ассоциативный кэш: выборка блока производится только по значению полного адресного тэга (паралелльная или псевдо-параллельная проверка всех блоков). Поля индекса для таких кэшей не существует.

2) Кэш прямого отбражения: выборка строки (блока) по значению поля индекса, далее считанный из этой строки адресный тэг сравнивается с адресом, полученным от ЦП.

3) Частично-ассоциативный кэш с N банками или каналами: выборка всех банков одновременно производится по значению поля набора (аналогичен индексу), выбор нужного банка производится путем сравнения N адресных тэгов с адресом, полученным от ЦП. Далее по тексту будет использоваться понятие **резидентности** данных по отношению к тому или иному уровню иерархии памяти. Резидентность блока данных означает, что этот блок скопирован (перемещен) в кэш. Понятие резидентности также распространяется на блоки данных в других уровнях иерархии памяти.

2.3. Основные параметры кэш-памяти

Можно выделить следующие основные параметры, которые характеризуют кэш-память различных типов:

- Объем кэш-памяти (сегмент данных и сегмент адресных тэгов)
- Размер строки (блока) кэш-памяти
- Степень (уровень) ассоциативности кэш-памяти (Прямое отображение, ассоциативный, частично-ассоциативный)
- Алгоритм замещения данных (выбор строки для удаления) при необходимости загрузки новых данных в кэш из основной памяти (LRU, Random)
- Алгоритм записи данных в кэш-память из ЦП (кэш с обратной записью и кэш со сквозной записью)

Алгоритм вытеснения

Если список свободных буферов пуст, то выполняется алгоритм вытеснения буфера. Алгоритмов вытеснения существенно влияет на производительность кэша. Существуют следующие алгоритмы:

- **LRU (Least Recently Used)** — вытесняется буфер, неиспользованный дольше всех;
- **MRU (Most Recently Used)** — вытесняется последний использованный буфер;
- **LFU (Least Frequently Used)** — вытесняется буфер, использованный реже всех;
- **ARC (англ.) (Adaptive Replacement Cache)** — алгоритм вытеснения, комбинирующий LRU и LFU, запатентованный IBM.

Применение того или иного алгоритма зависит от стратегии кэширования данных. LRU наиболее эффективен, если данные гарантированно будут повторно использованы в ближайшее время. MRU наиболее эффективен, если данные гарантированно не будут повторно использованы в ближайшее время. В случае, если приложение явно указывает стратегию кэширования для некоторого набора данных, то кэш будет функционировать наиболее эффективно.

Полный адрес данных в основной памяти
на 4 G байта с 32-бит адресным пространством

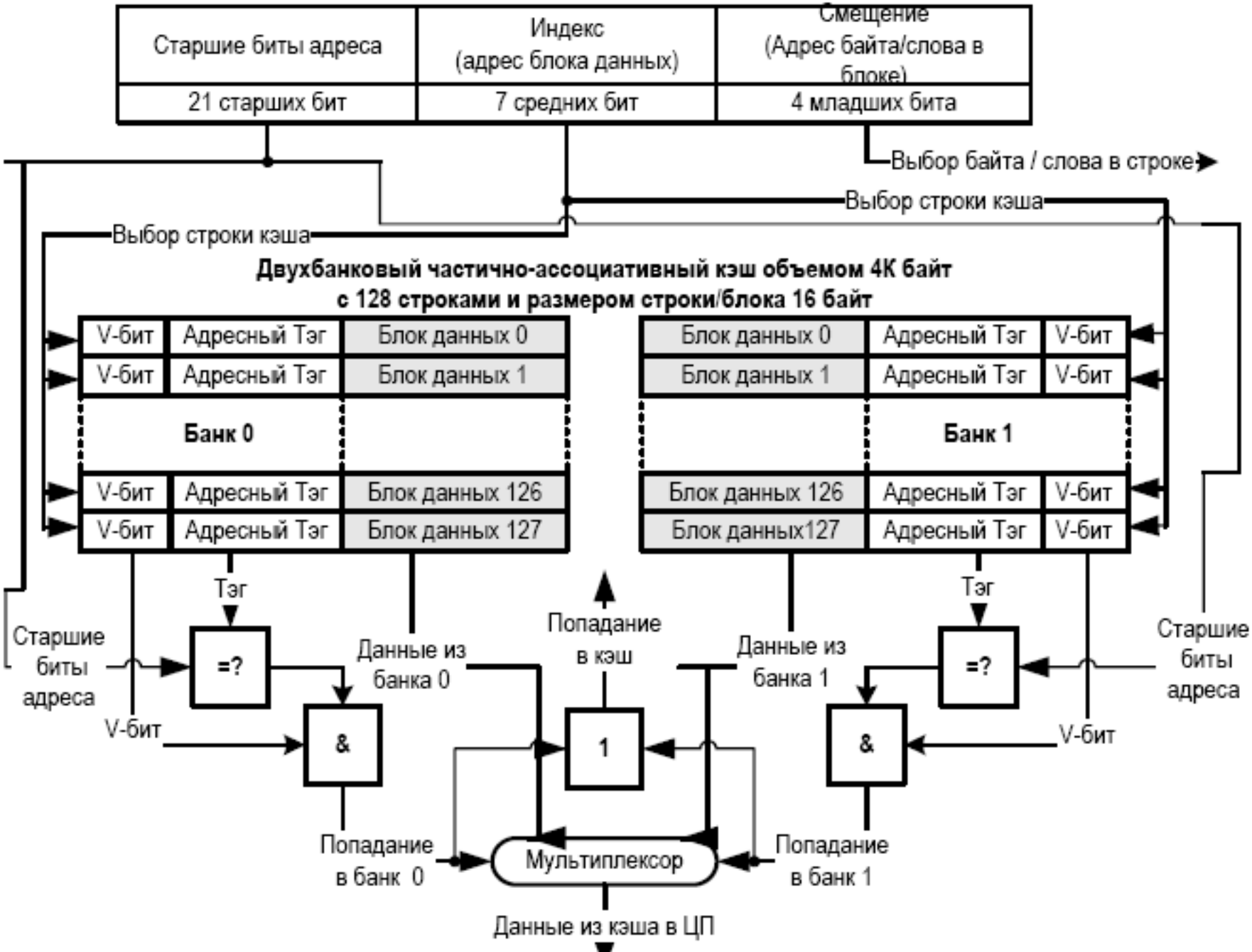


Рис. 8. Кэш команд процессора Альфа 21264, использующий предсказание доступа

3.4. Способы уменьшения доли промахов при обращении в кэш

Можно перечислить следующие общепринятые способы уменьшения доли промахов, которые можно отнести к специальным усовершенствованиям архитектуры:

- Увеличение объема кэш-памяти
- Увеличение размера блока (строки) кэша;
- Увеличение степени ассоциативности кэша;
- Использование дополнительного **кэша замещения** (**victim cache**);
- Использование **псевдо-ассоциативного кэша** (**pseudo-associative cache**);
- Использование **аппаратной предварительной подкачки** данных (**hardware prefetching**);
- **Реорганизация кода** компилятором для повышения локальности доступа
- Использование **предварительной подкачки под управлением компилятора** (**compiler generated prefetching**).
- Использование вспомогательного кэша замещения

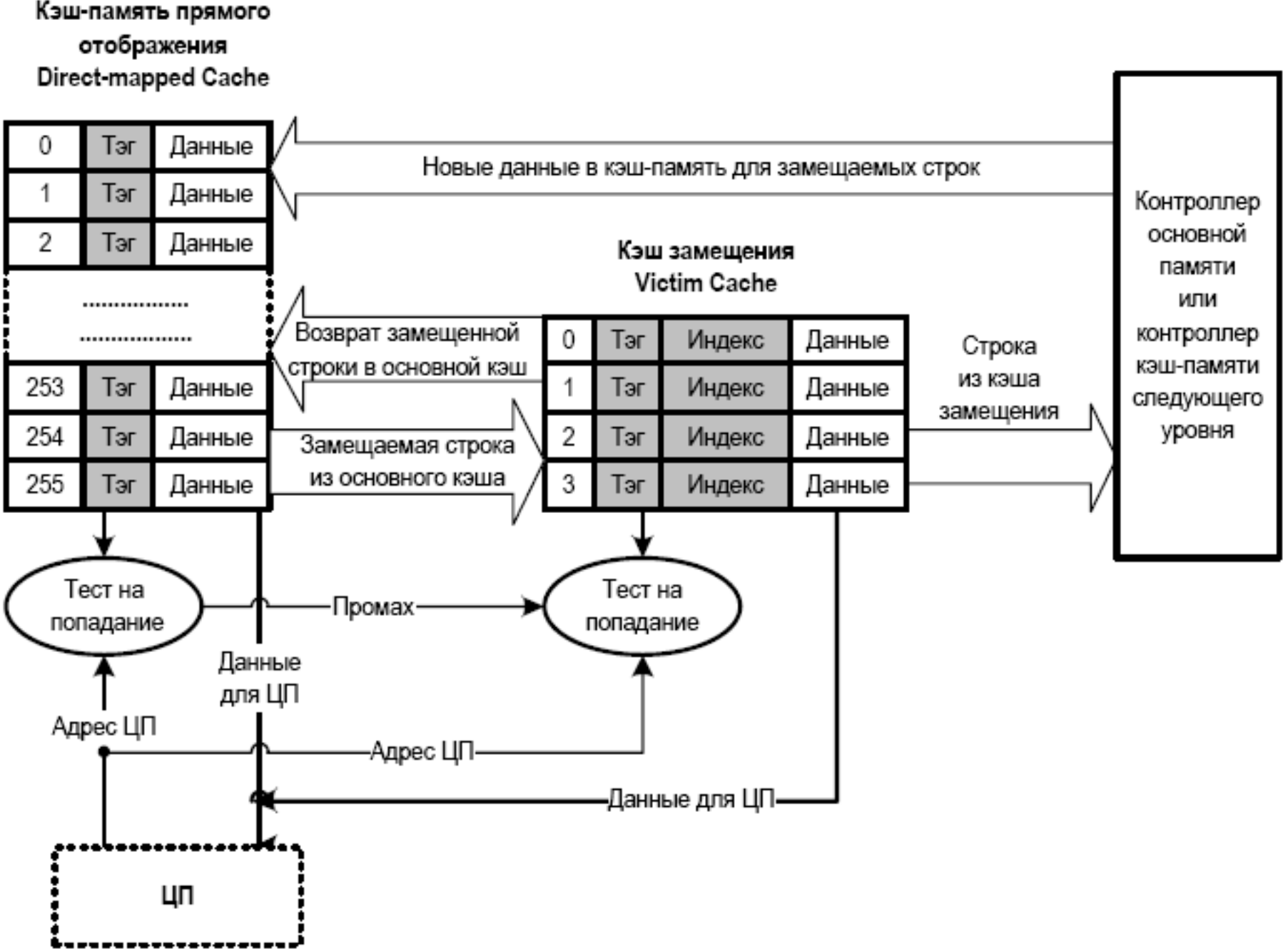
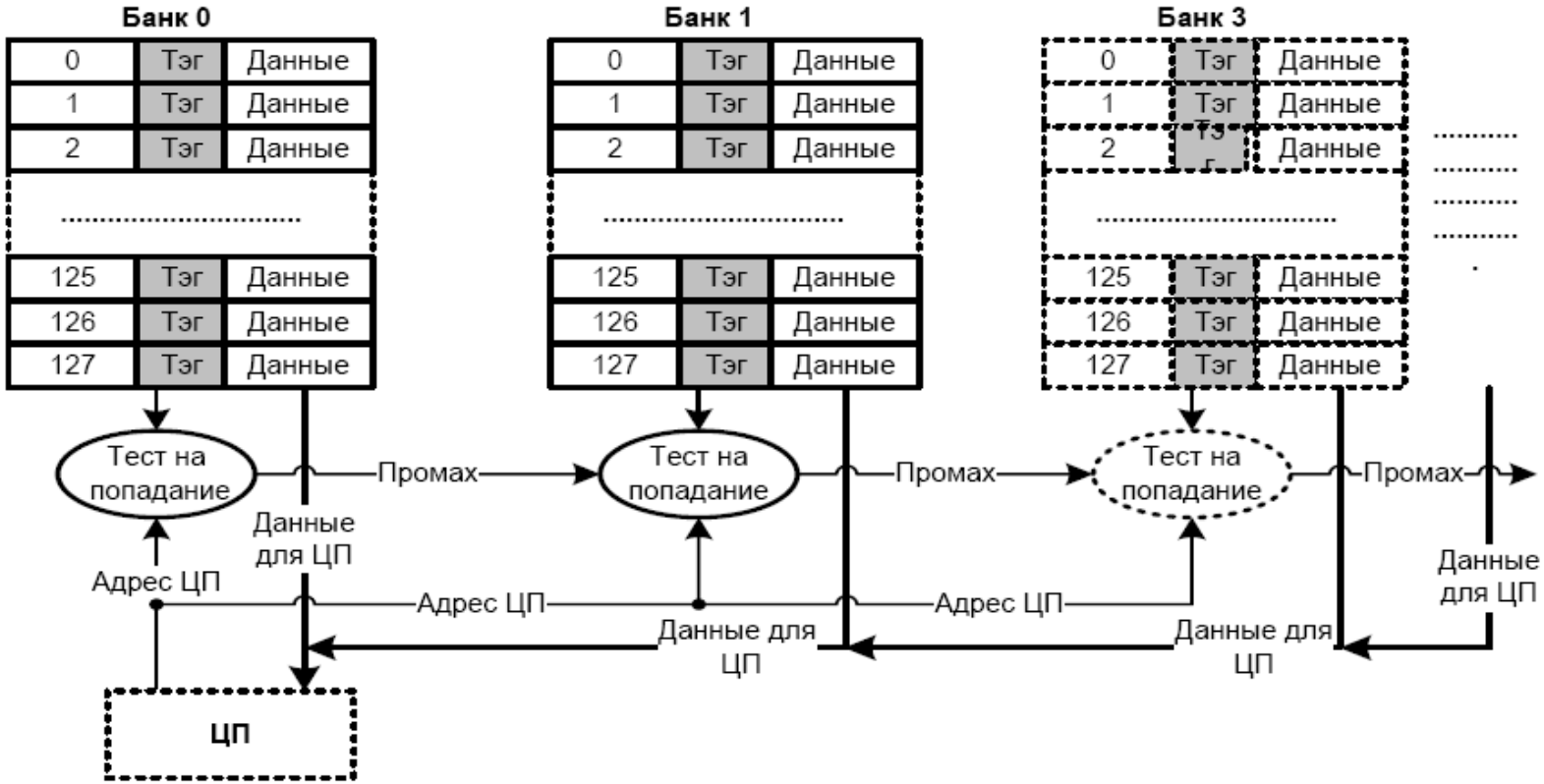


Рис. 9. Использование кэша замещения для уменьшения

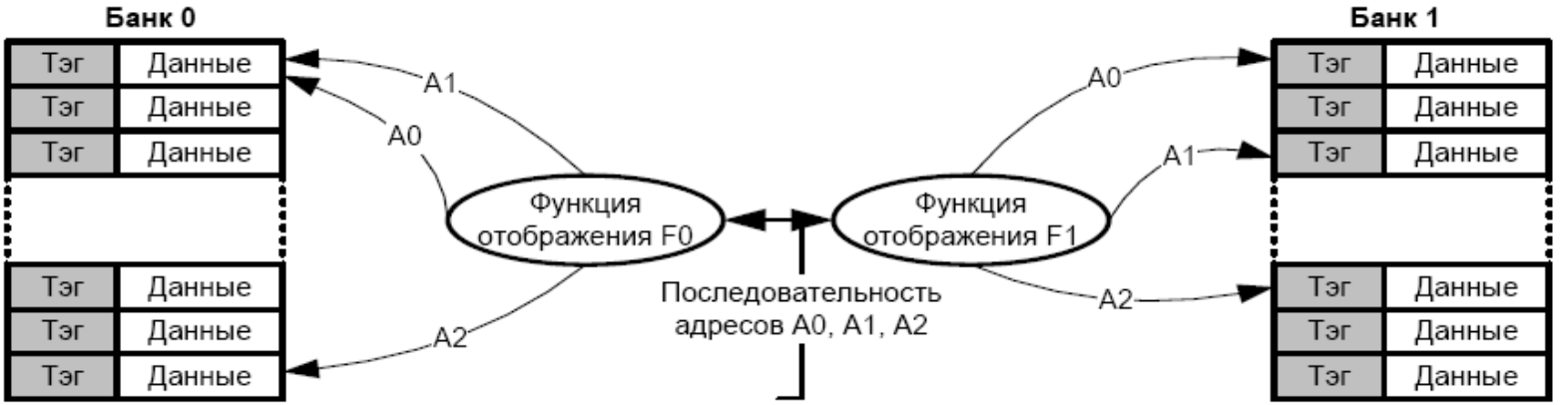
3.4.3. Использование псевдо-ассоциативного кэша

Использование псевдо-ассоциативного кэша тоже является одной из возможностей уменьшить число конфликтов отображения, сохранив простоту и быстродействие кэш-памяти прямого отображения. Наиболее простым способом является разделение кэша прямого отображения на две одинаковые половины с уменьшением поля индекса на один бит. Фактически мы имеем два последовательно проверяемых на попадание кэша прямого отображения. При тесте на попадание вначале тестируется первая половина и в случае промаха – вторая половина. Это увеличивает задержку попадания T_hit обычно на один такт , так как как надо провести только дополнительное сравнение уже считанных тэгов с обеих половин кэша. В принципе вышеописанных подход можно расширить и разделить кэш на 4 части и проводить тест на попадание последовательно начиная первой части. Такая конструкция называется вертикально-ассоциативным кэшем (**column-associative cache**) и имеет один серьезный недостаток: число тактов при доступе может варьироваться от одного до нескольких в зависимости от числа столбцов в таком кэше. Эта особенность не позволяет использовать такие кэши на первом уровне L1 вместе с ЦП. Более приемлемым вариантом является использование вертикально-ассоциативных кэшей на втором уровне L2, когда они не соединены непосредственно с ЦП. Время доступа T_hit в в вертикально-ассоциативном кэше минимально и близко к кэшу прямого отображения, в тоже время доля попаданий такая же, как в частично-ассоциативном кэше с 2-мя или 4-мя банками.

Вертикально-ассоциативный Кэш (Column-associative Cache)



Псевдо-ассоциативный кэш с несимметричным отображением в банках (Skewed-associative Cache)



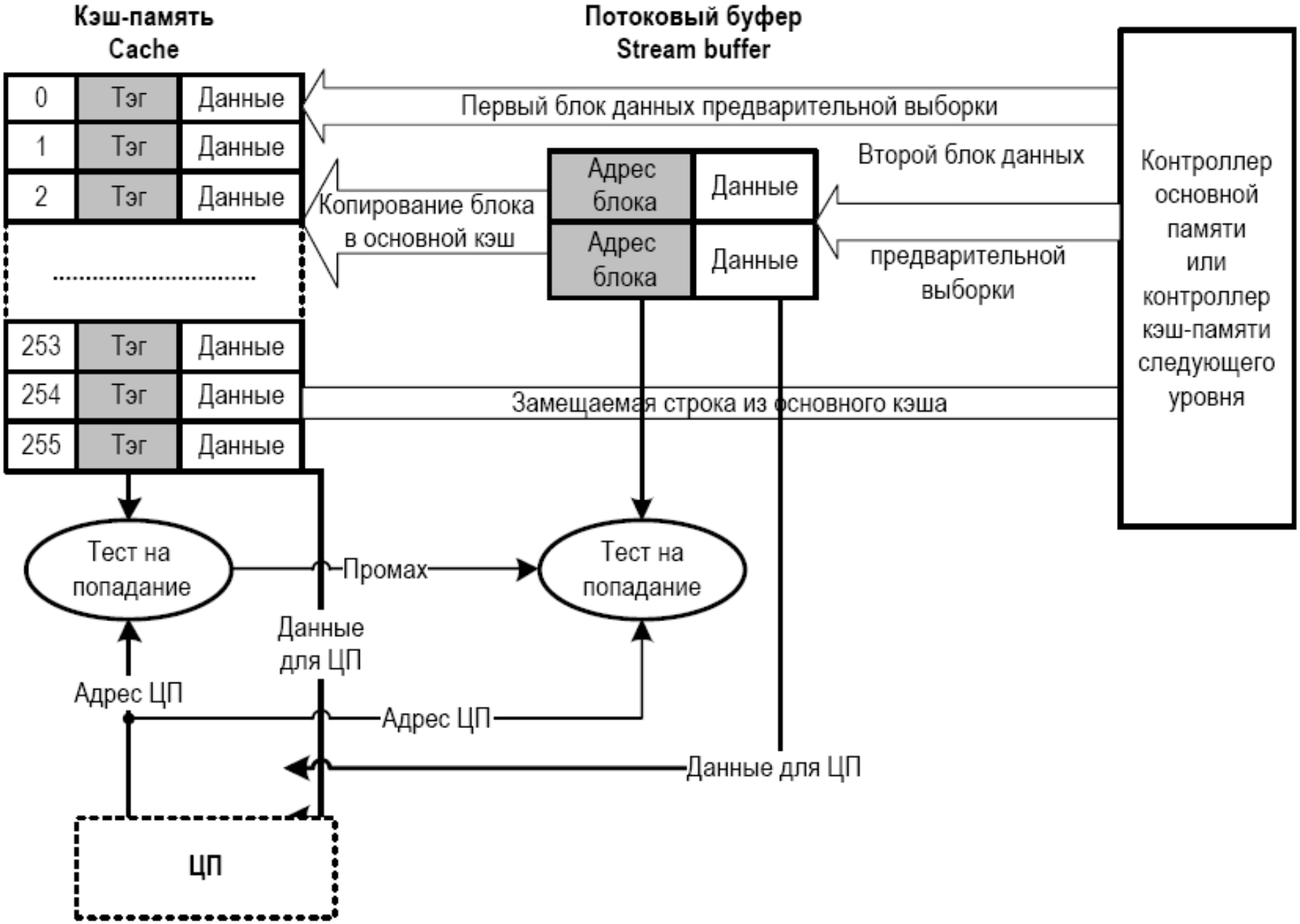


Рис. 11. Структура кэша с потоковым буфером

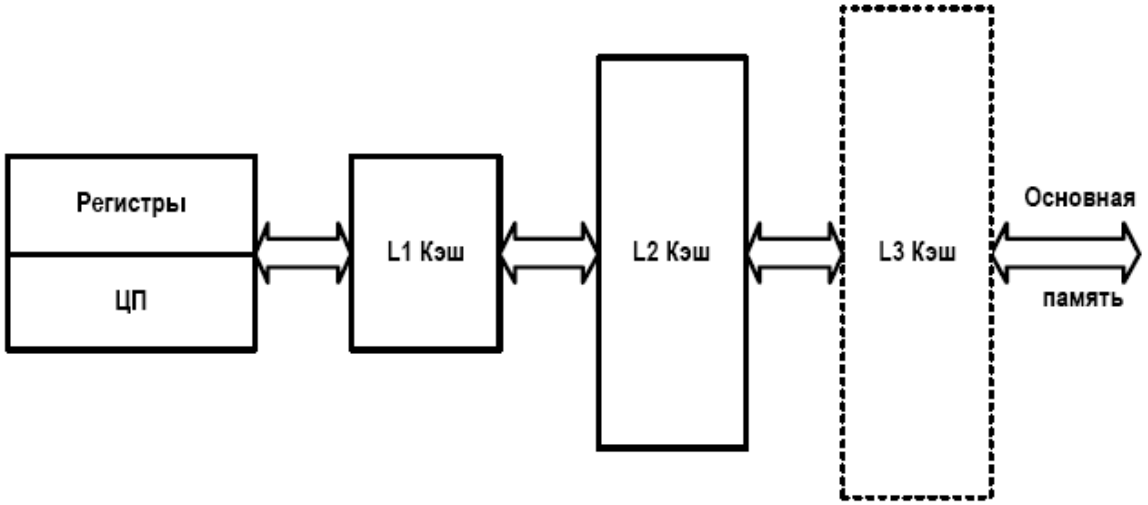


Рис. 12. Использование двух-уровневой кэш-памяти.

Тогда выражение для среднего времени доступа кэша L1 примет вид:

*T_average = T_hit(L1) + Miss_Rate(L1) * T_miss(L1)*

Где *T_average* – среднее время доступа к памяти, *Miss_Rate(L1)* – доля промахов в кэше L1, *T_miss(L1)* – задержка промаха для кэша L1.

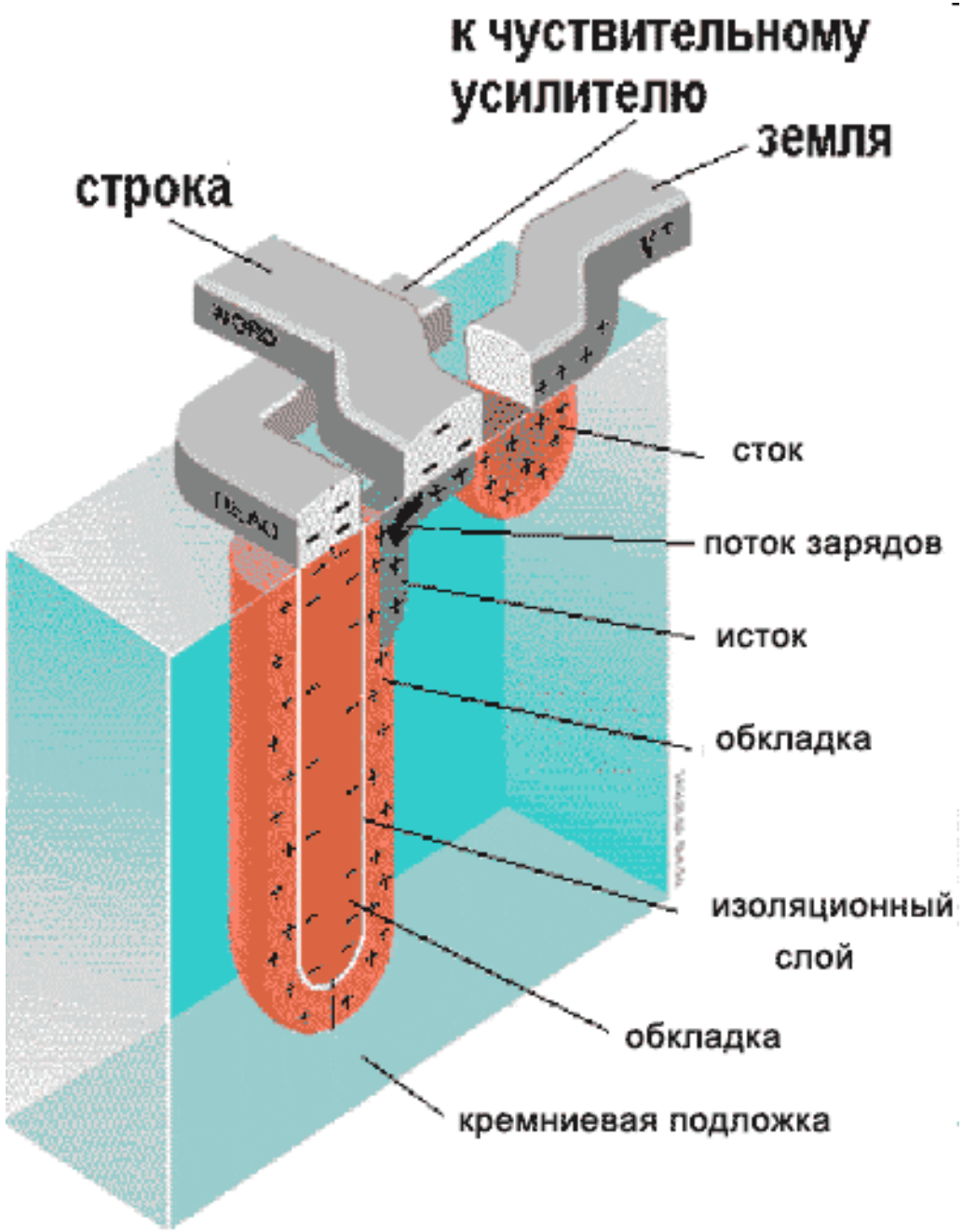
Учитывая, что кэш L1 теперь соединен с кэшем L2, можно выразить задержку промаха через аналогичную формулу для кэша L2:

*T_miss(L1) = T_hit(L2) + Miss_Rate(L2) * T_Miss(L2)*

Где *T_miss(L1)* – среднее время доступа к памяти, *Miss_Rate(L2)* – доля промахов в кэше L2, *T_miss(L2)* – задержка промаха для кэша L2.

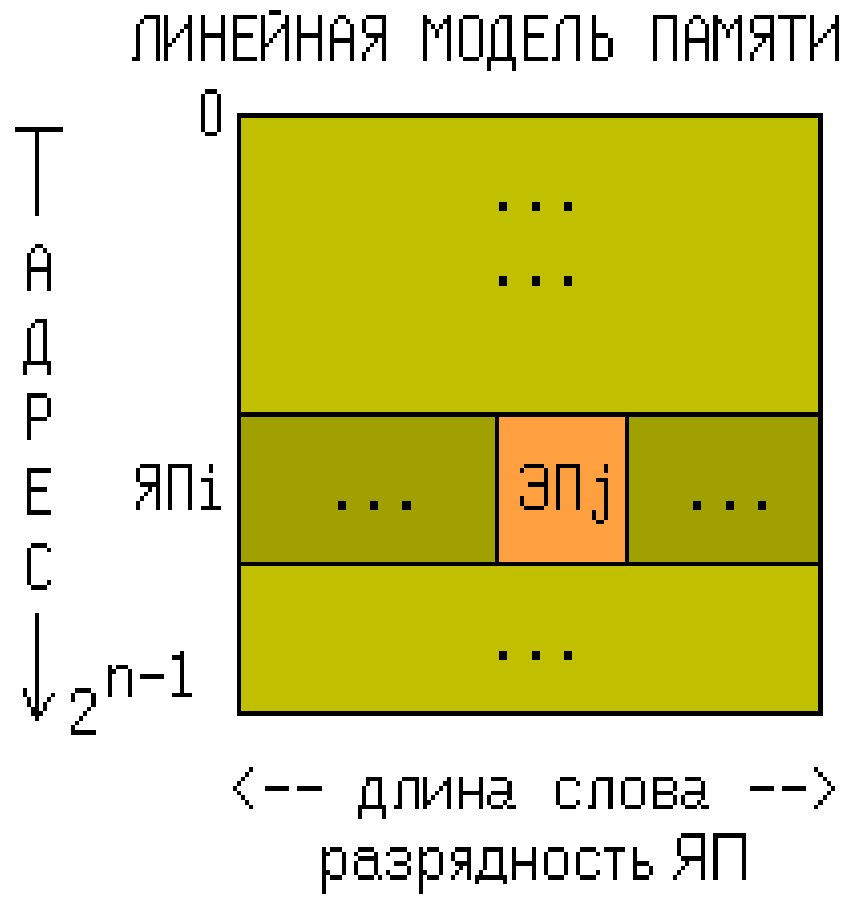
Подставив даное выражение в уравнение для среднего времени доступа, получаем выражение для среднего времени доступа к памяти в системе с двухуровневым кэшем:

*T_average = T_hit(L1) + Miss_Rate(L1) * [T_hit(L2) + Miss_Rate(L2) * T_miss(L2)]*

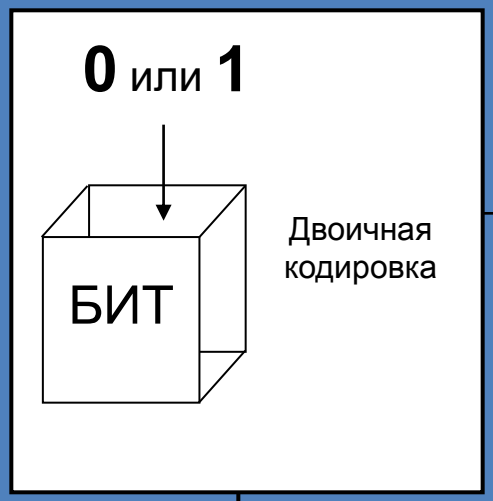


Устройство ячейки динамической памяти

.....	....	....
512	2^9	0,5K
1024	2^{10}	1K
2048	2^{11}	2K
4096	2^{12}	4K
8192	2^{13}	8K
16384	2^{14}	16K
32768	2^{15}	32K
65536	2^{16}	64K(но не 65K !)
.....	....	....



ВНУТРЕННЯЯ ПАМЯТЬ КОМПЬЮТЕРА



Байты	Биты							
0	0	1	0	1	1	0	0	0
1	0	1	0	0	1	1	0	1
2	1	0	1	1	0	1	1	0
3	0	0	1	0	1	1	0	0
. . . . .								

Дискретность

- Внутренняя память состоит из частиц – битов
- В одном бите памяти хранится один бит информации

Адресуемость

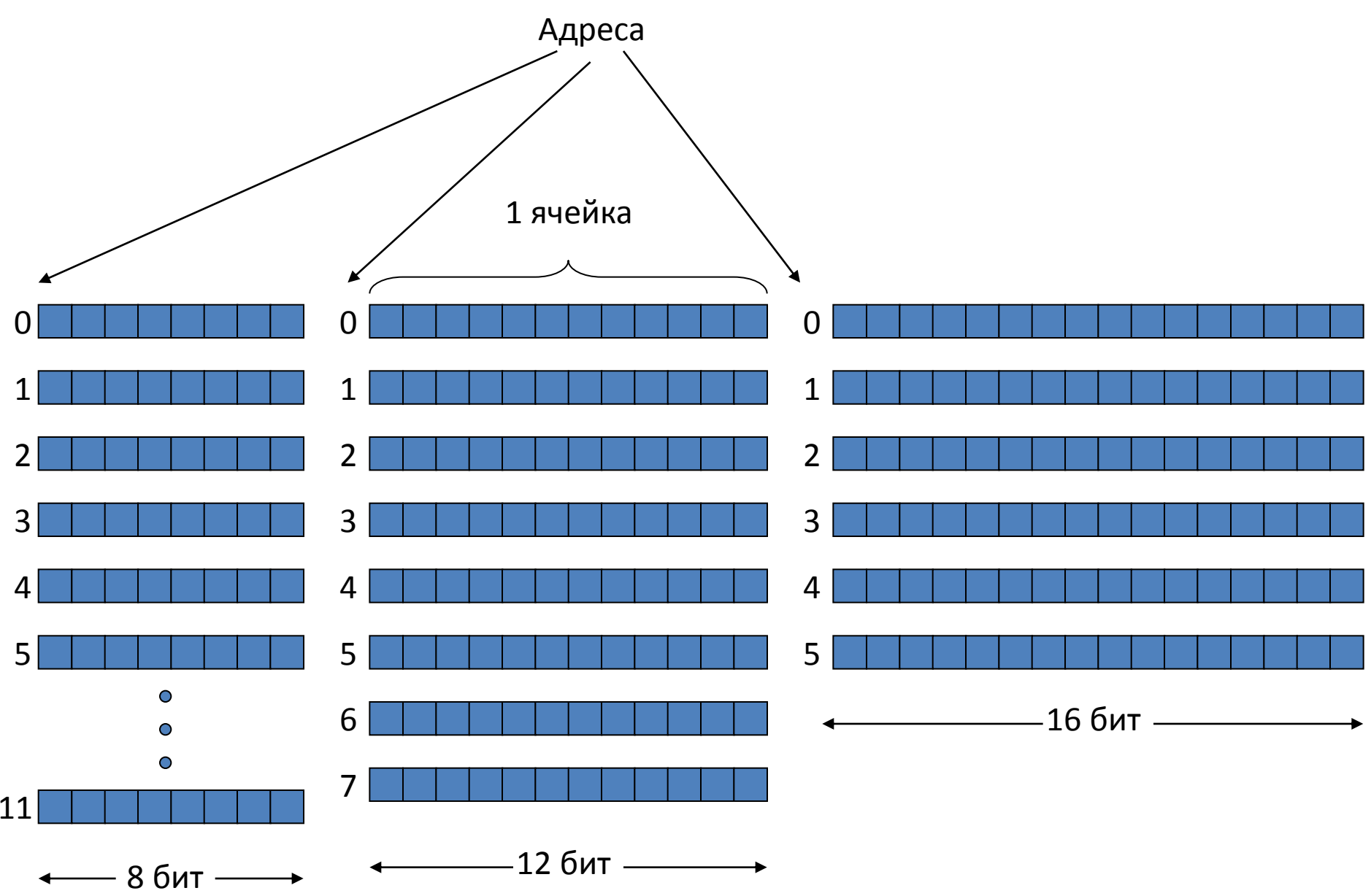
- Байт памяти – наименьшая адресуемая часть внутренней памяти (1 байт = 8 бит)
- Все байты пронумерованы, начиная от 0
- Номер байта – адрес байта памяти
- Процессор обращается к памяти по адресам

Адреса

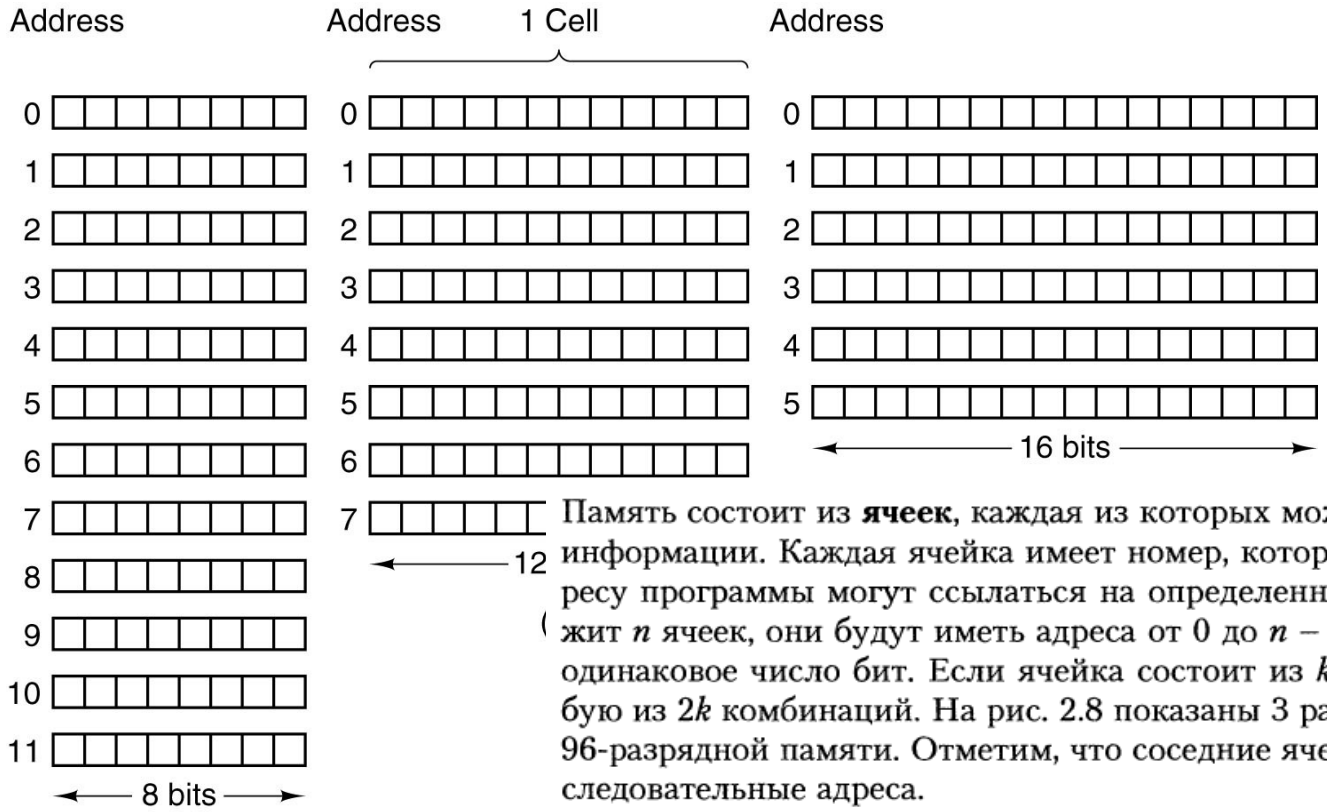
Память состоит из ячеек, каждая из которых может хранить некоторую порцию информации.

Каждая ячейка имеет номер, который называется адресом.

Все ячейки памяти содержат одинаковое число битов. Адреса также представляют из себя двоичное число. Количество битов в адресе определяет максимальное число адресованных ячеек, что в совокупность с размером ячейки определяет максимальный объем адресуемой памяти.



Пример: три способа организации 96-битной памяти



Память состоит из **ячеек**, каждая из которых может хранить некоторую порцию информации. Каждая ячейка имеет номер, который называется **адресом**. По адресу программы могут ссылаться на определенную ячейку. Если память содержит n ячеек, они будут иметь адреса от 0 до $n - 1$. Все ячейки памяти содержат одинаковое число бит. Если ячейка состоит из k бит, она может содержать любую из $2k$ комбинаций. На рис. 2.8 показаны 3 различных варианта организации 96-разрядной памяти. Отметим, что соседние ячейки по определению имеют последовательные адреса.

В компьютерах, в которых используется двоичная система счисления (включая восьмеричное и шестнадцатеричное представление двоичных чисел), адреса памяти также выражаются в двоичных числах. Если адрес состоит из m бит, максимальное число адресованных ячеек составит $2m$. Например, адрес для обращения к памяти, изображенной на рис. 2.8, *а*, должен состоять по крайней мере из 4 бит, чтобы выражать все числа от 0 до 11. При устройстве памяти, показанном на рис. 2.8, *б* и 2.8, *в*, достаточно 3-разрядного адреса. Число бит в адресе определяет максимальное количество адресуемых ячеек памяти и не зависит от числа бит в ячейке. 12-разрядные адреса нужны и памяти из 212 ячеек по 8 бит каждая, и памяти из 212 ячеек по 64 бит каждая.

Машинное слово

Ячейка - минимальная единица к которой можно обращаться.

Большинство современных машин оперирует с 8-битными ячейками памяти. 8 бит = 1 байт.

Байты формируют слова. Слово - максимальная единица памяти с которой оперирует большинство команд машины.

Например, 32-битная машина содержит 32-битные регистры и манипулирует с машинными словами из 4 байт, 64-битная соответственно содержит 64-битные регистры и манипулирует со словами из 8 байт.

Таблица 2.1. Число бит в ячейке памяти некоторых моделей коммерческих компьютеров

Компьютер	Число битов в ячейке
Burroughs B1700	1
IBM PC	8
DEC PDP-8	12
IBM 1130	16
DEC PDP-15	18
XDS 940	24
Electrologica X8	27
XDS Sigma 9	32
Honeywell 6180	36
CDC 3600	48
CDC Cyber	60

ЛОГИЧЕСКАЯ СТРУКТУРА ОПЕРАТИВНОЙ ПАМЯТИ

Оперативная память представляет собой множество ячеек.

Каждая ячейка имеет свой уникальный **адрес**.

Нумерация ячеек начинается с **нуля**.

Каждая ячейка памяти имеет объем **1 байт**.

Максимальный объем адресуемой памяти равен произведению количества ячеек N на 1 байт.

Для процессоров Pentium 4 (разрядность шины адреса = 36 бит) максимальный объем адресуемой памяти равен:

$$N \times 1 \text{ байт} = 2^I \times 1 \text{ байт} = 2^{36} \times 1 \text{ байт} = 68\,719\,476\,736 \text{ байт} =$$
$$= 67\,108\,864 \text{ Кбайт} = 65\,536 \text{ Мбайт} = \mathbf{64\ Гбайт}$$

Объем памяти	Ячейки	Десятичный адрес ячейки	Шестнадцатеричный адрес ячейки
64 Гбайт	10101010	68 719 476 735	FFFFFFFFF
…	…	…	…
4 Гбайт	10101010	4 294 967 295	FFFFFFFF
…	…	…	…
	10101010	0	0

Порядок байт

Байты в словах могут нумероваться слева направо или справа налево. Кажется, что никакой разницы в это нет, но на самом деле это не так!

Попробуем представить как будет храниться фраза “школа номер 495” в памяти.

При этом пусть число 495 представлено в виде int:
(00000000 00000000 00000001 11101111)

л	о	к	ш	0	0	ш	к	о	л
о	н		а	4	4	а		н	о
	р	е	м	8	8	м	е	р	
0	0	1	239	12	12	0	0	1	239

Нумерация справа налево

(прямой порядок)

Нумерация слева направо

Проблема

Если мы попробуем скопировать такие данные по одному байту с одной машины на другую, то вместо номера 495 получится номер 4009820160

(11101111 00000001 00000000 00000000). Простого решения у этой проблемы не существует!

Разобравшись с устройством и работой ядра памяти, перейдем к рассмотрению ее **интерфейса**.

В первую очередь выделим среди них **линии адреса** и **линии данных**. Линии адреса, как и следует из их названия, служат для выбора адреса ячейки памяти, а линии данных - для чтения и для записи ее содержимого. Необходимый режим работы определяется состоянием специального вывода ***W**rite **E**nable (Разрешение Записи)*.

Низкий уровень сигнала WE готовит микросхему к считыванию состояния линий данных и записи полученной информации в соответствующую ячейку, а высокий, наоборот, заставляет считать содержимое ячейки и "выплюнуть" его значения в линии данных.

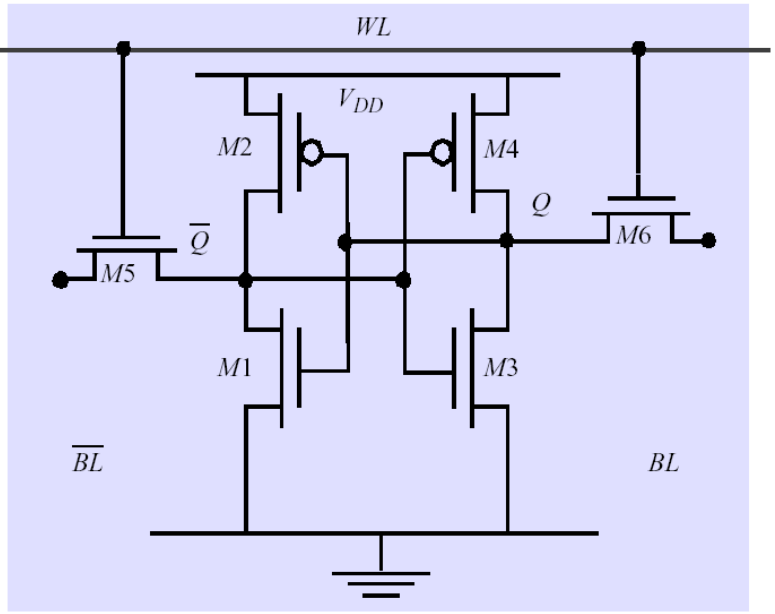
Совмещение выводов микросхемы увеличивает скорость обмена с памятью, но не позволяет осуществлять чтение и запись одновременно. Размещенные внутри кристалла процессора микросхемы кэш-памятина количество ножек не скупятся и беспрепятственно считывают ячейку во время записи другой).

Основными разновидностями статической памяти (SRAM) с точки зрения организации ее функционирования являются асинхронная (*Asynchronous*), синхронная пакетная (*Synchronous Burst*) и синхронная конвейерно-пакетная (*Pipeline Burst*) память.

Первой появилась асинхронная память, Интерфейс этой памяти включает шины данных, адреса и управления. В состав сигналов последней входят:
CS# (*Chip Select*) – сигнал выбора микросхемы;
WE# (*Write Enable*) – сигнал разрешения записи;
OE# (*Output Enable*) – сигнал включения выходов для выдачи данных.

Все сигналы управления инверсные, т.е. их активный (вызывающий соответствующее действие) уровень низкий. При единичном значении сигнала ***OE***# выход микросхемы переходит в состояние высокого выходного сопротивления.

Статическая оперативная память с произвольным доступом (*SRAM, static random access memory*)



Типичная ячейка статической двоичной памяти (двоичный триггер) на КМОП-технологии состоит из двух перекрёстно (кольцом) включённых инверторов и ключевых транзисторов для обеспечения доступа к ячейке (рис. 1.). Часто для увеличения плотности упаковки элементов на кристалле в качестве нагрузки применяют поликремниевые резисторы. Недостатком такого решения является рост статического энергопотребления.

Линия WL (Word Line) управляет двумя транзисторами доступа. Линии BL и BL (Bit Line) — битовые линии, используются и для записи данных, и для чтения данных.
Запись. При подаче «0» на линию BL или BL параллельно включенные транзисторные пары (M5 и M1) и (M6 и M3) образуют логические схемы 2ИЛИ, последующая подача «1» на линию WL открывает транзистор M5 или M6, что приводит к соответствующему переключению триггера.
Чтение. При подаче «1» на линию WL открываются транзисторы M5 и M6, уровни записанные в триггере выставляются на линии BL и BL и попадают на схемы чтения.

Статическая оперативная память с произвольным доступом (*SRAM, static random access memory*)

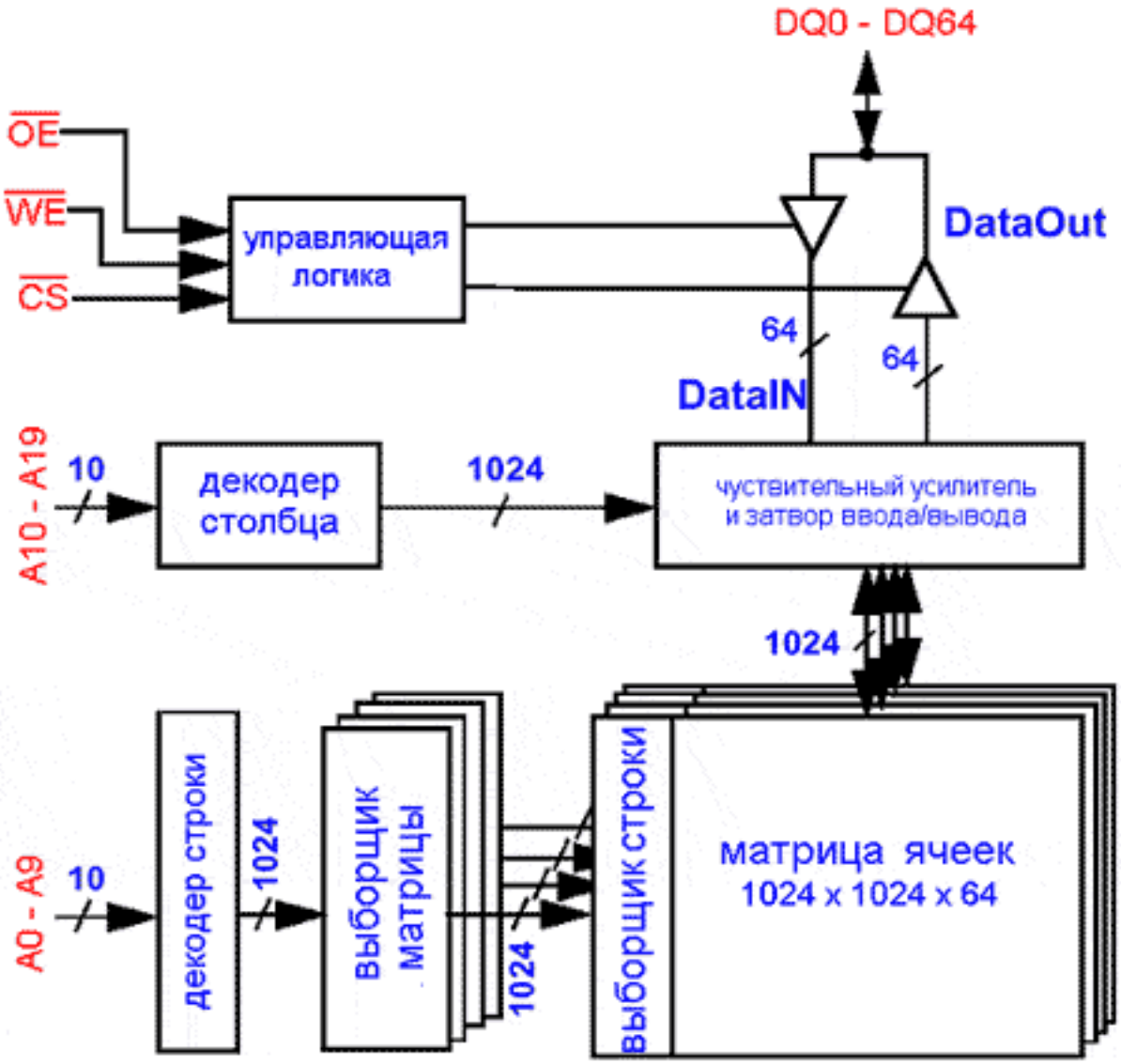
Преимущества

1. Быстрый доступ. SRAM — это действительно память *произвольного* доступа, доступ к любой ячейке памяти в любой момент занимает одно и то же время.
2. Простая схемотехника — SRAM не требуются сложные контроллеры.
3. Возможны очень низкие частоты синхронизации, вплоть до полной остановки синхроимпульсов.

Недостатки

1. Невысокая плотность записи (шесть-восемь элементов на бит вместо двух у DRAM).
2. Вследствие чего — дороговизна килобайта памяти.
3. Тем не менее, высокое энергопотребление не является принципиальной особенностью SRAM, оно обусловлено высокими скоростями обмена с данным видом внутренней памяти процессора. Энергия потребляется только в момент изменения информации в ячейке SRAM.

Устройство типовой микросхемы SRAM-памяти



OE (Output Enable)
WE (Write Enable)

Изображение микросхемы памяти

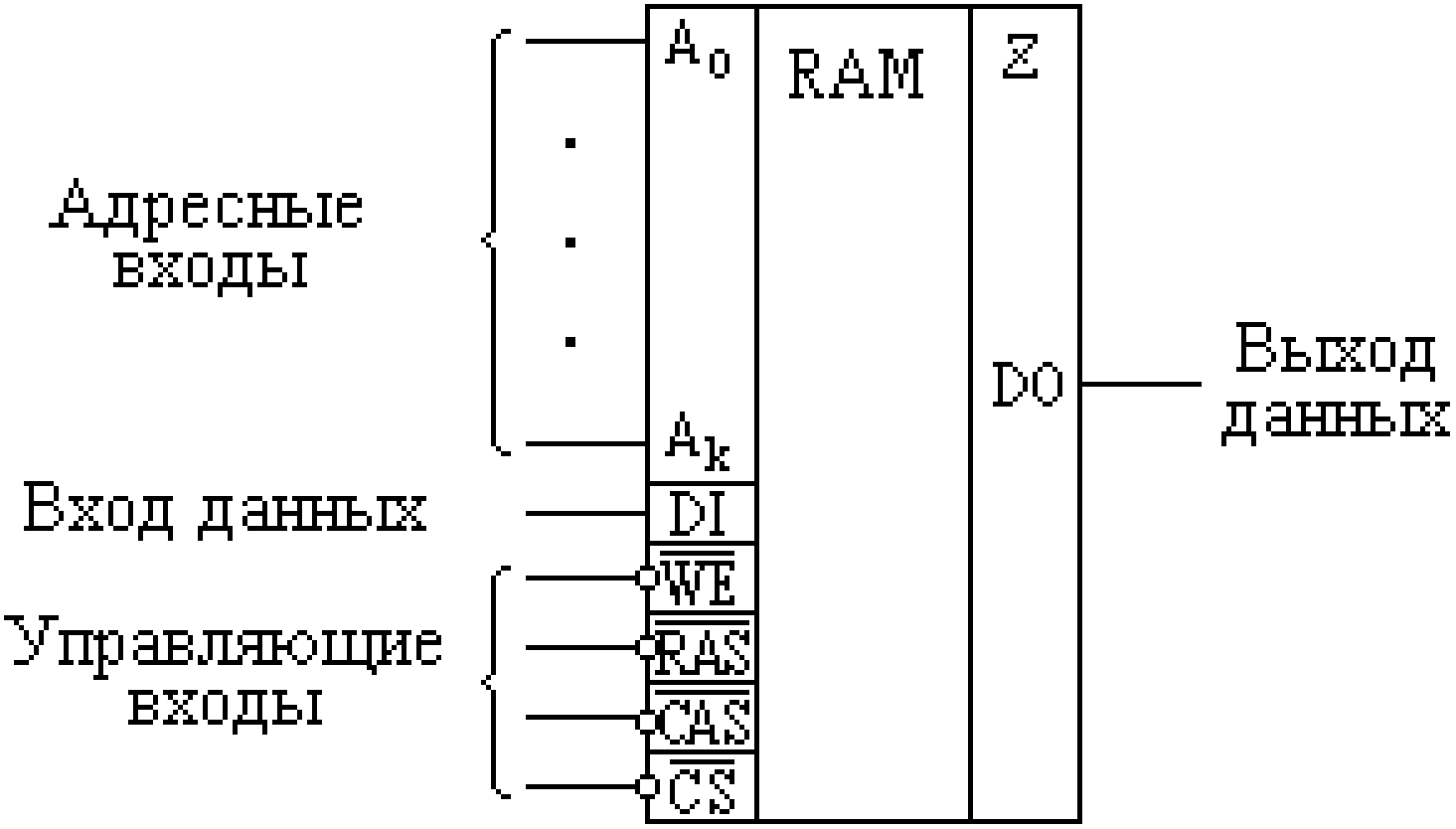


Рис. 9. Условное изображение ОЗУ на функциональных схемах

Синхронная пакетная статическая память (SBSRAM) ориентирована на выполнение пакетного обмена информацией, который характерен для кэш-памяти. Эта память включает в себя внутренний счетчик адреса, предназначенный для перебора адресов пакета, и использует сигналы синхронизации ***CLK,*** как и синхронная DRAM память (см. п. 2.3.2).

Для организации пакетного обмена, помимо имеющихся у асинхронной памяти управляющих сигналов ***CS#***, ***OE#*** и ***WE#***, в синхронную память также введены сигналы ***ADSP#*** (*Address Status of Processor*) и ***CADS#*** (*Cache Address Strobe*), сопровождающие передачу адреса нового пакета, а также сигнал ***ADV#*** (*Advance*) продвижения на следующий адрес пакета. Пакетный цикл всегда предусматривает передачу четырех элементов, так как внутренний счетчик имеет всего 2 бита, причем перебор адресов в пределах пакета может быть последовательным или с расслоением (чередованием) по банкам (при использовании процессоров семейства x86).

Структурная схема БИС динамического ОЗУ

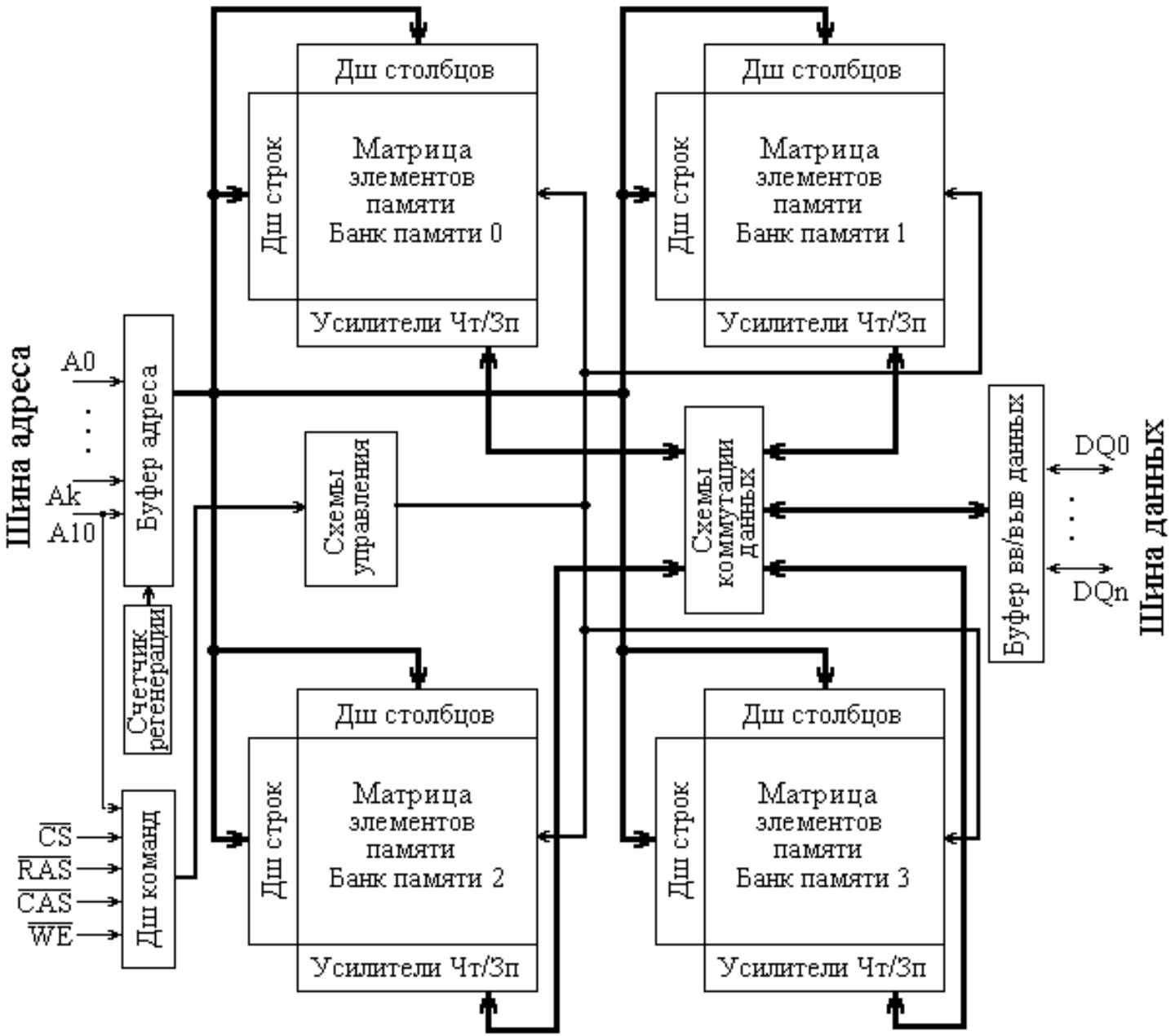
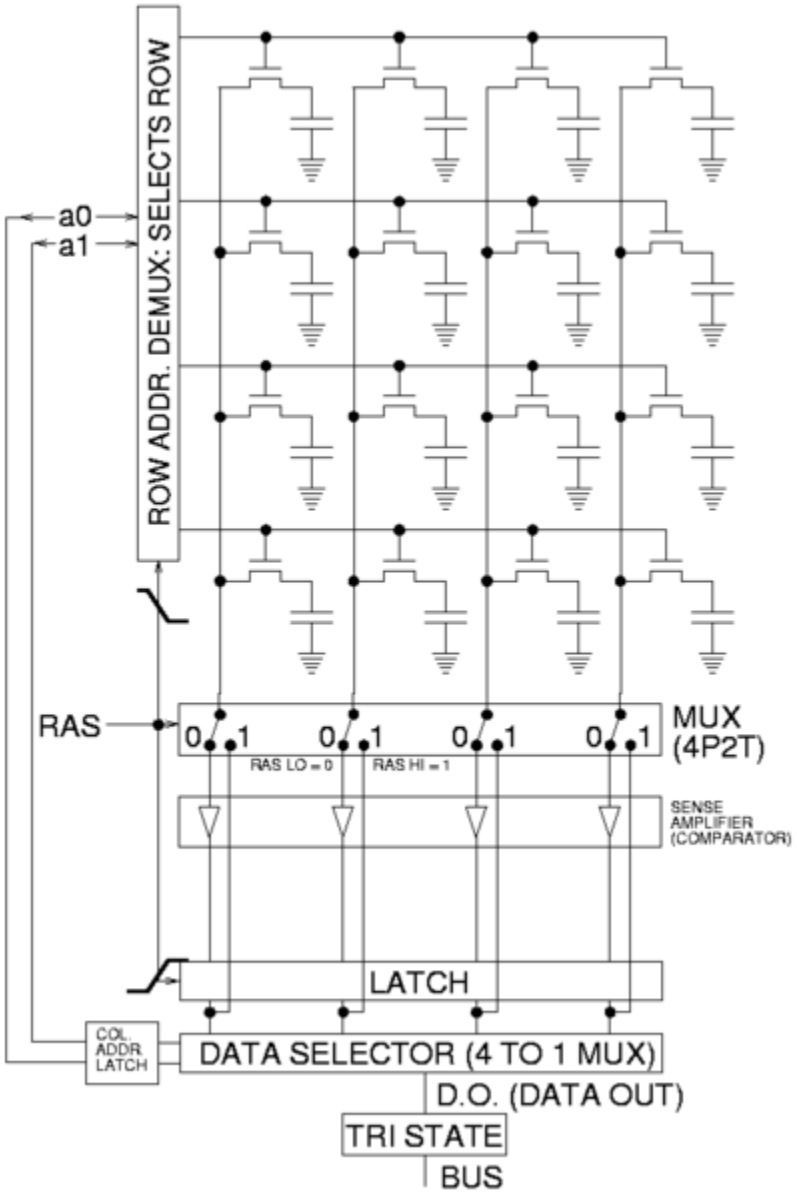
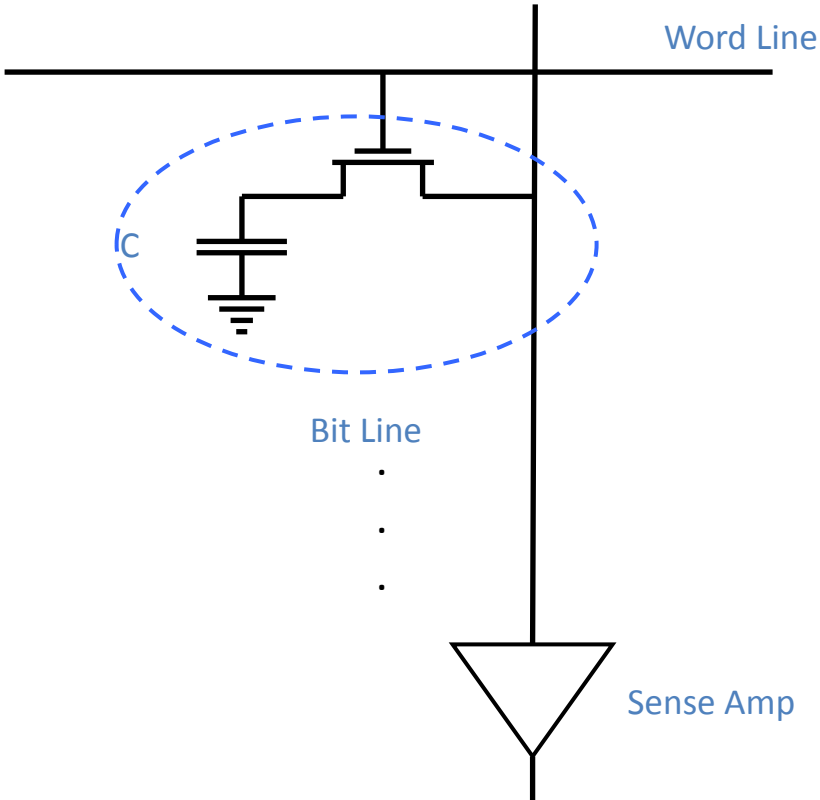


схема БИС динамического ОЗУ

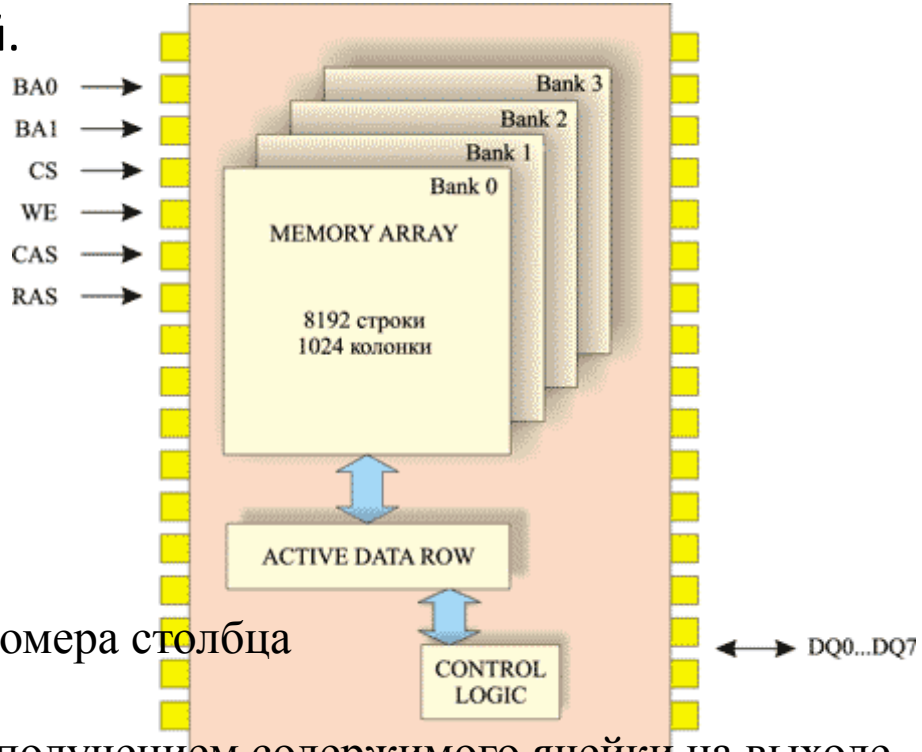


Основными таймингами DRAM являются: задержка между подачей номера строки и номера столбца, называемая временем полного доступа(англ. *RAS to CAS delay*), задержка между подачей номера столбца и получением содержимого ячейки, называемая временем рабочего цикла(англ. *CAS delay*), задержка между чтением последней ячейки и подачей номера новой строки (англ. *RAS precharge*). Тайминги измеряются в наносекундах или тактах, и чем меньше величина этих таймингов, тем быстрее работает оперативная память.



DRAM (Dynamic Random Access Memory) — один из видов компьютерной памяти с произвольным доступом

Конструктивно память DRAM состоит из «ячеек» размером в 1 или 4 бит, в каждой из которых можно хранить определённый объём данных. Совокупность «ячеек» такой памяти образуют условный «прямоугольник», состоящий из определённого количества *строк и столбцов*. Один такой «прямоугольник» называется *страницей*, а совокупность страниц называется *банком*. Весь набор «ячеек» условно делится на несколько областей.



Задержка между подачей номера строки и номера столбца "*RAS to CAS delay*" (**tRCD**).

Задержка между подачей номера столбца и получением содержимого ячейки на выходе - "*CAS delay*" (или **tCAC**),

задержка между чтением последней ячейки и подачей номера новой строки "*RAS precharge*" (**tRP**).

Временные диаграммы асинхронной статической памяти

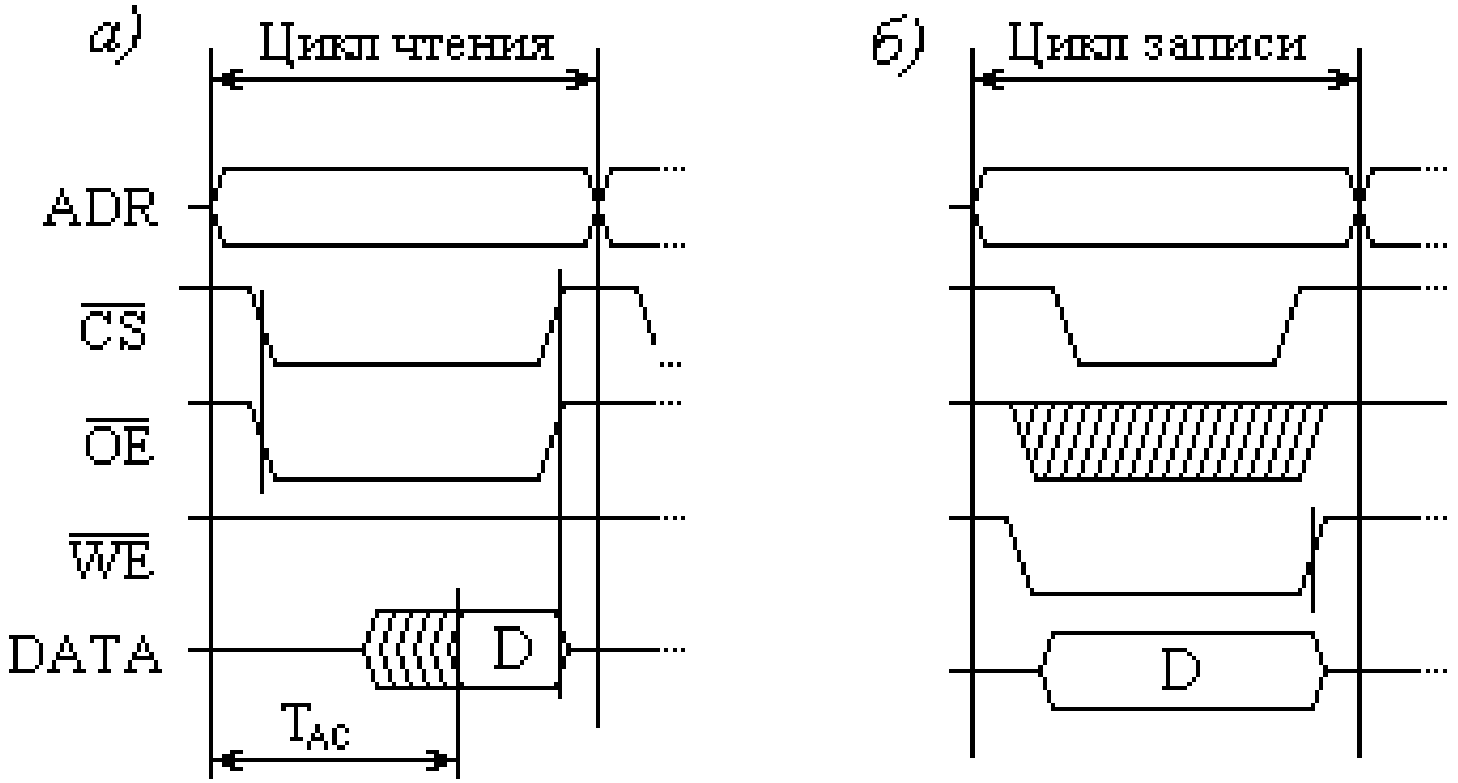
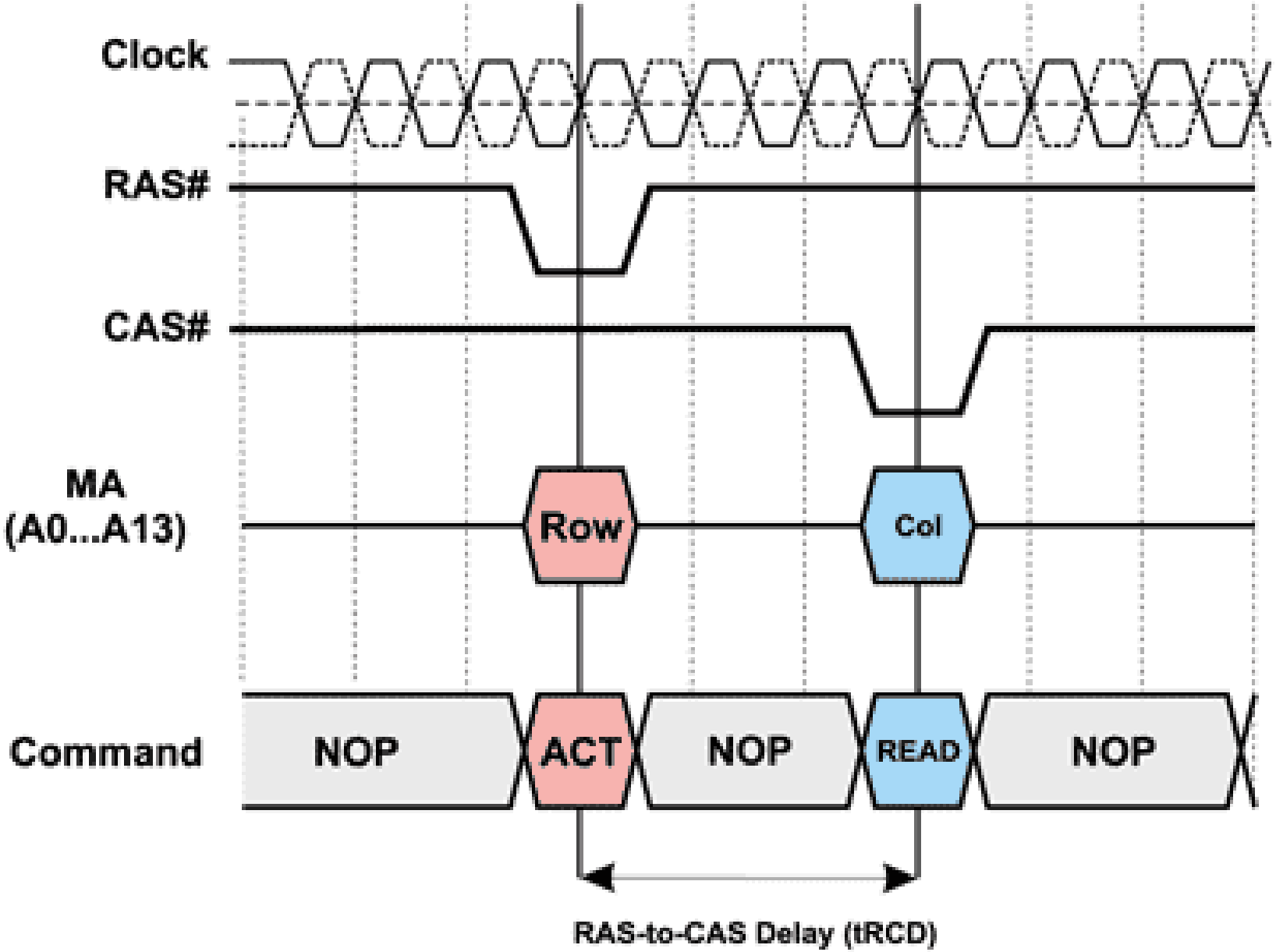
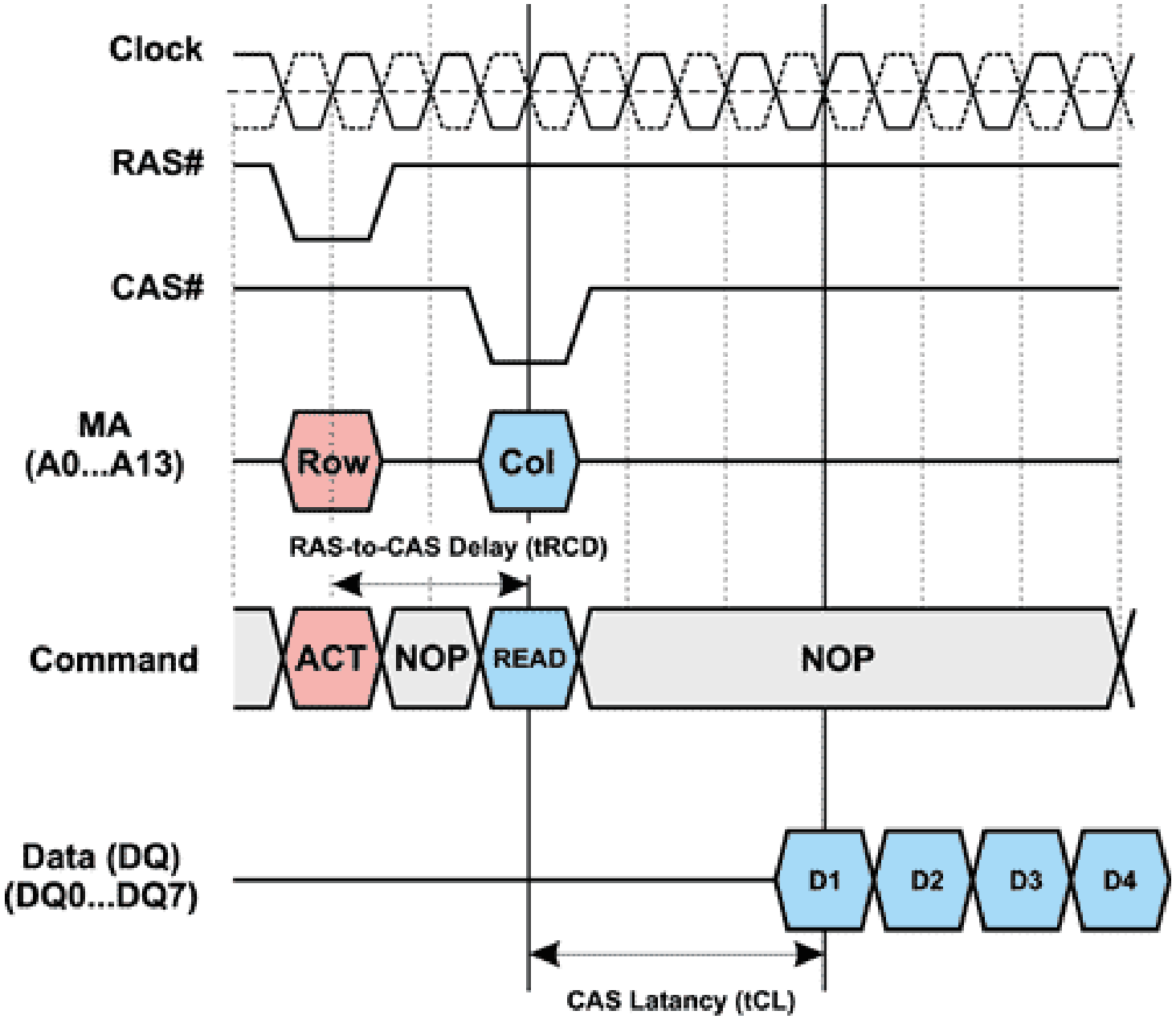


Рис.11. Временная диаграмма простых циклов чтения *а)* и записи *б)* асинхронной статической памяти

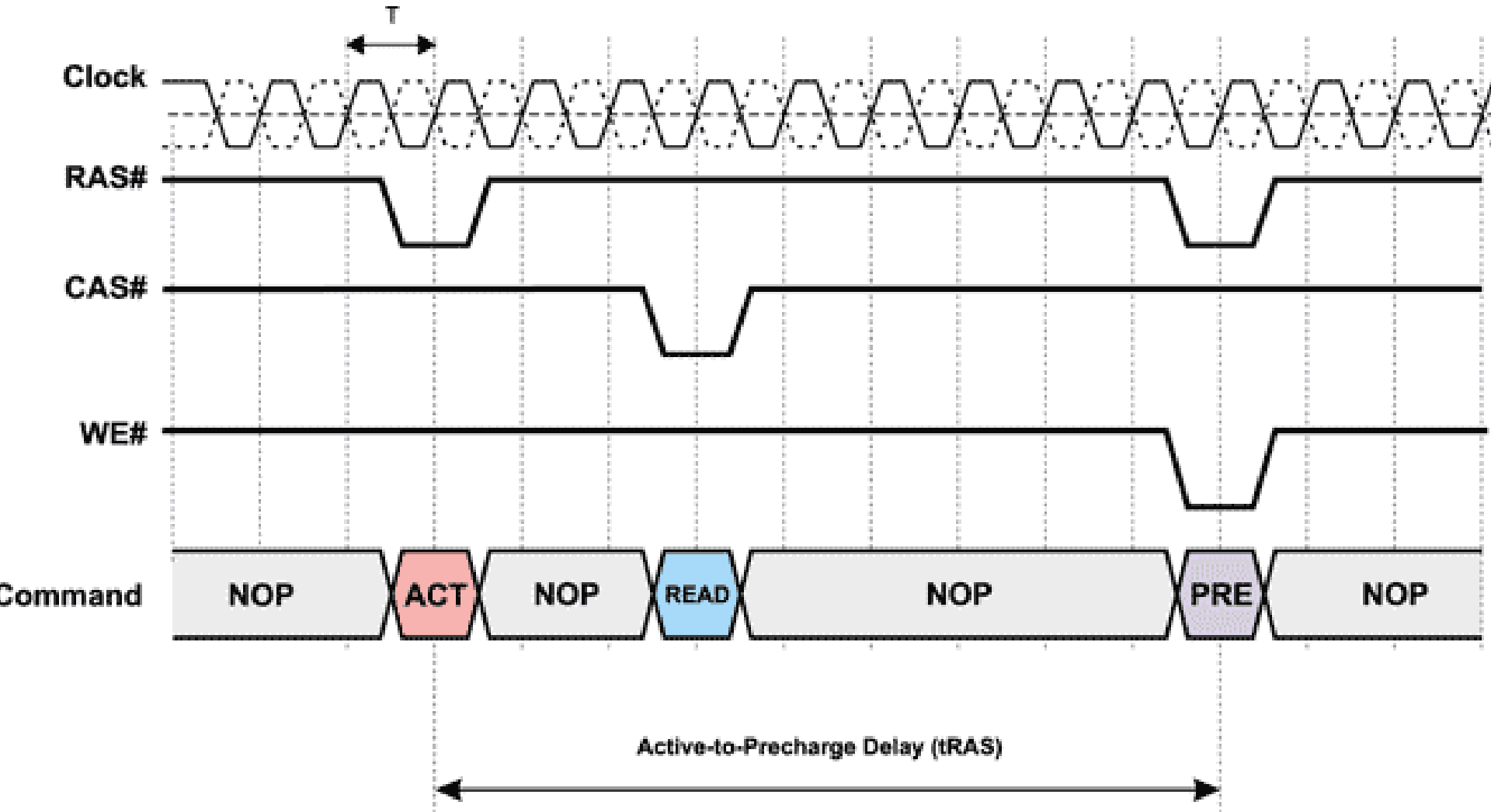
Задержка RAS-to-CAS Delay (tRCD)



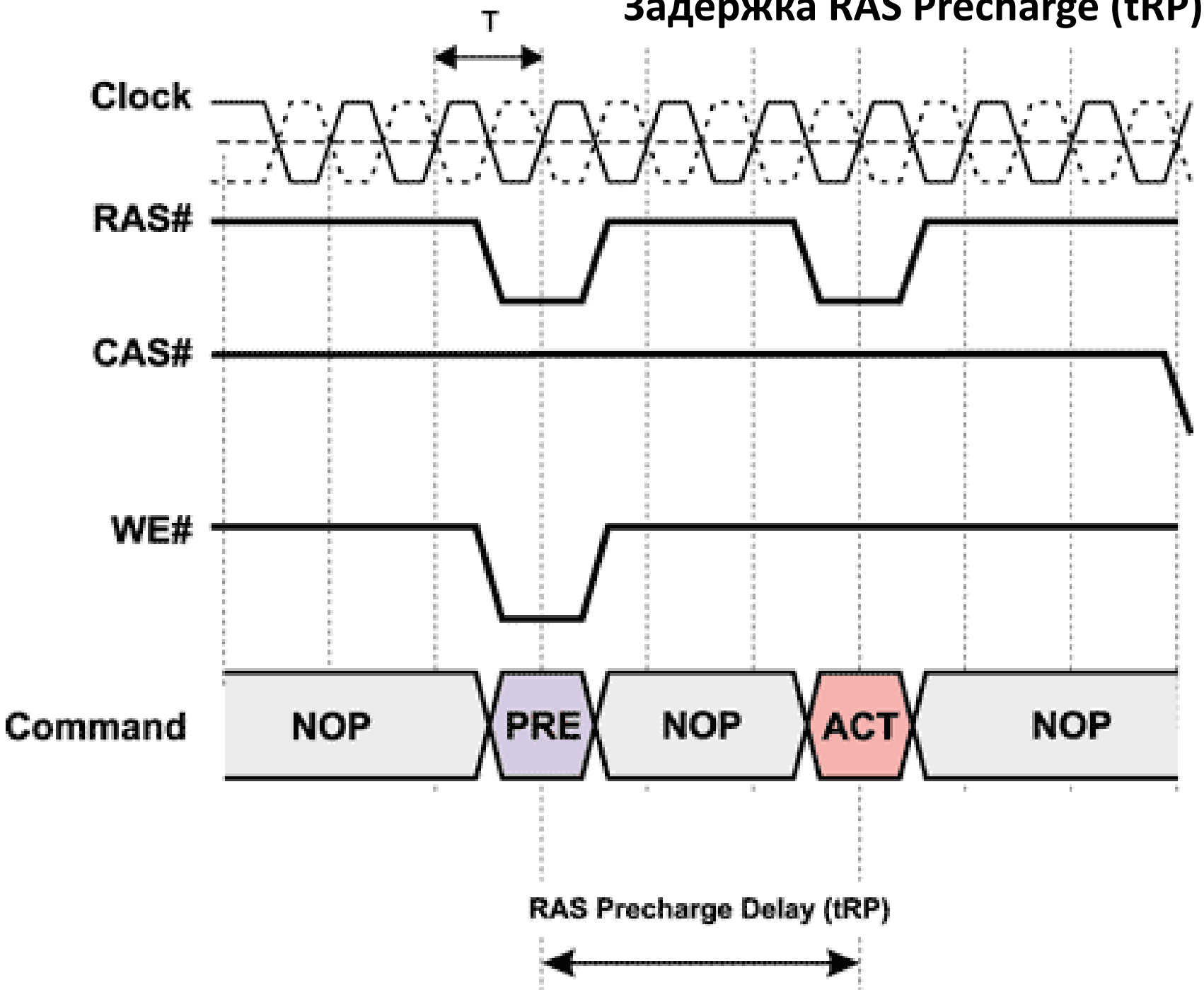
Задержка CAS Latency (tCL)



Задержка Active-to-precharge (tRAS)



Задержка RAS Precharge (tRP)



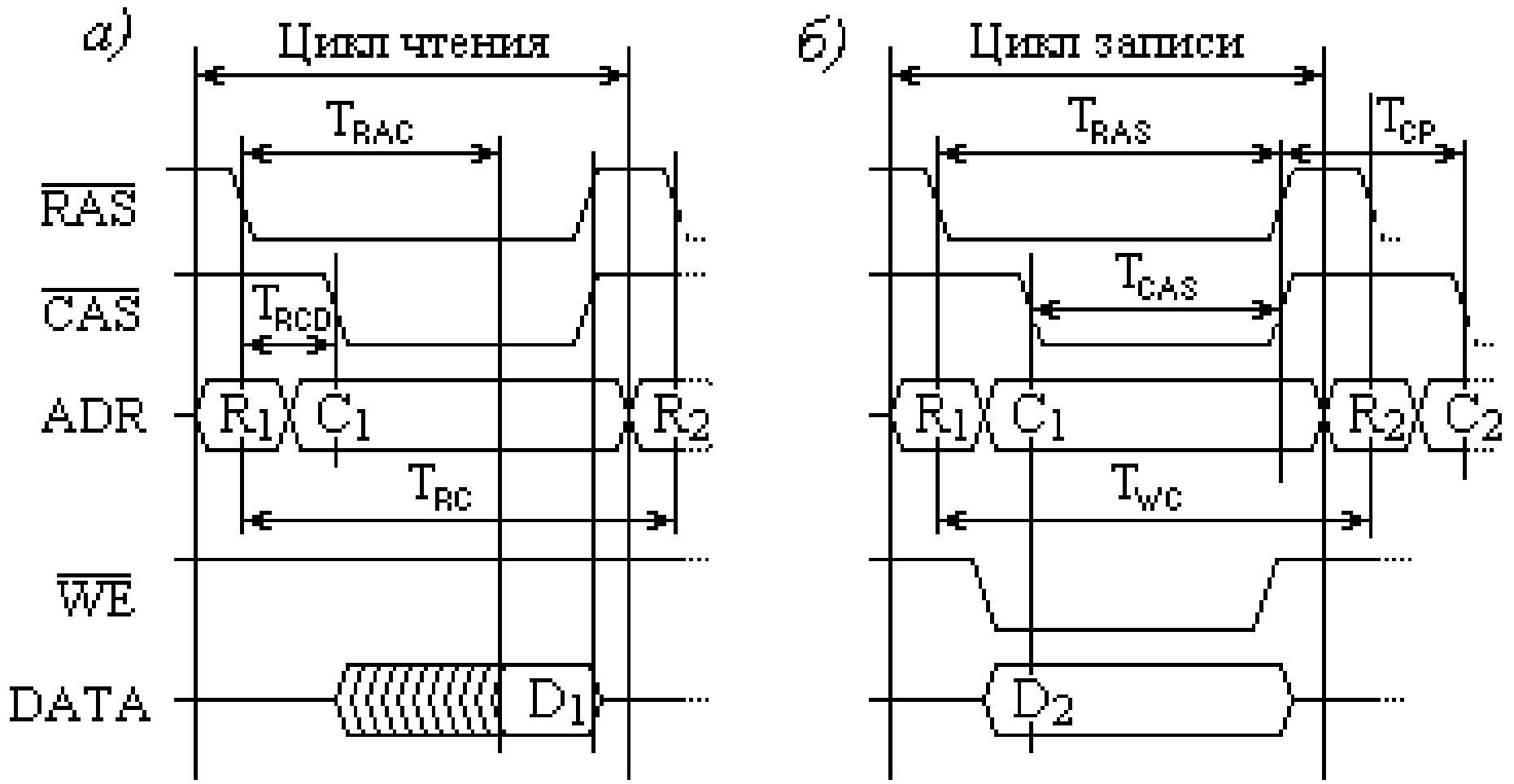


Рис. 15. Временные диаграммы простых циклов чтения *а)* и записи *б)* (асинхронной) динамической памяти

- T_{RCD} – минимальное время задержки между подачей сигналов ***RAS#*** и ***CAS#*** (*RAS-to-CAS Delay*);
- T_{RAS} и T_{CAS} – длительности (активного уровня) сигналов ***RAS#*** и ***CAS#***;
- T_{RC} и T_{WC} – длительности циклов чтения и записи соответственно;
- T_{RP} и $_{TCP}$ – времена подзаряда строки и столбца соответственно (время подзаряда определяет минимальную задержку, необходимую перед подачей очередного сигнала ***RAS#*** или ***CAS#*** после снятия (подъема в “1”) текущего).

  Значения времен T_{RC} и T_{WC} для памяти (90-х годов) составляли порядка 50 – 100 нс, так что на одно (полное) обращение уходило от 5 до 7 циклов системной шины в зависимости от ее частоты, особенностей используемого чипсета и, собственно, быстродействия памяти. Так, для системной шины с частотой 66 Мгц длительность цикла составляет порядка 15 нс, что для 5 – 7 циклов дает диапазон 75 – 100 нс, если же частота системной шины составляла 100 Мгц, то 5 циклов занимают 50 нс.

Первой ласточкой стала **FPM-DRAM** - ***Fast-Page Mode*** *DRAM* (Память быстрого страничного режима), разработанная в 1995 году. Основным отличием от памяти предыдущего поколения стала поддержка ***сокращенных адресо***в. Если очередная запрашиваемая ячейка находится в той же самой строке, что и предыдущая, ее адрес однозначно определяется одним лишь номером столбца и передача номера строки уже не требуется. За счет чего это достигается? Обратимся к диаграмме, изображенной на рис.4. Смотрите, в то время как при работе с обычной DRAM (верхняя диаграмма) после считывания данных сигнал RAS дезактивируется, подготавливая микросхему к новому циклу обмена, контроллер FPM-DRAM удерживает RAS в низком состоянии, избавляясь от повторной пересылки номера строки.

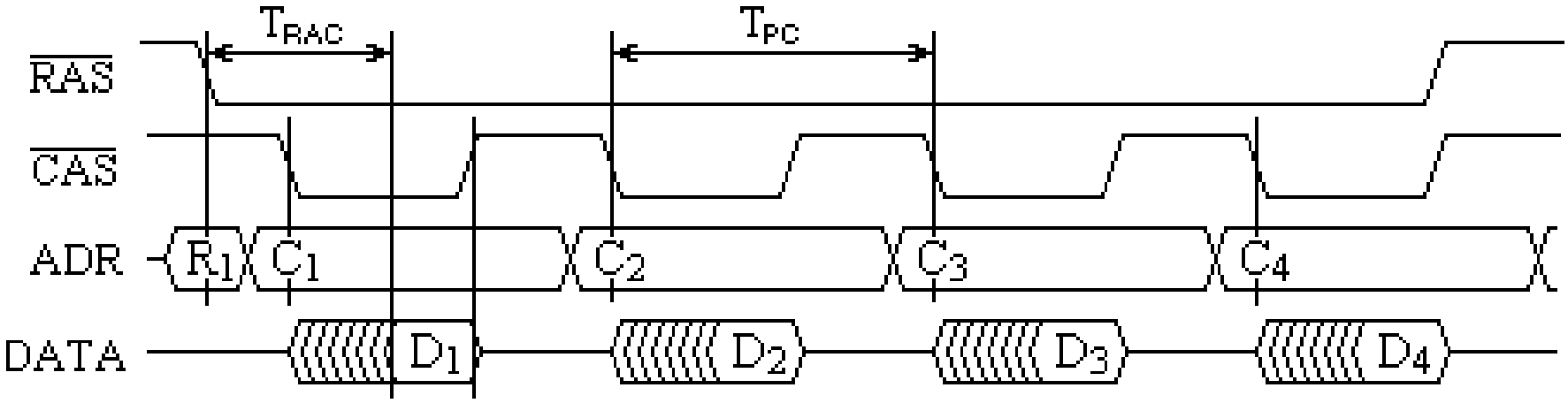


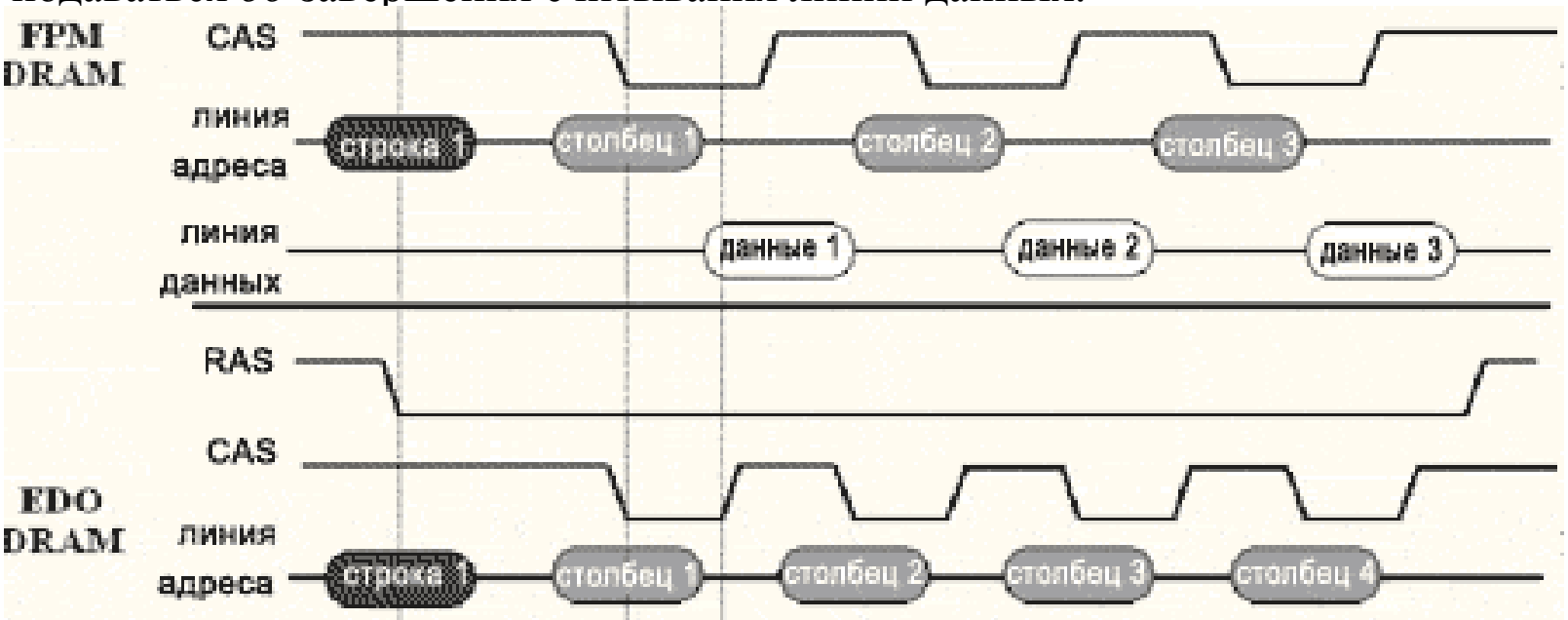
Рис. 16. Временна́я диаграмма цикла чтения последовательных адресов динамической памяти DRAM в режиме FPM

Ситуация, когда запрашиваемая ячейка находится в открытой строке, называется ***"попаданием на страницу"*** *(Page Hit)*, в противном случае говорят, что произошел ***промах*** *(Page Miss)*. Поскольку, промах облагается штрафными задержками, критические к быстродействию модули должны разрабатываться с учетом особенностей архитектуры FPM-DRAM, так что абстрагироваться от ее устройства уже не получается.

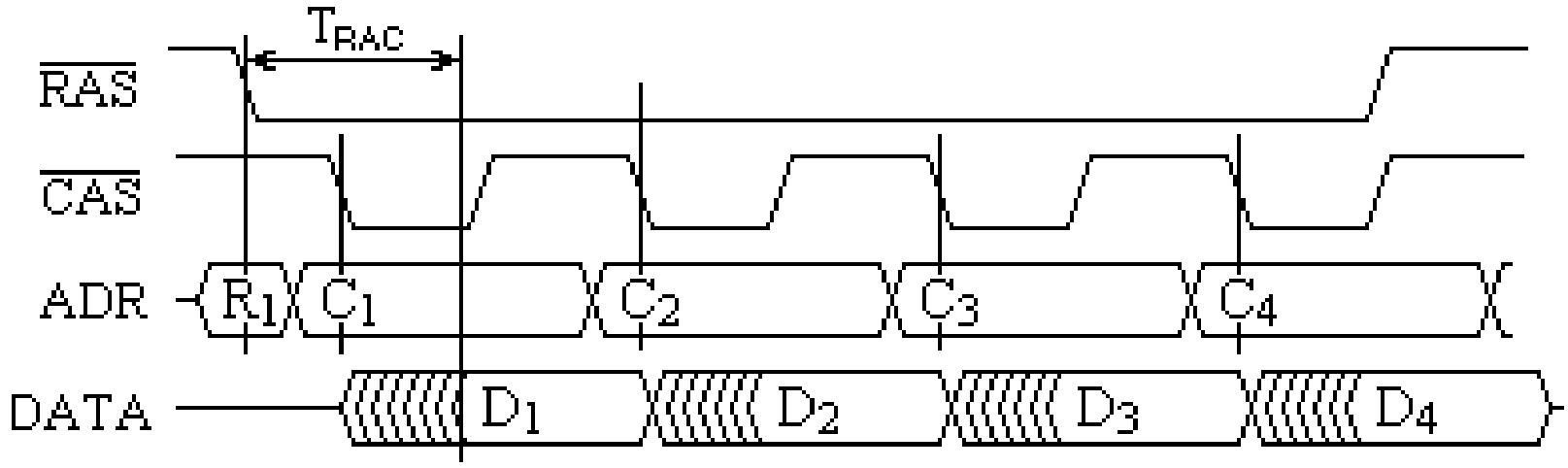
Возникла и другая проблема: непостоянство времени доступа затрудняет измерение производительности микросхем памяти и сравнение их скоростных показателей друг с другом. В худшем случае обращение к ячейке составляет **RAS to CAS Delay + CAS Delay + RAS precharge** нс., а в лучшем: **CAS Delay** нс. Хаотичное, но не слишком интенсивное обращение к памяти (так, чтобы она успевала перезарядиться) требует не более **RAS to CAS Delay + CAS Delay** нс.

К середине девяностых среднее значение RAS to CAS Delay составляло порядка 30 нс., CAS Delay - 40 нс., а RAS precharge - менее 30 нс. (наносекунд). Таким образом, при частоте системной шины в 60 МГц (т.е. ~17 нс.) на открытие и доступ к первой ячейки страницы уходило около 6 тактов, а на доступ к остальным ячейкам открытой страницы - около 3 тактов. Схематически это записывается как **6-3-x-x и называется *формулой памяти***. Формула памяти упрощает сравнение различных микросхем друг с другом, однако для сравнения необходимо знать преобладающий тип обращений к памяти: последовательный или хаотичный. Например, как узнать, что лучше: 5-4-x-x или 6-3-x-x? В данной постановке вопрос вообще лишен смысла. Лучше для *чего*? Для потоковых алгоритмов с последовательной обработкой данных, бесспорно, предпочтительнее последний тип памяти, в противном случае сравнение бессмысленно, т.к. чтение двух несмежных ячеек займет не 5-5-x-x и, соответственно, 6-6-x-x тактов, а 5+RAS precharge-5+RAS precharge-x-x и 6+RAS prechange-6+RAS prechange-x-x. Поскольку время регенерации обоих микросхем не обязательно должно совпадать, вполне может сложиться так, что микросхема 6-3-x-x окажется быстрее и для последовательного, и для хаотичного доступа. Поэтому, практическое значение имеет сравнение лишь вторых цифр - времени рабочего цикла. Совершенствуя ядро памяти, производители сократили его сначала до 35, а затем и до 30 нс., достигнув практически семикратного превосходства над микросхемами прошлого поколения.

Между тем тактовые частоты микропроцессоров не стояли на месте, а стремительно росли, вплотную приближаясь к рубежу в 200 МГц. Рынок требовал качественного нового решения, а не изнуряющей борьбы за каждую наносекунду. Инженеров вновь отправили к чертежным доскам, где (году эдак в 1996) их осенила очередная идея. Если оснастить микросхему специальным триггером-защелкой, удерживающим линии данных после исчезновения сигнала CAS, станет возможным дезактивировать CAS до окончания чтения данных, подготавливая в это время микросхему к приему номера следующего столбца. Взгляните на диаграмму рис. 4: видите, у FPM низкое состояние CAS удерживается до окончания считывания данных, затем CAS дезактивируется, выдерживается небольшая пауза на перезарядку внутренних цепей, и только после этого на адресную шину подается номер колонки следующей ячейки. В новом типе памяти, получившем название **EDO-DRAM** (Extend Data Output), напротив, CAS дезактивируется в процессе чтения данных *параллельно* с перезарядкой внутренних цепей, благодаря чему номер следующего столбца может подаваться *до* завершения считывания линий данных.

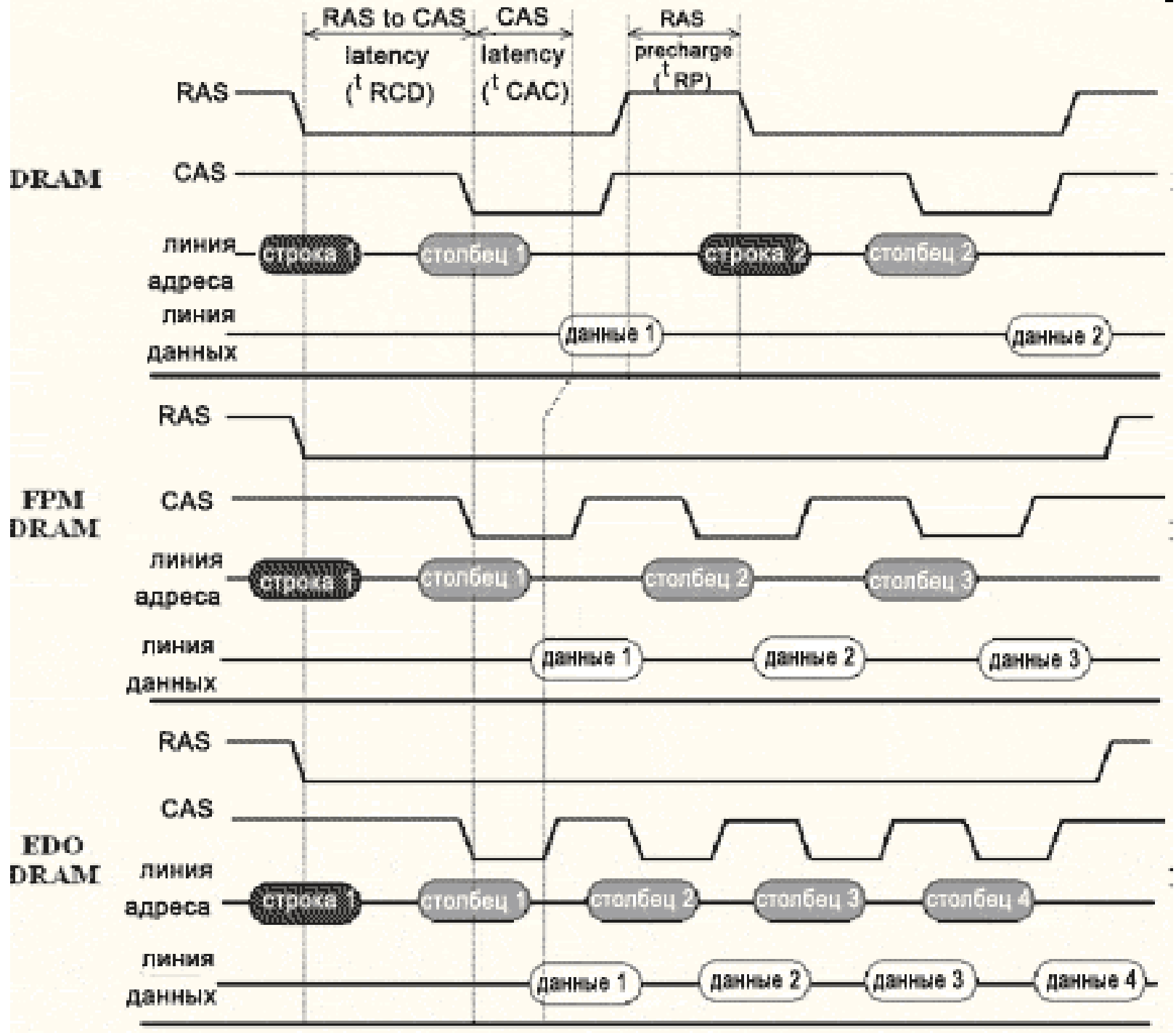


Следующей модификацией асинхронной динамической памяти стала память EDO (*Extended Data Output* – растянутый выход данных). В микросхеме EDO памяти на выходе был установлен буфер-защелка, фиксирующий данные после их извлечения из матрицы памяти при подъеме сигнала ***CAS#*** и удерживающий их на выходе до следующего его спада. Это позволило сократить длительность сигнала ***CAS#*** и соответственно цикла памяти, доведя пакетный цикл до соотношения с циклами системной шины 5-2-2-2 (т.е. сократить длительность второго и последующих циклов в 1,5 раза только за счет выходного регистра-буфера).



Продолжительность рабочего цикла EDO-DRAM (в зависимости от качества микросхемы) составляла 30, 25 и 20 нс., что соответствовало всего двум тактам в 66 МГц системе. Совершенствование производственных технологий сократило и полное время доступа. На частоте 66 МГц формула лучших EDO-микросхем выглядела так: **5-2-x-x**. Простой расчет позволяет установить, что пиковый прирост производительности (в сравнении с FPM-DRAM) составляет около 30%, однако, во многих компьютерных журналах тех лет фигурировала совершенно немыслимая цифра 50%, - якобы настолько увеличивалась скорость компьютера при переходе с FPM на EDO. Это могло быть лишь при сравнении худшей FMP-DRAM с самой "крутой" EDO-памятью, т.е. фактически сравнивались не технологии, а старые и новые микросхемы.

Двукратное увеличение производительности было достигнуто лишь в BEDO-DRAM (Burst EDO). Добавив в микросхему генератор номера столбца, конструкторы ликвидировали задержку CAS Delay, сократив время цикла до 15 нс. После обращения к произвольной ячейке микросхема BEDO *автоматически*, без указаний со стороны контроллера, увеличивает номер столбца на единицу, не требуя его явной передачи. По причине ограниченной разрядности адресного счетчика (конструкторы отвели под него всего лишь два бита) максимальная длина пакета не могла превышать четырех ячеек (22=4).
Забегая вперед, отметим, что процессоры Intel 80486 и Pentium в силу пакетного режима обмена с памятью *никогда* не обрабатывают менее четырех смежных ячеек за раз. Поэтому, независимо от порядка обращения к данным, BEDO всегда работает на максимально возможной скорости и для частоты 66 Мгц ее формула выглядит так: **5-1-1-1**, что на ~40% быстрее EDO-DRAM! Все же, несмотря на свои скоростные показатели, BEDO оказалась не конкурентоспособной и не получила практически никакого распространения. Просчет состоял в том, что BEDO, как и все ее предшественники, оставалась ***асинхронной*** памятью. Это накладывало жесткие ограничения на максимально достижимую тактовую частоту, ограниченную 60 - 66 (75) мегагерцами.
Действительно, пусть время рабочего цикла составляет 15 нс. (1 такт в 66 MHz системе). Однако, поскольку "часы" контроллера памяти и самой микросхемы памяти не синхронизованы, нет никаких гарантий, что начало рабочего цикла микросхемы памяти совпадет с началом такового импульса контроллера, вследствие чего минимальное время ожидания составляет ***два*** такта. Вернее, если быть совсем точным, *рабочий цикл микросхемы памяти никогда не совпадает с началом тактового импульса*.


RAS to CAS latency (tRCD)
CAS latency (tCAC)
RAS precharge (tRP)
DRAM
RAS
CAS
линия адреса
строка 1
столбец 1
строка 2
столбец 2
линия данных
данные 1
данные 2
FPM DRAM
RAS
CAS
линия адреса
строка 1
столбец 1
столбец 2
столбец 3
линия данных
данные 1
данные 2
данные 3
EDO DRAM
RAS
CAS
линия адреса
строка 1
столбец 1
столбец 2
столбец 3
столбец 4
линия данных
данные 1
данные 2
данные 3
данные 4

SDRAM (Synchronous DRAM) – синхронное управление выборкой из памяти.

В отличие от других типов **DRAM**, использовавших асинхронный обмен данными, ответ на поступивший в устройство управляющий сигнал возвращается не сразу, а лишь при получении следующего тактового сигнала. Тактовые сигналы позволяют организовать работу **SDRAM** в виде конечного автомата, исполняющего входящие команды. При этом входящие команды могут поступать в виде непрерывного потока, не дожидаясь, пока будет завершено выполнение предыдущих инструкций (конвейерная обработка): сразу после команды записи может поступить следующая команда, не ожидая, когда данные окажутся записаны. Поступление команды чтения приведёт к тому, что на выходе данные появятся спустя некоторое количество тактов — это время называется *задержкой* (*latency*) и является одной из важных характеристик данного типа устройств.
Циклы обновления выполняются сразу для целой строки, в отличие от предыдущих типов DRAM, обновлявших данные по внутреннему счётчику, используя способ обновления по команде CAS перед RAS.

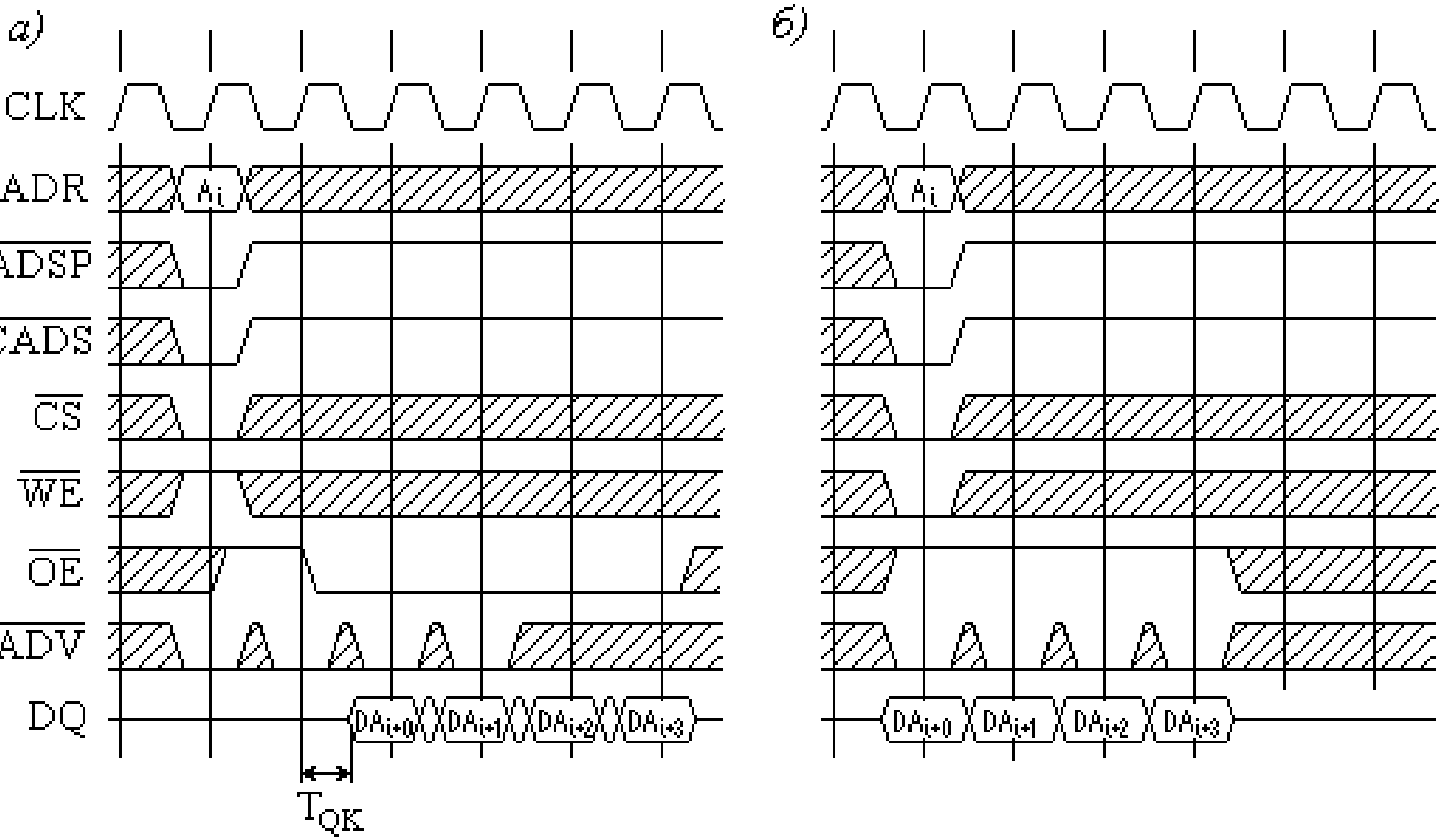
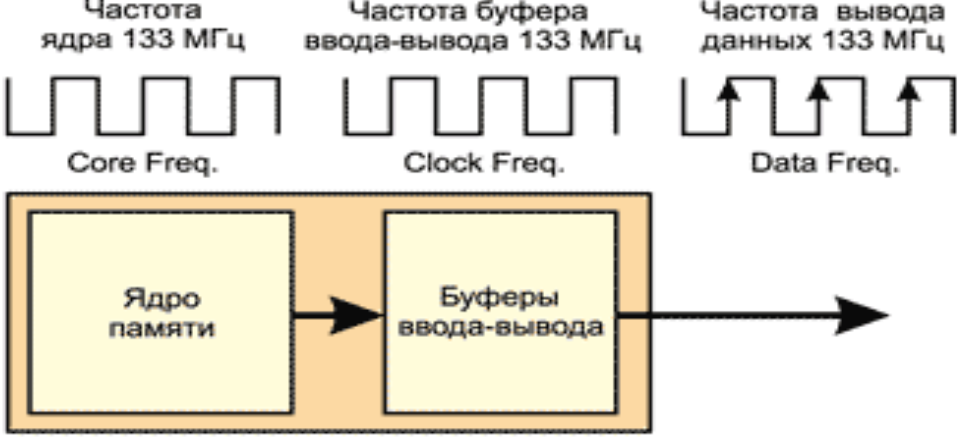


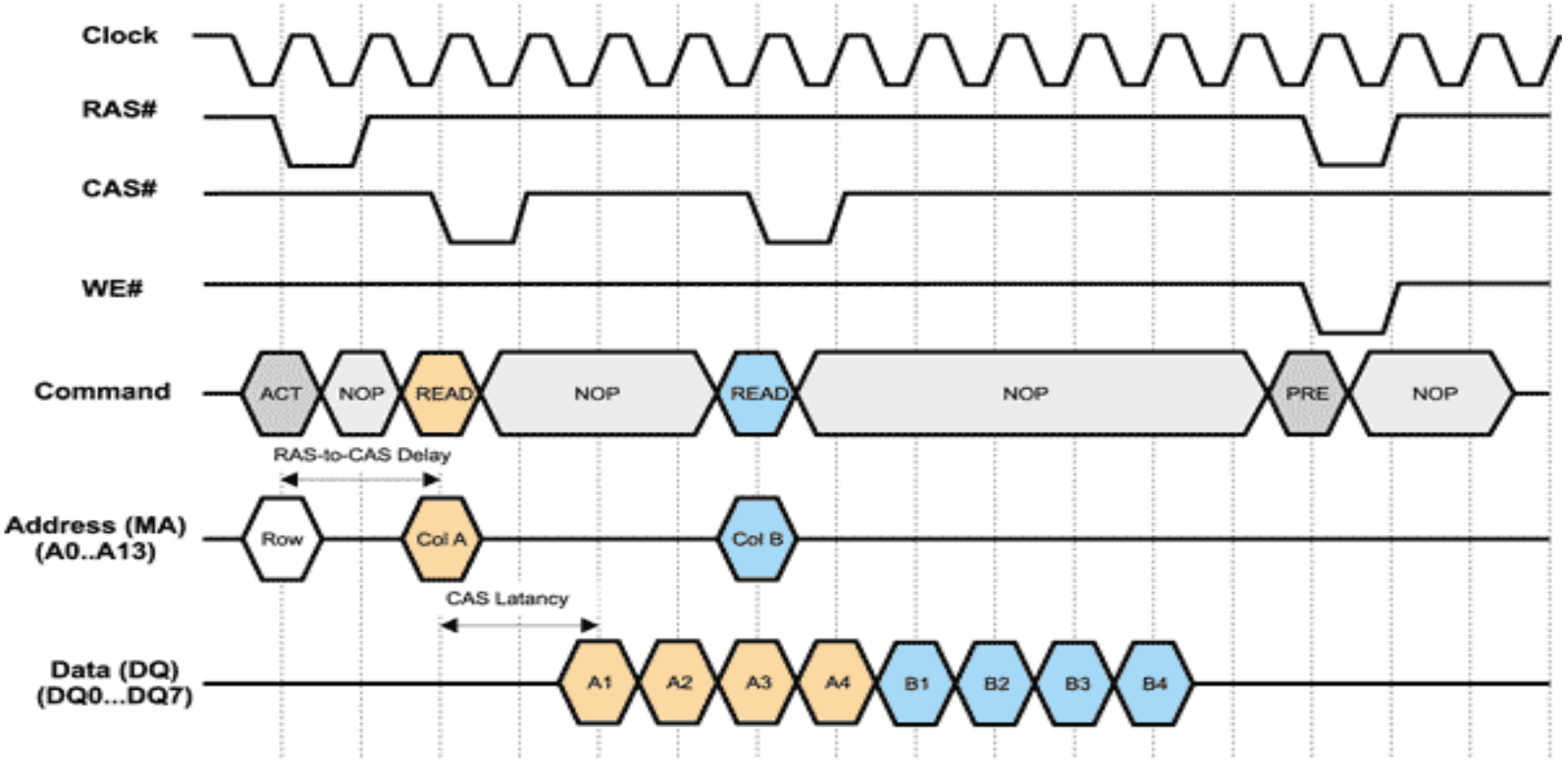
Рис. 12. Пакетные циклы чтения *а)* и записи *б)* синхронной пакетной статической памяти (SBSRAM)

Синхронная динамическая память

1. У памяти SDRAM присутствует синхросигнал ***CLK***, по переднему фронту которого производятся все переключения в микросхеме. Кроме этого сигнала имеется также сигнал ***CKE*** (*Clock Enable*), разрешающий работу микросхемы при высоком уровне, а при низком – переводящий ее в один из режимов энергосбережения.
2. В интерфейсе SDRAM имеются сигналы выбора банка ***BS***0 и ***BS***1 (*Bank Select*), позволяющие адресовать конкретные обращения в один из четырех имеющихся в микросхемах SDRAM банков (массивов элементов) памяти.
3. Присутствуют сигналы ***DQM*** маски линий данных, позволяющие блокировать запись данных в цикле записи или переключать шину данных в состояние высокого выходного сопротивления (z-состояние) при чтении.
4. Имеет место специфическое использование одной из адресных линий (A_{10}) в момент подачи сигнала ***CAS***#. Значение сигнала на этой линии задает способ подзаряда строки банка.



SDR SDRAM



tRCD = 2; tCL = 2; Длина пакета BL = 4

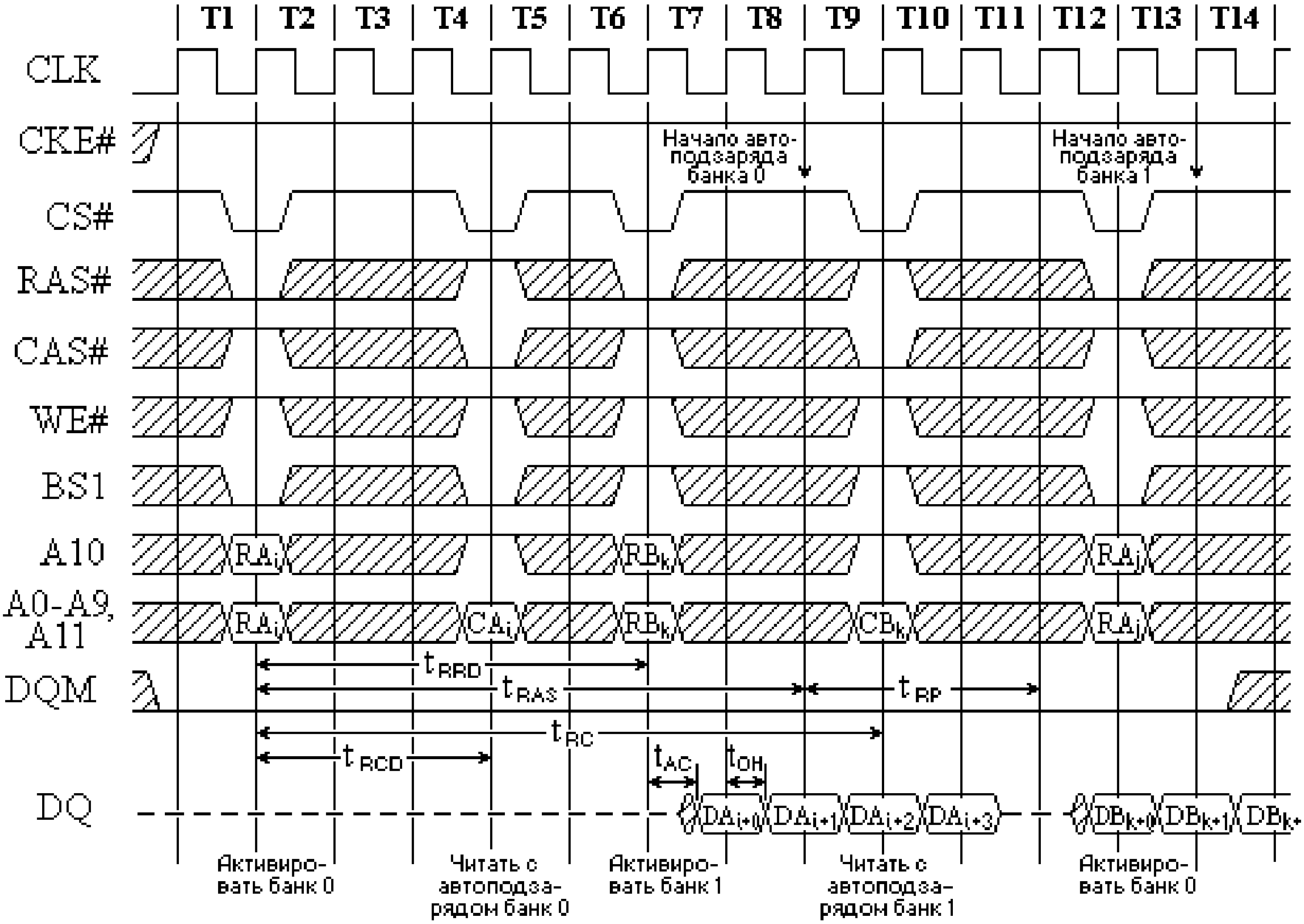


Рис.18. Временная диаграмма пакетного чтения из SDRAM (длина пакета = 4, задержка появления данных CAS Latency = 3)

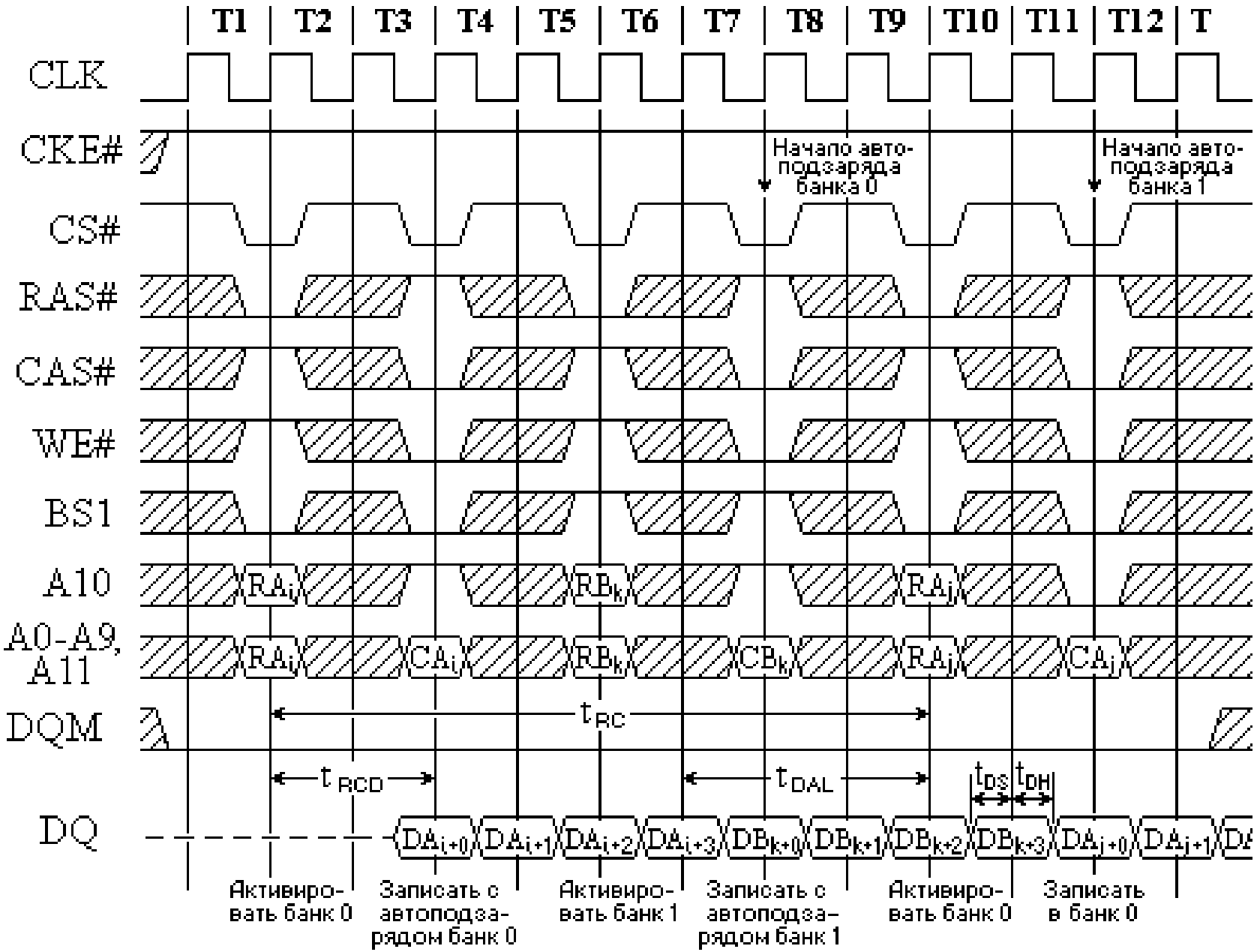
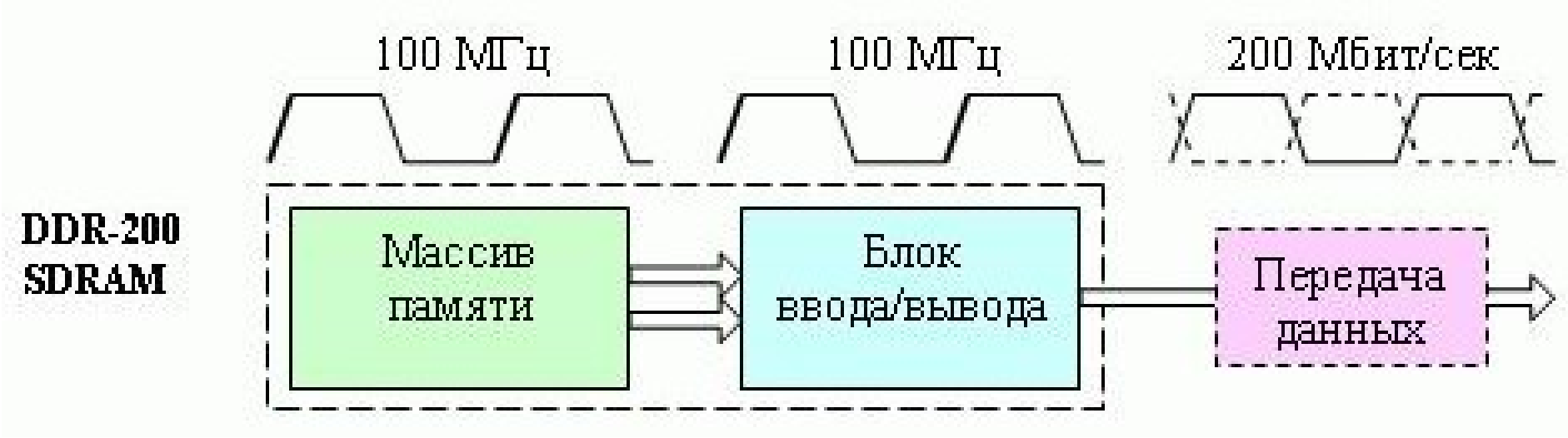
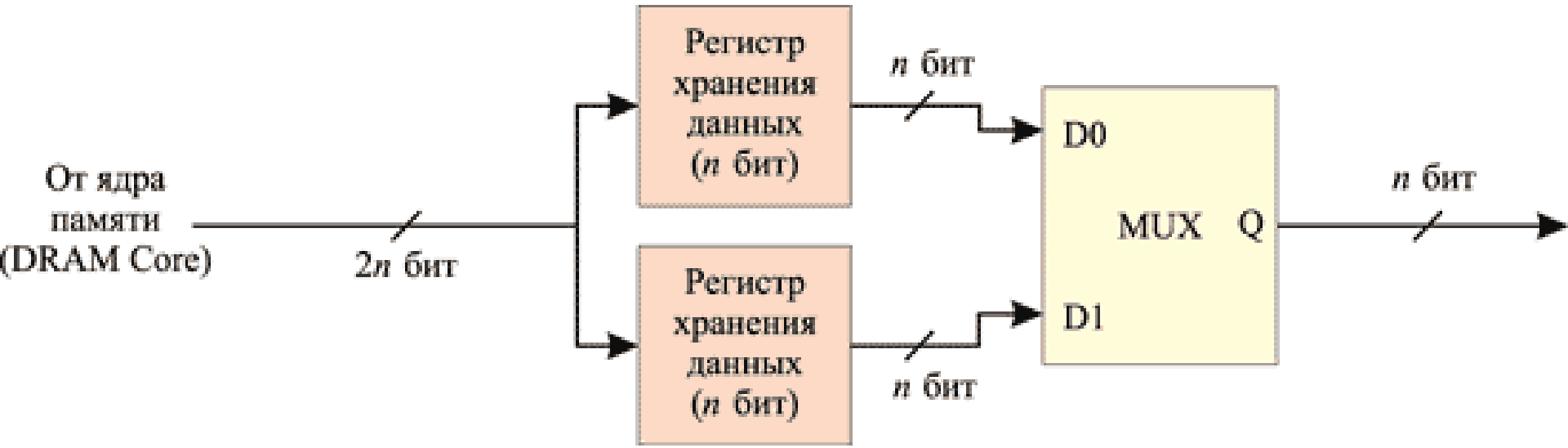


Рис.19. Временная диаграмма пакетной записи в SDRAM (длина пакета = 4, задержка появления данных CAS Latency = 2)

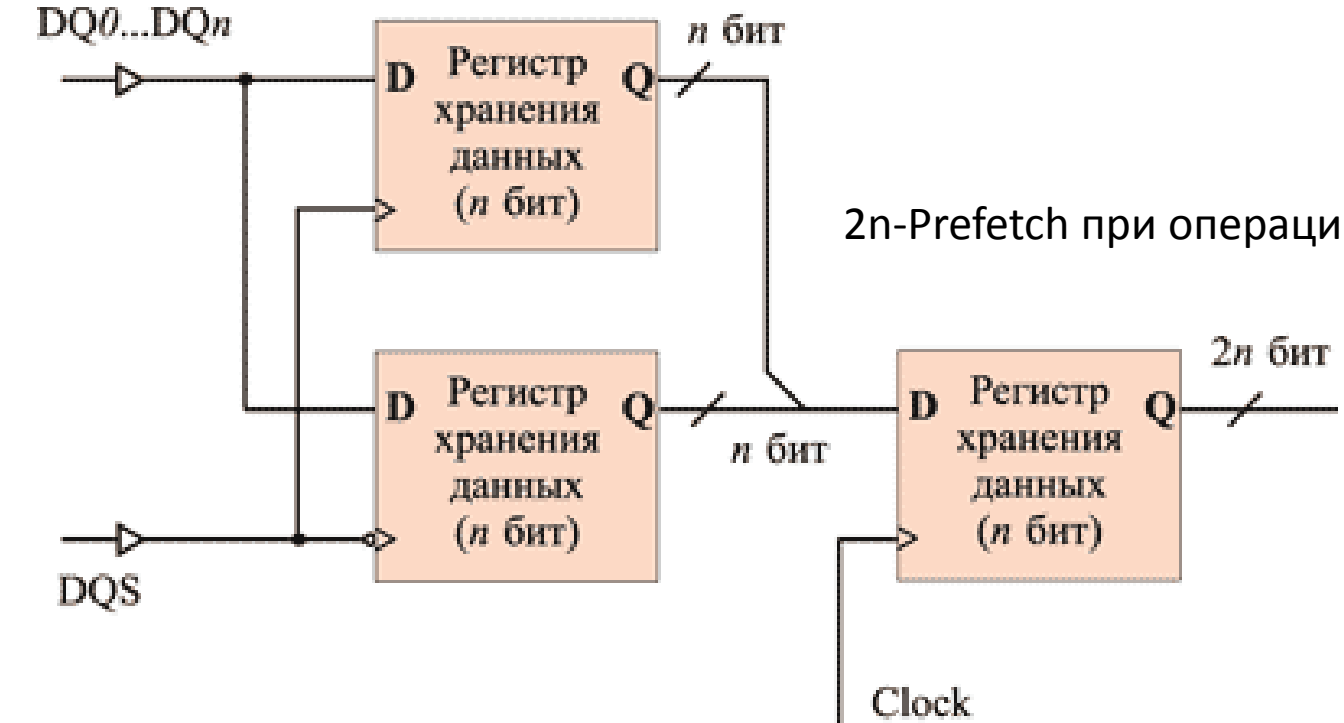
DDR SDRAM (от англ. ***Double Data Rate Synchronous Dynamic Random Access Memory*** — синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных). Удвоенная скорость работы достигается за счёт считывания команд и данных не только по фронту, как в SDRAM, но и по срезу тактового сигнала. За счёт этого удваивается скорость передачи данных, не увеличивая при этом частоты тактового сигнала шины памяти. Таким образом, при работе DDR на частоте 100 МГц мы получим эффективную частоту 200МГц (при сравнении с аналогом SDR SDRAM).

Чтобы обеспечить передачу данных дважды за такт, используется специальная архитектура «2n Prefetch». Внутренняя шина данных имеет ширину в два раза больше внешней. При передаче данных сначала передаётся первая половина шины данных по переднему фронту тактового сигнала, а затем вторая половина шины данных по срезу.

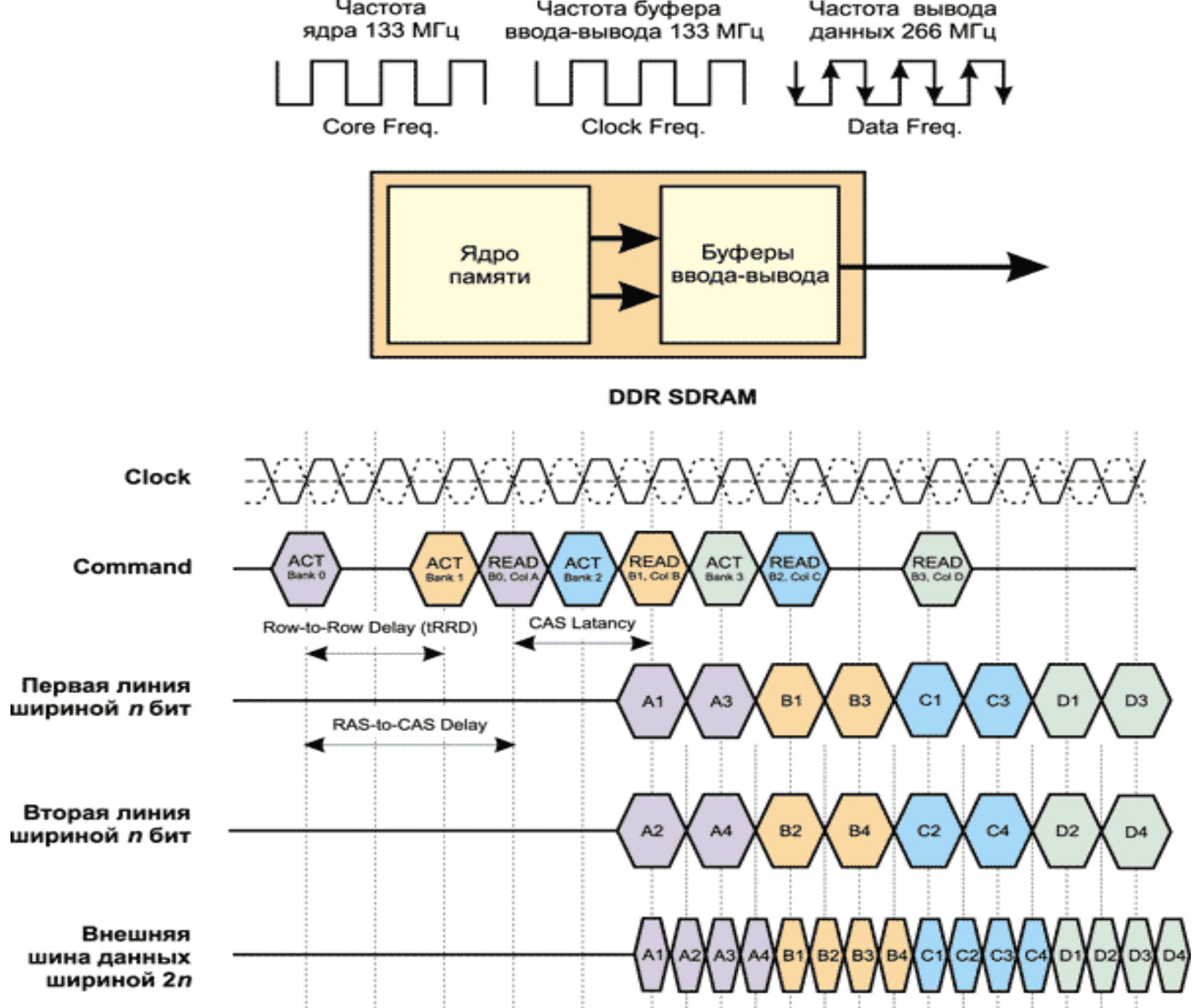




2n-Prefetch при операции чтения данных

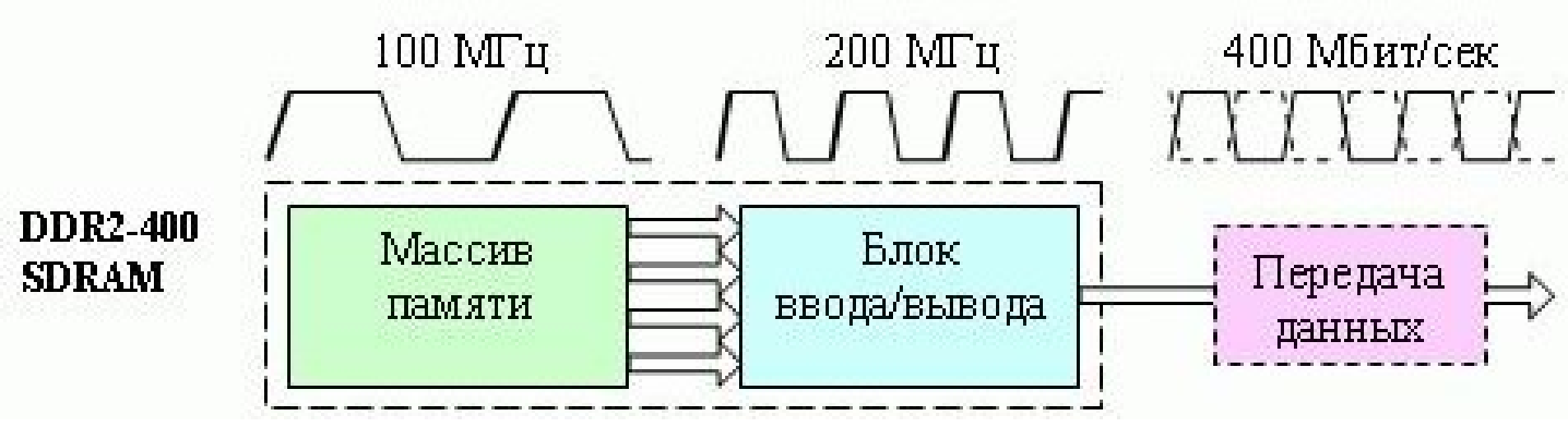


2n-Prefetch при операции записи данных

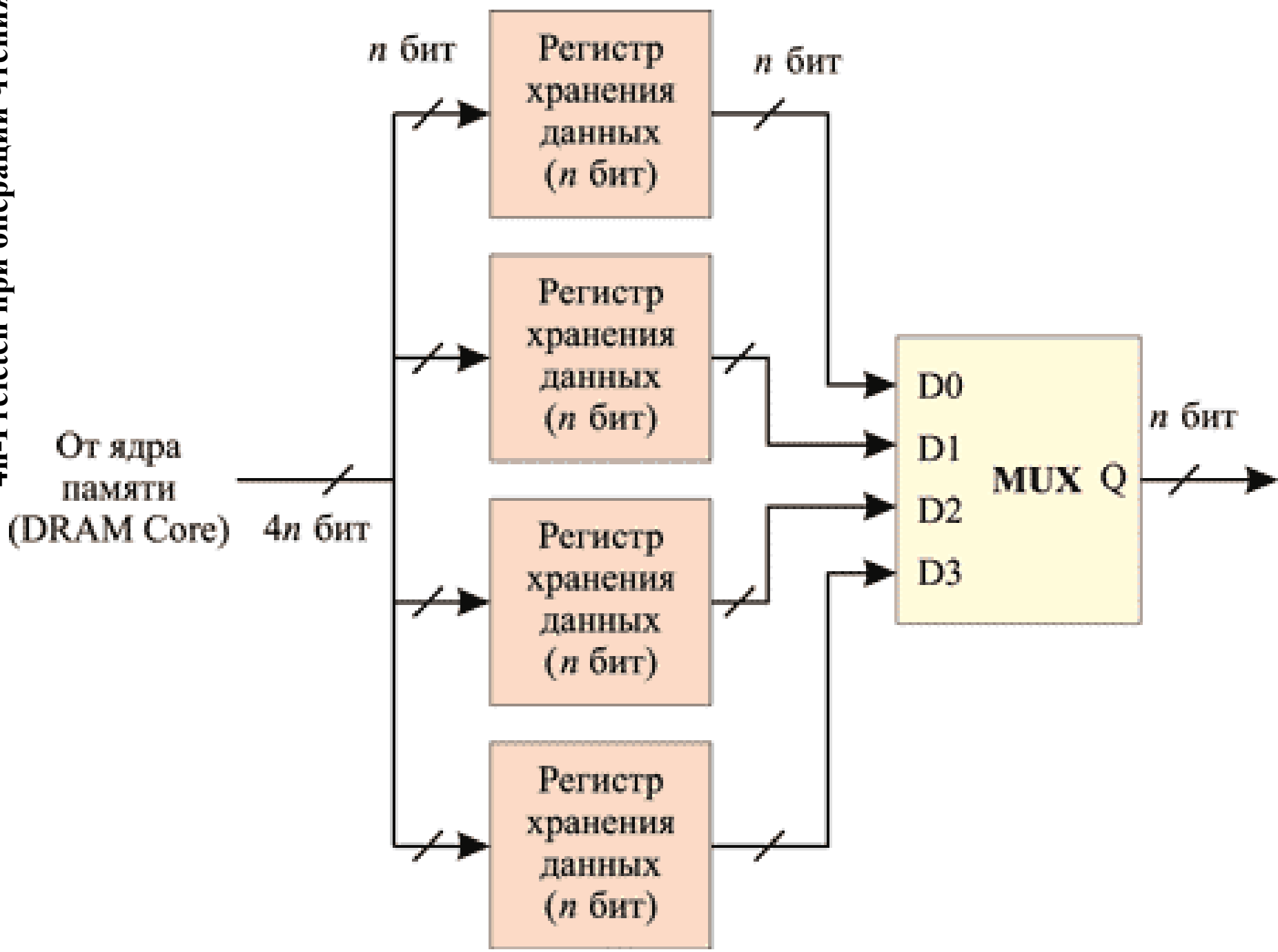


Упрощенная временная диаграмма работы DDR SDRAM-памяти

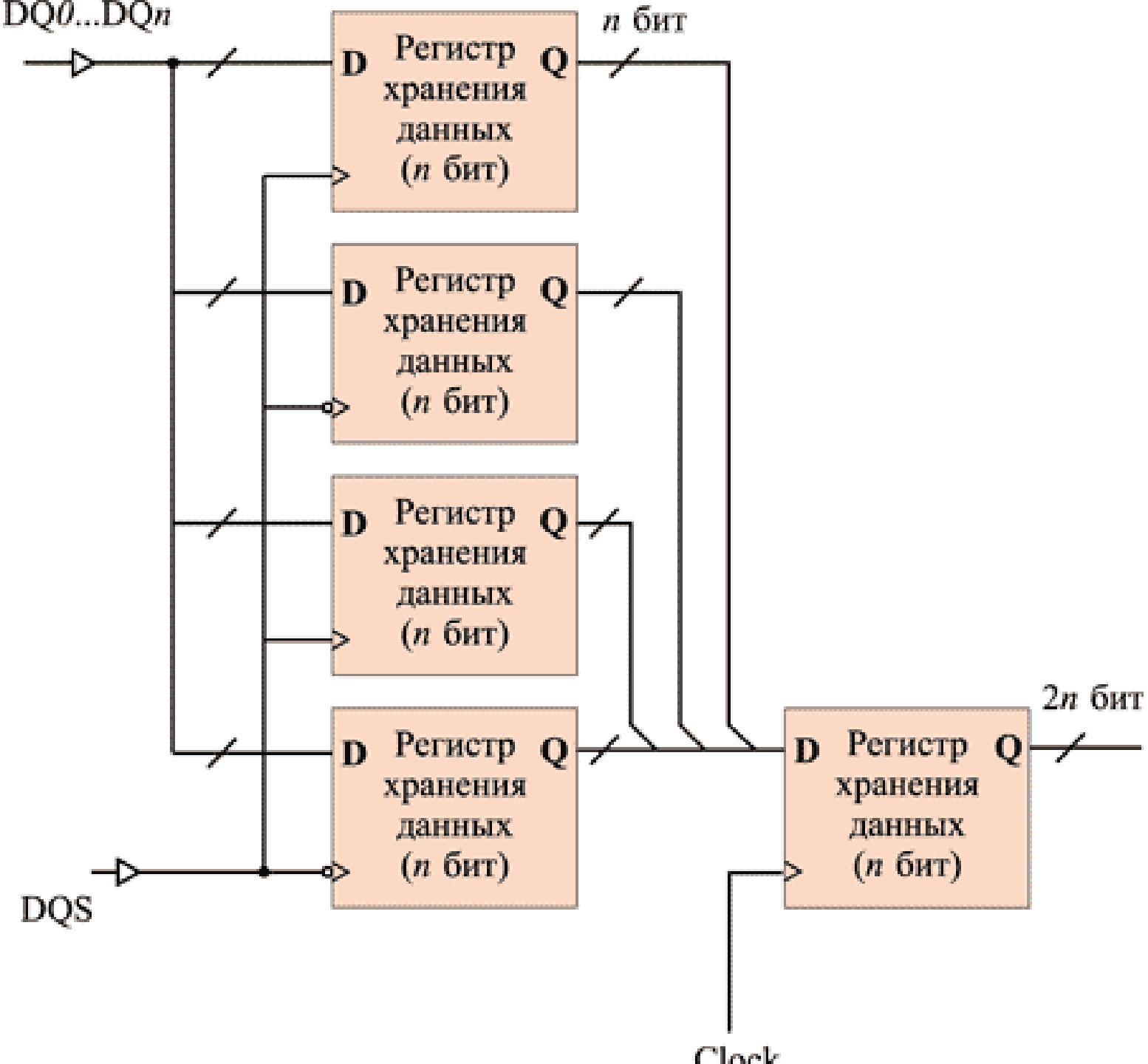
Как и DDR SDRAM, **DDR2 SDRAM** использует передачу данных по обоим срезам тактового сигнала, за счёт чего при такой же частоте шины памяти, как и в обычной SDRAM, можно фактически удвоить скорость передачи данных (например, при работе DDR2 на частоте 100 МГц эквивалентная эффективная частота для SDRAM получается 200 МГц). Основное отличие DDR2 от DDR - вдвое большая частота работы шины, по которой данные передаются в буфер микросхемы памяти. При этом, чтобы обеспечить необходимый поток данных, передача на шину осуществляется из четырёх мест одновременно. Итоговые задержки оказываются выше, чем для DDR.

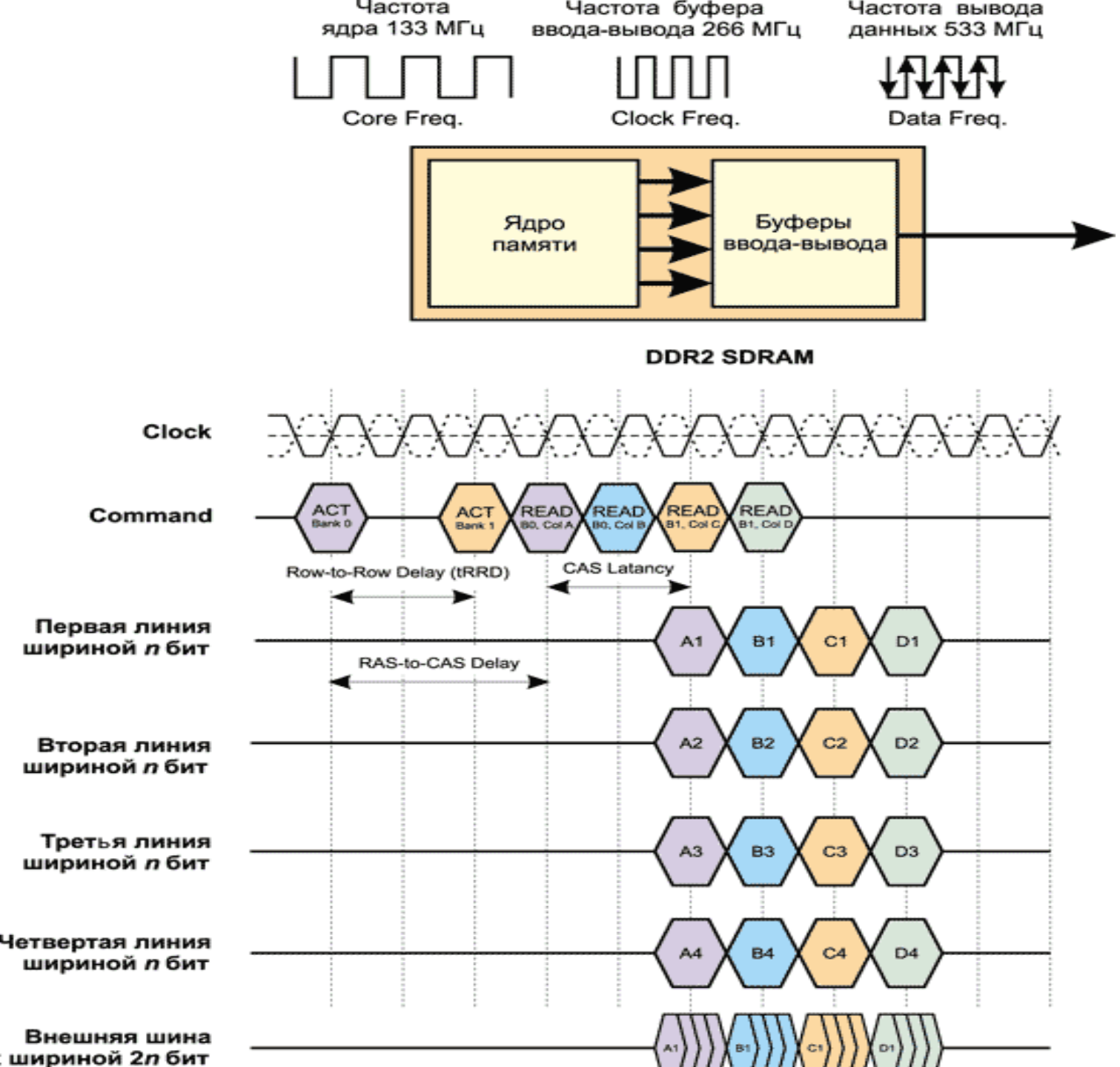


4n-Prefetch при операции чтения



4n-Prefetch при операции записи

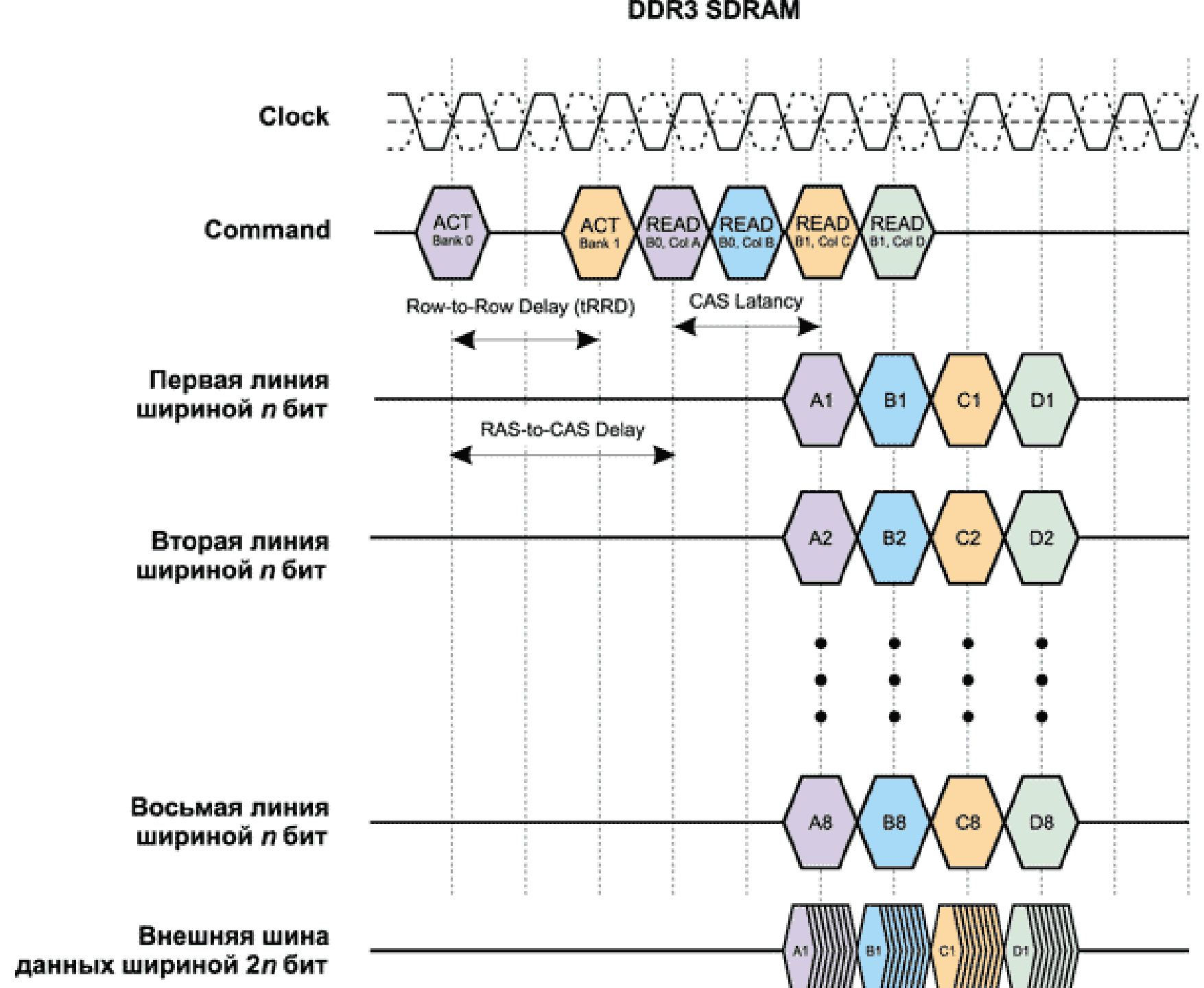


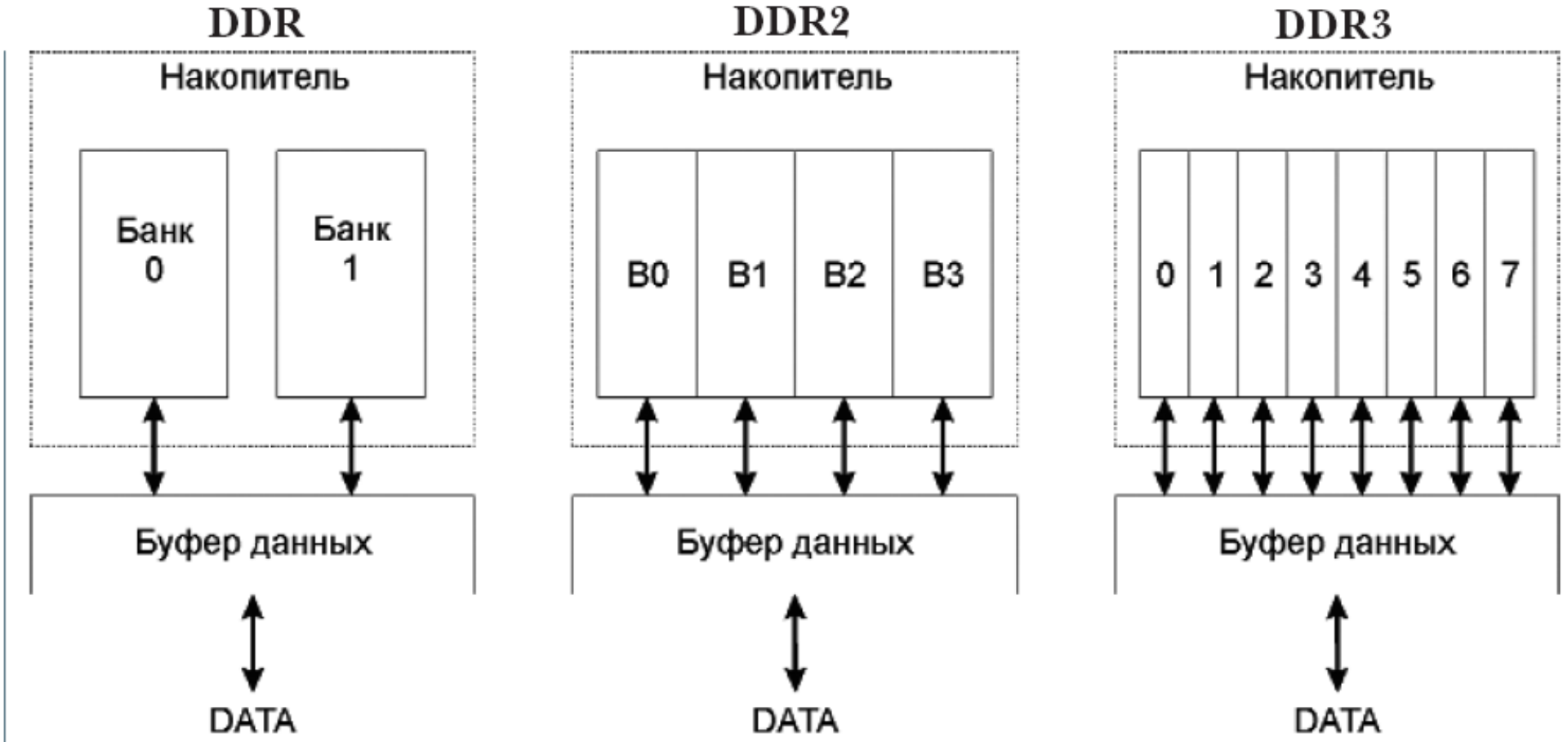


Упрощенная временная диаграмма работы SDRAM-памяти DDR2

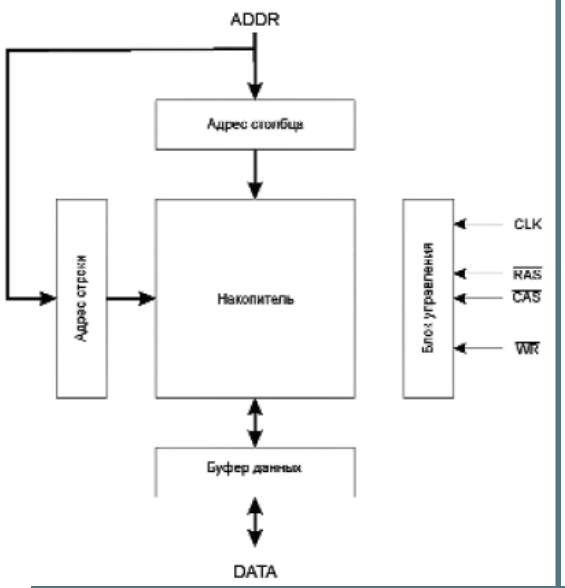
Как и DDR SDRAM, **DDR2 SDRAM** использует передачу данных по обоим срезам тактового сигнала, за счёт чего при такой же частоте шины памяти, как и в обычной SDRAM, можно фактически удвоить скорость передачи данных (например, при работе DDR2 на частоте 100 МГц эквивалентная эффективная частота для SDRAM получается 200 МГц). Основное отличие DDR2 от DDR - вдвое большая частота работы шины, по которой данные передаются в буфер микросхемы памяти. При этом, чтобы обеспечить необходимый поток данных, передача на шину осуществляется из четырёх мест одновременно. Итоговые задержки оказываются выше, чем для DDR.

DDR3 SDRAM (англ. double-data-rate three synchronous dynamic random access memory — синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, тип 3) — это тип оперативной памяти используемой в компьютерах, разработанный как последователь DDR2 SDRAM. DDR3 обещает сокращение потребления энергии на 40% по сравнению с модулями DDR2, благодаря применению 90-нм (в дальнейшем 65-нм и 50-нм) технологии производства, что позволяет снизить эксплуатационные токи и напряжения (1,5 В, по сравнению с 1,8 В для DDR2 и 2,5 В для DDR). "Dual-gate" транзисторы будут использоваться для сокращения утечки тока.


DDR3 SDRAM
Clock
Command
ACT
Bank 0
ACT
Bank 1
READ
B0, Col A
READ
B0, Col B
READ
B1, Col C
READ
B1, Col D
Row-to-Row Delay (tRRD)
CAS Latancy
Первая линия шириной n бит
A1
B1
C1
D1
RAS-to-CAS Delay
Вторая линия шириной n бит
A2
B2
C2
D2
Восьмая линия шириной n бит
A8
B8
C8
D8
Внешняя шина данных шириной 2n бит
A1
B1
C1
D1

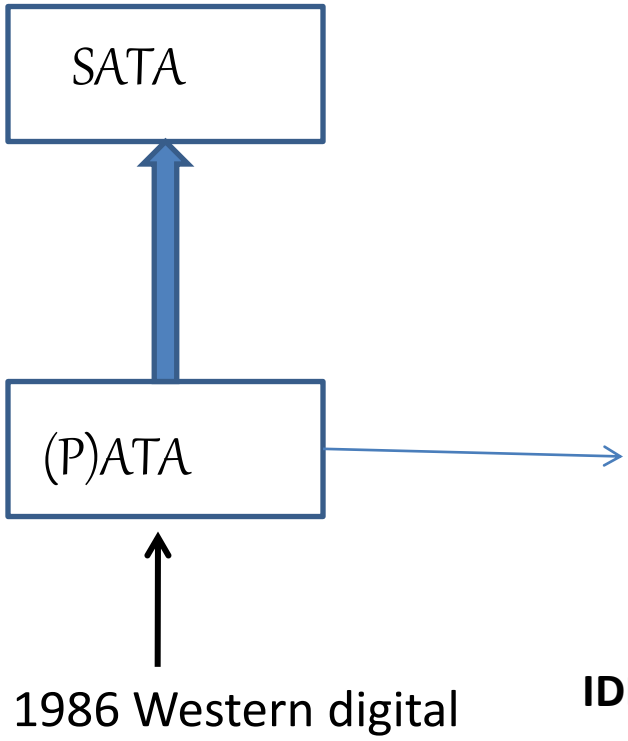


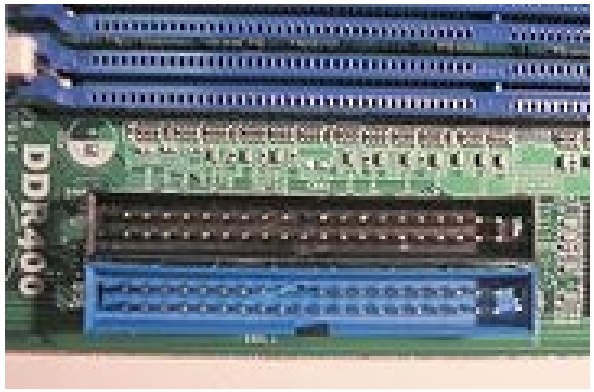
- **Передача данных между накопителем и буфером для разных типов DDR-памяти**



Модуль	Тайминги	Длительность такта, нс	Пропускная способность, Мбайт/с	Задержка доступа к ячейке закрытой строки, нс	Задержка доступа к ячейке открытой строки, нс
DDR2-800	5-5-5	2,5	6400	25	12,5
DDR2-800	4-4-4	2,5	6400	20	10
DDR3-800	6-6-6	2,5	6400	30	15
DDR3-800	5-5-5	2,5	6400	25	12,5
DDR3-1066	8-8-8	1,875	8528	30	15
DDR3-1066	7-7-7	1,875	8528	26,25	13,125
DDR3-1066	6-6-6	1,875	8528	22,5	11,25
DDR3-1333	9-9-9	1,5	10664	27	13,5
DDR3-1333	8-8-8	1,5	10664	24	12
DDR3-1333	7-7-7	1,5	10664	21	10,5
DDR3-1600	10-10-10	1,25	12800	25	12,5
DDR3-1600	9-9-9	1,25	12800	22,5	11,25
DDR3-1600	8-8-8	1,25	12800	20	10

Внешняя память





Для подсоединения к 16 битной шине ISA (шина AT)

Advanced Technology Attachment
IDE, EIDE, UDMA, ATAPI

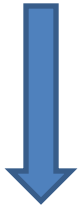
IDE Integrated Drive Electronics

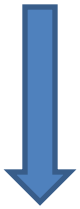
контроллер канала IDE перешёл от прямого управления приводом к обмену данными с ним по протоколу

В стандарте ATA определён интерфейс между контроллером и накопителем, а также передаваемые по нему команды

8 регистров, 8 адресов в пространстве ввода-вывода. Ширина шины данных — 16 бит.

Принцип адресации CHS: Сперва блок головок устанавливается позиционером на требуемую дорожку (Cylinder), после этого выбирается требуемая головка (Head), а затем считывается информация из требуемого сектора (Sector).





Стандарт EIDE (Enhanced IDE, т. е. «расширенный IDE»), — преодолен порог 528 МБ (504 МиБ), расширени до 8,4 ГБ.

переход от PIO (Programmed input/output, Программный ввод/вывод) к DMA (Direct memory access, Прямой доступ к памяти).

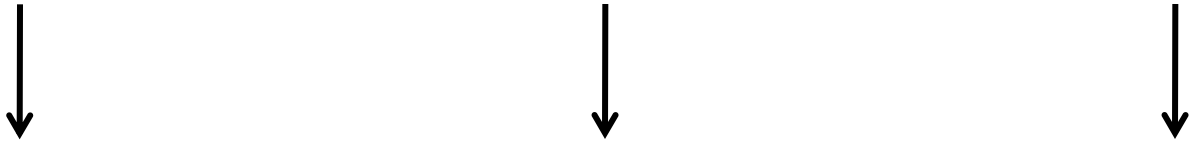
накопитель↔память без CPU
жёсткий диск → сигнал DMARQ на операцию DMA контроллеру. Если операция DMA возможна, контроллер → сигнал DMACK и жёсткий диск начинает выдавать данные в 1-й регистр (DATA), с которого контроллер считывает данные в память без участия процессора.

(ATA-3) → UltraDMA 2 (UDMA 33).
DMA Mode 2, данные передаются и по переднему, и по заднему фронту сигнала DIOR/DIOW. Это вдвое увеличивает скорость передачи данных по интерфейсу. Также введена проверка на чётность CRC.

Оригинальная спецификация ATA предусматривала 28-битный режим адресации.

228 (268 435 456 бит) секторов по 512 байт каждый→max емкость 137 ГБ (128 ГиБ).
PC BIOS поддерживал до 7,88 ГиБ (8,46 ГБ), допуская максимум 1024 цилиндра, 256 головок и 63 сектора.

ограничение адресуемого пространства в 504 МиБ (528 МБ).



LBA (Logical Block Address), что позволило адресовать до 7,88 ГиБ. → 128 ГиБ

28 разрядов (в ATA-4)

32 ГиБ

Запись 28-битного числа организована путём записи его частей в соответствующие регистры накопителя (с 1 по 8 бит в 4-й регистр, 9-16 в 5-й, 17-24 в 6-й и 25-28 в 7-й).

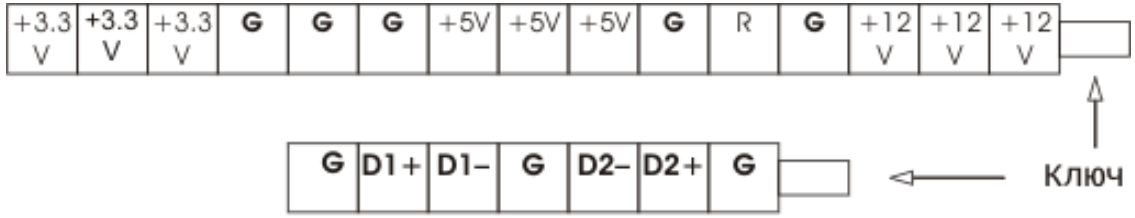
Версии стандарта ATA, скорость передачи и свойства

Стандарт	Другие названия	Добавлены режимы передачи (МБ/с)	Максимально поддерживаемый размер диска	Другие свойства	ANSI Reference
ATA-1	ATA, IDE	PIO 0,1,2 (3.3, 5.2, 8.3) Single-word DMA 0,1,2 (2.1, 4.2, 8.3) Multi-word DMA 0 (4.2)	up to 137 GB	28-bit LBA	X3.221-1994 (obsolete since 1999)
ATA-2	EIDE, Fast ATA, Fast IDE, Ultra ATA	PIO 3,4: (11.1, 16.6) Multi-word DMA 1,2 (13.3, 16,6)			X3.279-1996 (obsolete since 2001)
ATA-3	EIDE			S.M.A.R.T., Security	X3.298-1997 (obsolete since 2002)
ATA/ATAPI-4	ATAPI-4, ATA-4, Ultra ATA/33	Ultra DMA 0,1,2 (16.7, 25.0, 33.3) aka Ultra-DMA/33		Support for CD-ROM, etc., via ATAPI packet commands	NCITS 317—1998
ATA/ATAPI-5	ATA-5, Ultra ATA/66	Ultra DMA 3,4 (44.4, 66.7) aka Ultra DMA 66		80-wire cables	NCITS 340—2000
ATA/ATAPI-6	ATA-6, Ultra ATA/100	UDMA 5 (100) aka Ultra DMA 100	up to 144 PB	48-bit LBA Automatic Acoustic Management	NCITS 347—2001
ATA/ATAPI-7	ATA-7, Ultra ATA/133	UDMA 6 (133) aka Ultra DMA 133 SATA/150		SATA 1.0, Streaming feature set, long logical/physical sector feature set for non-packet devices	NCITS 361—2002
ATA/ATAPI-8	ATA-8	—		--	in progress

SATA (англ. *Serial ATA*) — последовательный интерфейс обмена данными с накопителями информации. SATA является развитием параллельного интерфейса ATA (IDE), который после появления SATA был переименован в PATA (Parallel ATA).

Стандарт	Частота шины	Пропускная способность	комментарий
SATA/150	1,5 ГГц	1,2 Гбит/с (150 МБ/с)	система кодирования 8B/10B, на каждые 8 бит полезной информации —2 служебных бита)
SATA/300	3 ГГц	2,4 Гбит/с (300 МБ/с)	SATA II
SATA/600		6 Гбит/с (600 МБ/с)	SATA Revision 3.0

Разъёмы SATA



G — заземление (англ. *Ground*)
R — зарезервировано
D1+,**D1−**,**D2+**,**D2−** — два канала передачи данных (от контроллера к устройству и от устройства к контроллеру соответственно). Для передачи сигнала используется технология LVDS, провода каждой пары (D1+, D1− и D2+, D2−) являются экранированными витыми парами.

eSATA (External SATA) — интерфейс подключения внешних устройств, поддерживающий режим «горячей замены» (англ. *Hot-plug*). Был создан несколько позже SATA (в середине 2004).

Основные особенности eSATA:

- Разъёмы менее хрупкие и конструктивно рассчитаны на большее число подключений(~9000).
- Не требует для подключения два провода: шину данных и кабель питания.
- Так же, как и SATA требует отдельного питания.
- Ограничен по длине кабеля данных (около 2 м).
- Средняя практическая скорость передачи данных выше, чем у USB или IEEE 1394.
- Существенно меньше нагружается центральный процессор.

Serial Attached SCSI

Интерфейс SAS (англ. *Serial Attached SCSI*) обеспечивает подключение по физическому интерфейсу, аналогичному SATA, устройств, управляемых набором команд SCSI. Обладая обратной совместимостью с SATA, он даёт возможность подключать по этому интерфейсу любые устройства, управляемые набором команд SCSI — не только НЖМД, но и сканеры, принтеры и др. По сравнению с SATA, SAS обеспечивает более развитую топологию, позволяя осуществлять параллельное подключение одного устройства по двум или более каналам. Также поддерживаются расширители шины, позволяющие подключить несколько SAS устройств к одному порту.

SAS и SATA2 в первых редакциях были синонимами. Но, позже производители посчитали, что реализовывать SCSI полностью в настольных компьютерах нецелесообразно, поэтому мы сейчас наблюдаем такое разделение. К слову, такие высокие скорости, заложенные в стандарте SATA на первый взгляд могут показаться излишними — обычный SATA HDD (Hard Disk Drive — жёсткий диск) использует, в лучшем случае, 40-45 % пропускной способности шины. Однако работа с буфером винчестера происходит на полной скорости интерфейса.

Типичная система с интерфейсом **SAS** состоит из следующих компонентов:

Инициаторы (англ. *Initiators*)
Инициатор — устройство, которое порождает запросы на обслуживание для *целевых устройств* и получает подтверждения по мере исполнения запросов.

Целевые устройства (англ. *Targets*)
Целевое устройство содержит логические блоки и целевые порты, которые осуществляют приём запросов на обслуживание, исполняет их; после того, как закончена обработка запроса, инициатору запроса отсылается подтверждение выполнения запроса. Целевое устройство может быть как отдельным жёстким диском, так и целым дисковым массивом.

Подсистема доставки данных (англ. *Service Delivery Subsystem*)
Является частью системы ввода-вывода, которая осуществляет передачу данных между инициаторами и целевыми устройствами. Обычно подсистема доставки данных состоит из кабелей, которые соединяют инициатор и целевое устройство. Дополнительно, кроме кабелей в состав подсистемы доставки данных могут входить *расширители SAS*.

Расширители (англ. *Expanders*)
Расширители SAS — устройства, входящие в состав подсистемы доставки данных и позволяют облегчить передачи данных между устройствами SAS, например, позволяет соединить несколько целевых устройств SAS к одному порту инициатора. Подключение через расширитель является абсолютно прозрачным для целевых устройств.

SCSI (англ. *Small Computer System Interface*) — интерфейс, разработанный для объединения на одной шине различных по своему назначению устройств, таких как жёсткие диски, накопители на магнитооптических дисках, приводы CD, DVD, стримеры, сканеры, принтеры

SE (англ. *single-ended*) - ассиметричный SCSI, для передачи каждого сигнала используется отдельный проводник.

50-контактный неэкранированный разъем для внутрисистемных соединений и аналогичный экранированный разъем типа Centronics (Alternative 2) для внешних подключений. Передача сигналов осуществляется 50 контактным кабелем типа - A-50 на 8 разрядной (битной) шине.

LVD (англ. *low-voltage-differential*) — интерфейс дифференциальной шины низкого напряжения, сигналы положительной и отрицательной полярности идут по разным физическим проводам - витой паре. На один сигнал приходится по одной витой паре проводников. Используемое напряжение при передаче сигналов ±1,8 В.

Для 8 битной шины предусматривался кабель типа A, который как и в SCSI-1 поддерживал 50-контактными разъемами типа D с уменьшенным шагом выводов (Alternative 1). Разъемы типа Centronics (Alternative 2) в SCSI-2 построенны 8 и 16 битной шине. Передача информации осуществляется по 68-контактным кабелям типа - A-68 и P-68(Wide). Для 32 битной версии шины был предусмотрен тип кабеля B, который должен был параллельно подключаться одновременно с кабелем A в одно устройство. Однако кабель B не получил широкого признания и из стандарта SCSI-3 исключен.

HVD (англ. *high-voltage-differential*) — интерфейс дифференциальной шины высокого напряжения, отличается от LVD повышенным напряжением и специальными приемопередатчиками.

Кабеля A-68 и P-68 поддерживались экранированными, либо неэкранированными разъемами типа D. Кабеля в SCSI-3 снабжены фиксаторами-защелками, а не проволочными кольцами, как разъемы Centronics. Начиная с этой версии SCSI в массивах накопителей используется 80-контактный разъем, называемый Alternative 4. Накопители с таким разъемом поддерживают "горячее" подключение устройств

Обзор интерфейсов SCSI

Наименование	**Разрядность шины**	**Частота шины**	**Пропускная способность**	**Максимальная длина кабеля**	**Максимальное количество устройств**
SCSI	8 бит	5 МГц	5 МБайт/сек	6 м	8
Fast SCSI	8 бит	10 МГц	10 МБайт/сек	1,5-3 м	8
Wide SCSI	16 бит	10 МГц	20 МБайт/сек	1,5-3 м	16
Ultra SCSI	8 бит	20 МГц	20 МБайт/сек	1,5-3 м	5-8
Ultra Wide SCSI	16 бит	20 МГц	40 МБайт/сек	1,5-3 м	5-8
Ultra2 SCSI	8 бит	40 МГц	40 МБайт/сек	12 м	8
Ultra2 Wide SCSI	16 бит	40 МГц	80 МБайт/сек	12 м	16
Ultra3 SCSI	16 бит	40 МГц DDR	160 МБайт/сек	12 м	16
Ultra-320 SCSI	16 бит	80 МГц DDR	320 МБайт/сек	12 м	16

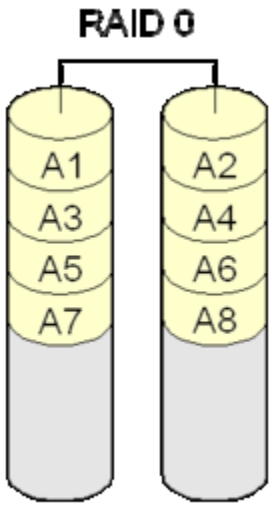
Команды SCSI посылаются в виде блоков описания команды (англ. *Command Descriptor Block, CDB*). Длина каждого блока может составлять 6, 10, 12 или 16 байт. В последних версиях SCSI блок может иметь переменную длину. Блок состоит из однобайтового кода команды и параметров команды.

Все команды SCSI делятся на четыре категории: N (non-data), W (запись данных от инициатора целевым устройством), R (чтение данных) и B (двусторонний обмен данными). Всего существует порядка 60 различных команд SCSI, из которых наиболее часто используются:

- Test unit ready — проверка готовности устройства, в т.ч. наличия диска в дисководе.
- Inquiry — запрос основных характеристик устройства.
- Send diagnostic — указание устройству провести самодиагностику и вернуть результат.
- Request sense — возвращает код ошибки предыдущей команды.
- Read capacity — возвращает ёмкость устройства.
- Format Unit
- Read (4 варианта) — чтение.
- Write (4 варианта) — запись.
- Write and verify — запись и проверка.
- Mode select — установка параметров устройства.
- Mode sense — возвращает текущие параметры устройства.

Каждое устройство на SCSI-шине имеет как минимум один номер логического устройства (LUN — англ. *Logical Unit Number*). В некоторых более сложных случаях одно физическое устройство может представляться набором LUN.

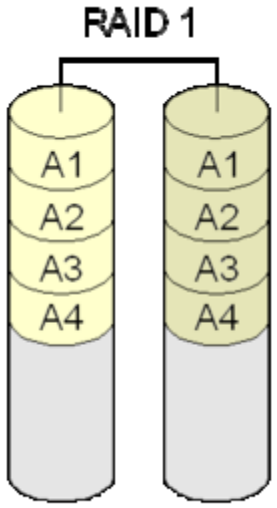
RAID (англ. redundant array of independent/inexpensive disks) избыточный массив независимых/недорогих жёстких дисков — матрица из нескольких дисков управляемых контроллером, взаимосвязанных скоростными каналами и воспринимаемых как единое целое. Обеспечивает отказоустойчивость, в случае повреждения одного из дисков, восстановление данных на нём происходит автоматически.
Калифорнийский университет в Беркли представил следующие уровни спецификации RAID, которые были приняты как стандарт де-ф



- **RAID 0** представлен как неотказоустойчивый дисковый массив.

RAID 0 («Striping») — дисковый массив из двух или более жёстких дисков с отсутствием избыточности. Информация разбивается на блоки данных (Ai) и записывается на оба/несколько дисков одновременно.
(+): За счёт этого существенно повышается производительность (от количества дисков зависит кратность увеличения производительности).
(+): RAID 0 может быть реализован как программно, так и аппаратно.
(−): Страдает надёжность всего массива (при выходе из строя любого из входящих в RAID 0 винчестеров полностью и безвозвратно пропадает вся информация). В соответствии с теорией вероятностей, надёжность массива RAID 0 равна произведению вероятностей безотказной работы составляющих его дисков, каждая из которых меньше единицы, таким образом совокупная надёжность заведомо ниже надёжности любого из дисков.

•**RAID 1** определён как зеркальный дисковый массив.

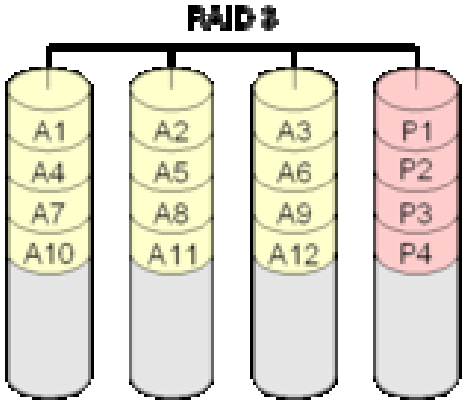


(+): Обеспечивает приемлемую скорость записи и выигрыш по скорости чтения при распараллеливании запросов.
(+): Имеет высокую надёжность — работает до тех пор, пока функционирует хотя бы один диск в массиве.
(-): Недостаток заключается в том, что приходится выплачивать стоимость двух жёстких дисков, получая полезный объем одного жёсткого диска (классический случай, когда массив состоит из двух дисков).

•**RAID 2** зарезервирован для массивов, которые применяют код Хемминга.

В массивах такого типа диски делятся на две группы — для данных и для кодов коррекции ошибок, причем если данные хранятся на **n** дисках, то для складирования кодов коррекции необходимо **n – 1** дисков. Данные записываются на соответствующие винчестеры так же, как и в RAID 0, они разбиваются на небольшие блоки по числу дисков, предназначенных для хранения информации. Оставшиеся диски хранят коды коррекции ошибок, по которым в случае выхода какого-либо винчестера из строя возможно восстановление информации. Метод Хемминга давно применяется в памяти типа ECC и позволяет на лету исправлять однократные и обнаруживать двукратные ошибки.

•**RAID 3, 4, 5** используют чётность для защиты данных от одиночных неисправностей



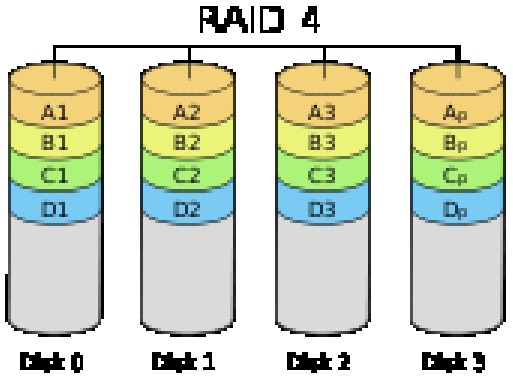
Структура массива RAID 3 такова: в массиве из n дисков данные разбиваются на блоки размером 1 байт и распределяются по n − 1 дискам, а еще один диск используется для хранения блоков четности. В RAID 2 для этой цели стояло n − 1 дисков, но большая часть информации на этих дисках использовалась только для коррекции ошибок на лету, а для простого восстановления в случае поломки диска достаточно меньшего ее количества, хватает и одного выделенного винчестера.

Соответственно, отличия RAID 3 от RAID 2 очевидны: невозможность коррекции ошибок на лету и меньшая избыточность.

(+): скорость чтения и записи данных высока, а для создания массива требуется совсем немного дисков, всего три.

(-): массив этого типа хорош только для однозадачной работы с большими файлами, так как наблюдаются проблемы со скоростью при частых запросах данных небольшого обьёма.

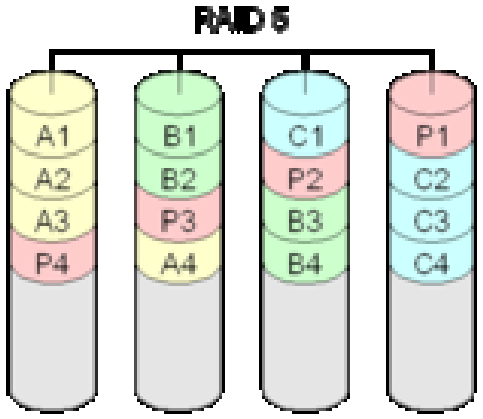
(-): большая нагрузка на контрольный диск, что приводит к тому, что его надёжность сильно падает по сравнению с дисками с данными.

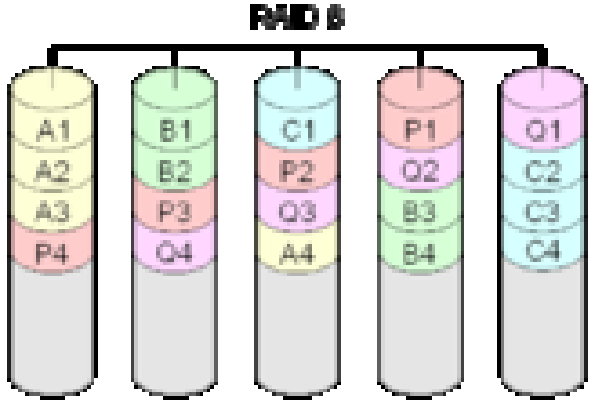


RAID 4 похож на RAID 3, но отличается от него тем, что данные разбиваются на блоки, а не на байты. Таким образом, удалось «победить» проблему низкой скорости передачи данных небольшого объема. Запись же производится медленно из-за того, что четность для блока генерируется при записи и записывается на единственный диск. Используются массивы такого типа очень редко

Большой недостаток уровней RAID от 2-го до 4-го — это наличие отдельного диска (или дисков), хранящего информацию о четности. Скорость выполнения операций считывания достаточно высока, так как не требует обращения к этому диску. Но при каждой операции записи на нем изменяется информация, поэтому схемы RAID 2—4 не позволяют проводить параллельные операции записи. **RAID5** не имеет этого недостатка. Блоки данных и контрольные суммы циклически записываются на все диски массива, отсутствует выделенный диск для хранения информации о четности, нет асимметричности конфигурации дисков. Самый популярный из уровней, в первую очередь благодаря своей экономичности. Жертвуя ради избыточности ёмкостью всего одного диска из массива, мы получаем защиту от выхода из строя любого из винчестеров тома. На запись информации на том **RAID5** тратятся дополнительные ресурсы, так как требуются дополнительные вычисления, зато при чтении (по сравнению с отдельным винчестером) имеется выигрыш, потому что потоки данных с нескольких накопителей массива распараллеливаются

- **RAID 6** используют чётность для защиты данных от двойных неисправностей



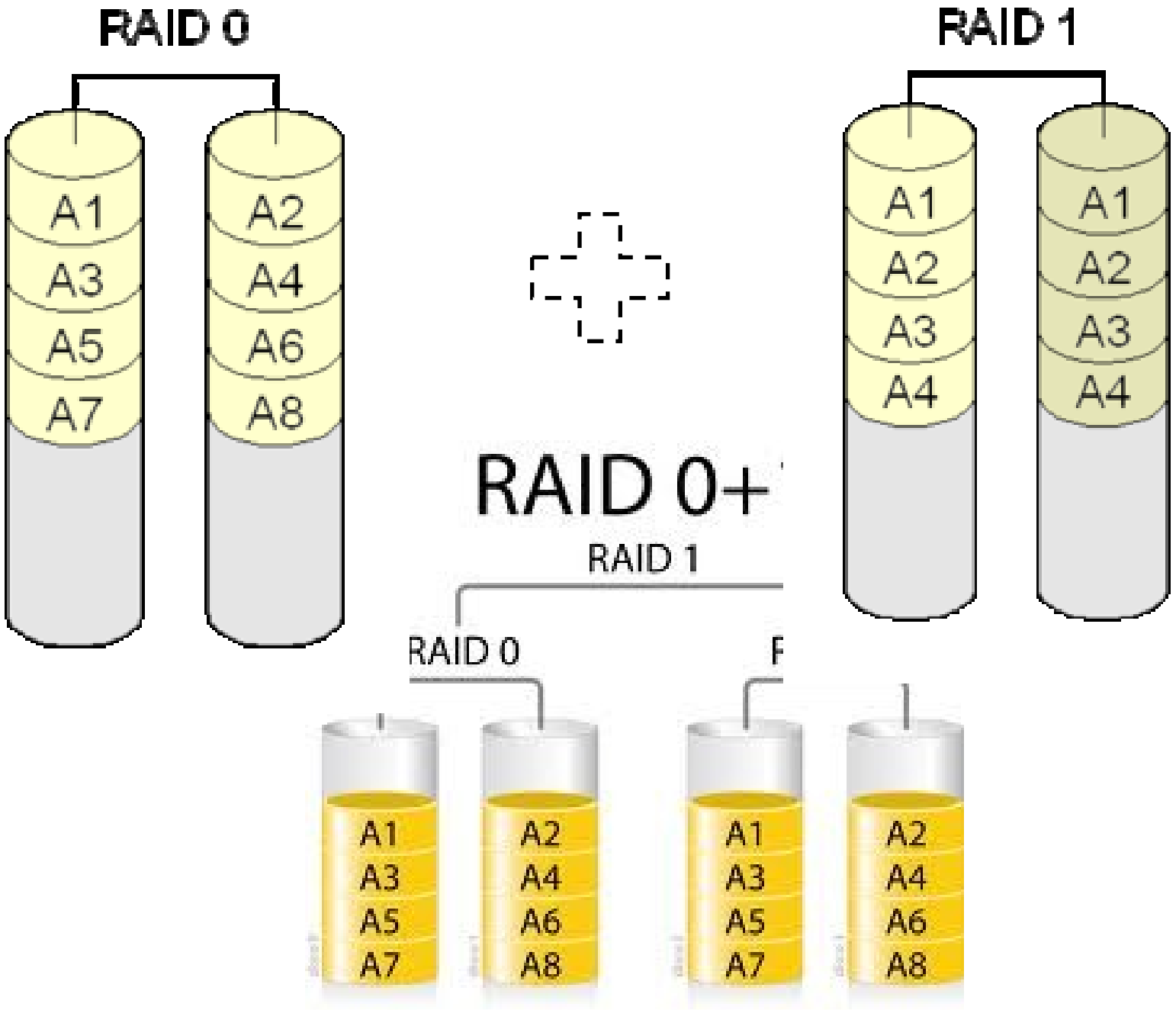


RAID 6 (Advanced Data Guarding) — похож на **RAID 5**, но имеет более высокую степень надежности — под контрольные суммы выделяется емкость 2-х дисков, рассчитываются 2 суммы по разным алгоритмам. Требует более серьезный процессор контроллера — сложная материнская плата. Обеспечивает работоспособность после «смерти» одновременно 2-х дисков. Для организации массива требуется минимум 4 диска.

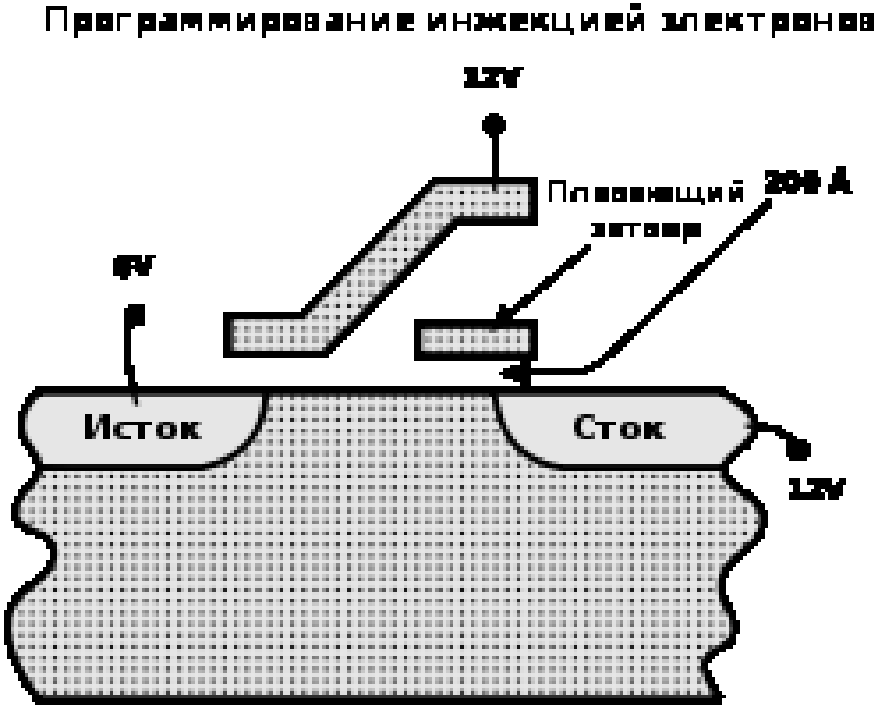
RAID 7 — зарегистрированная марка компании Storage Computer Corporation. Структура массива такова: на n − 1 дисках хранятся данные, один диск используется для складирования блоков четности. Но добавилось несколько важных деталей, призванных ликвидировать главный недостаток массивов такого типа: кэш данных и быстрый контроллер, заведующий обработкой запросов. Это позволило снизить количество обращений к дискам для вычисления контрольной суммы данных. В результате удалось значительно повысить скорость обработки данных (кое-где в пять и более раз).

Помимо базовых уровней RAID 0 — RAID 5, описанных в стандарте, существуют комбинированные уровни RAID 1+0, RAID 3+0, RAID 5+0, RAID 1+5, которые различные производители интерпретируют каждый по-своему.

RAID 1+0 — это сочетание зеркалирования и чередования.
Нынешние контроллеры используют этот режим по умолчанию для RAID 1. То есть, 1 диск основной, 2-й диск — зеркало, причем чтение производится с них поочередно, как для RAID 0. Собственно, сейчас можно считать что RAID 1 и RAID 1+0 — это просто разное название одного и того же метода аппаратного зеркалирования дисков. Но не стоит забывать, что полноценный RAID 1+0 должен содержать как минимум 4 диска.
RAID 5+0 — это чередование томов 5-го уровня. RAID 1+5 — зеркалирование «пятерок». И так далее.


RAID 0
A1
A3
A5
A7
A2
A4
A6
A8
RAID 1
A1
A2
A3
A4
A1
A2
A3
A4
RAID 0+
RAID 1
RAID 0
A1
A3
A5
A7
A2
A4
A6
A8
A1
A3
A5
A7
A2
A4
A6
A8

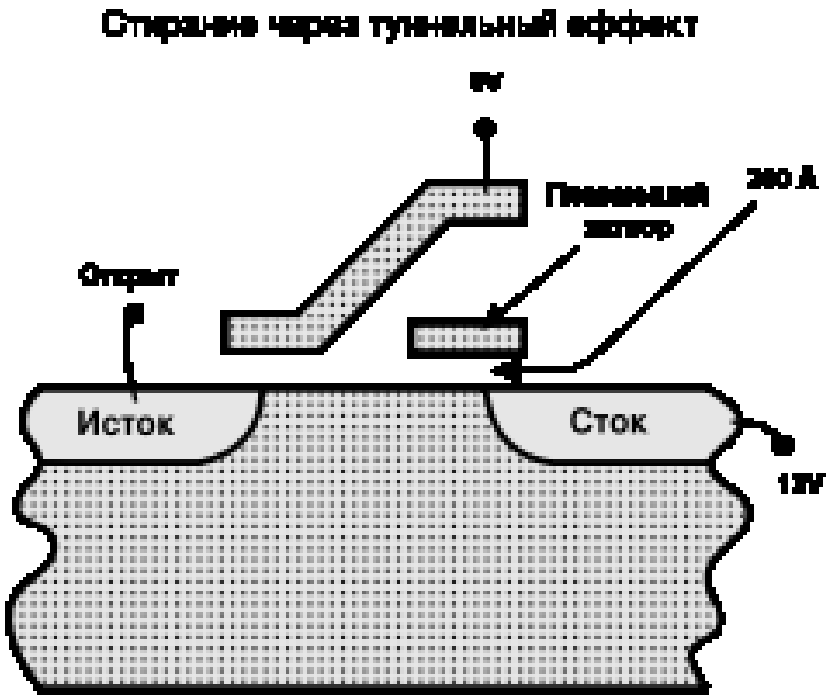
Флеш-память



Флеш-память хранит информацию в массиве транзисторов с плавающим затвором, называемых ячейками (англ. cell). В традиционных устройствах с одноуровневыми ячейками (англ. single-level cell, SLC), каждая из них может хранить только один бит. Некоторые новые устройства с многоуровневыми ячейками (англ. multi-level cell, MLC) могут хранить больше одного бита, используя разный уровень электрического заряда на плавающем затворе транзистора.

Флеш-память была изобретена Фудзи Масуока (Fujio Masuoka), когда он работал в Toshiba в 1984 году. Имя «флеш» было придумано также в Toshiba коллегой Фудзи, Сёдзи Ариизуми (Shoji Ariizumi), потому что процесс стирания содержимого памяти ему напомнил фотовспышку (англ. flash). Масуока представил свою разработку на IEEE 1984 International Electron Devices Meeting (IEDM), проходившей в Сан-Франциско, Калифорния. Intel увидела большой потенциал в изобретении и в 1988 году выпустила первый коммерческий флеш-чип NOR-типа.

NAND-тип флеш-памяти был анонсирован Toshiba в 1989 году на International Solid-State Circuits Conference. У него была больше скорость записи и меньше площадь чипа



NOR

В основе этого типа флеш-памяти лежит ИЛИ-НЕ элемент (англ. NOR), потому что в транзисторе с плавающим затвором низкое напряжение на затворе обозначает единицу.

Транзистор имеет два затвора: управляющий и плавающий. Последний полностью изолирован и способен удерживать электроны до 10 лет. В ячейке имеются также сток и исток. При программировании напряжением на управляющем затворе создаётся электрическое поле и возникает туннельный эффект. Некоторые электроны туннелируют через слой изолятора и попадают на плавающий затвор, где и будут пребывать. Заряд на плавающем затворе изменяет «ширину» канала сток-исток и его проводимость, что используется при чтении.

Программирование и чтение ячеек сильно различаются в энергопотреблении: устройства флеш-памяти потребляют достаточно большой ток при записи, тогда как при чтении затраты энергии малы.

Для стирания информации на управляющий затвор подаётся высокое отрицательное напряжение, и электроны с плавающего затвора переходят (туннелируют) на исток.

В NOR архитектуре к каждому транзистору необходимо подвести индивидуальный контакт, что увеличивает размеры схемы. Эта проблема решается с помощью NAND архитектуры.

NAND

В основе NAND типа лежит И-НЕ элемент (англ. NAND). Принцип работы такой же, от NOR типа отличается только размещением ячеек и их контактами. В результате уже не требуется подводить индивидуальный контакт к каждой ячейке, так что размер и стоимость NAND чипа может быть существенно меньше. Так же запись и стирание происходит быстрее. Однако эта архитектура не позволяет обращаться к произвольной ячейке.

NAND и NOR архитектуры сейчас существуют параллельно и не конкурируют друг с другом, поскольку находят применение в разных областях хранения данных.

Скорость некоторых устройств с флеш-памятью может доходить до 100 Мб/с[4]. В основном флеш-карты имеют большой разброс скоростей и обычно маркируются в скоростях стандартного CD-привода (150 Кб/с). Так указанная скорость в 100x означает 100 × 150 Кб/с = 15 000 Кб/с= 14.65 Мб/с.

В основном объём чипа флеш-памяти измеряется от килобайт до нескольких гигабайт.

В 2005 году Toshiba и SanDisk представили NAND чипы объёмом 1 Гб[5], выполненные по технологии многоуровневых ячеек, где один транзистор может хранить несколько бит, используя разный уровень электрического заряда на плавающем затворе.

Компания Samsung в сентябре 2006 года представила 8 Гб чип, выполненный по 40-нм технологическому процессу[6]. В конце 2007 года Samsung сообщила о создании первого в мире MLC (multi-level cell) чипа флеш-памяти типа NAND, выполненного по 30-нм технологическому процессу. Ёмкость чипа также составляет 8 Гб. Ожидается, что в массовое производство чипы памяти поступят в 2009 году.