ТЕМА: Понятие виртуального ресурса. Иерархия запоминающих устройств. Принцип кэширования данных.

ТИП: лекция

ЦЕЛИ :

**Учебная :**

* Определить понятия виртуального ресурса;
* Изучить принцип кэширования данных;

**Развивая :**

* • развивать логическое мышление , внимание , память ;
* • развивать кругозор .

Воспитательная :

• развивать познавательный интерес к предмету.

Структура занятия

1 . Организационный момент ( 3 мин )

2 . Сообщения темы , цели занятия ( 5 мин )

3 . Актуализация знаний ( 15 мин )

4 . Выдача теоретического материала ( 30 мин )

5 . Проверка полученных знаний ( 10 мин )

6 . Подведение итогов , выставление оценок ( 5 мин )

7 . Выдача домашнего задания ( 2 мин )

Литература.

1. БатаевА.В. Операционные системы и среды : учебник для студ. учреждений сред. проф. образования / А. В. Батаев, Н. Ю. Налютин, С. В. Синицын. - М. : Издательский центр «ДХаде­мия», 2014. - 272 с.
2. Иртеrов Д. В. Введение в операционные системы. ­ 2-е изд., перераб. и доп. ­ СПб.: БХВ..Петербурr, 2008. ­ 1040 с.
3. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. — СПб.: Питер, 2015. — 1120 с.

Вопрос актуализации знаний:

1. Работа с внешними устройствами.
2. Интерфейсы ввода-вывода
3. Понятие прерывания

План

[Адресация к памяти 2](#_Toc533198976)

[Классификация алгоритмов распределения памяти 4](#_Toc533198977)

[Распределение памяти фиксированными разделами 5](#_Toc533198978)

[Распределение памяти динамическими разделами 5](#_Toc533198979)

[Перемещаемые разделы 5](#_Toc533198980)

[Свопинг и виртуальная память 6](#_Toc533198981)

[Страничное распределение памяти 7](#_Toc533198982)

[Сегментное распределение памяти 10](#_Toc533198983)

[Сегментно-страничное распределение памяти. 12](#_Toc533198984)

[Кэширование данных 13](#_Toc533198985)

[Иерархия запоминающих устройств в компьютере 13](#_Toc533198986)

[Кэш-память 14](#_Toc533198987)

[Отображение основной памяти на кэш 16](#_Toc533198988)

Особая роль памяти объясняется тем, что процессор может выполнять инструкции программы только в том случае, если они находятся в оперативной памяти компьютера. Память распределяется как между модулями прикладных программ, так и между модулями самой операционной системы. Функции распределения и защиты памяти в ОС выполняет подсистема управления памятью.

# **Адресация к памяти**

Для идентификации переменных и команд на различных этапах жизненного цикла программы используются символьные имена, виртуальные адреса и физические адреса ячеек памяти.

**Символьные имена** присваиваются пользователем при написании программы на алгоритмическом языке.

**Виртуальные адреса**, называемые иногда математическими, или логическими адресами, вырабатывает транслятор, переводящий программу на машинный язык. Поскольку во время трансляции не известно, в какое место памяти будет загружена программа, то транслятор присваивает переменным и командам виртуальные (условные) адреса, обычно считая по умолчанию, что начальным адресом программы будет нулевой адрес.

**Физические адреса** соответствуют номерам ячеек оперативной памяти, где в действительности расположены или будут расположены переменные и команды программы.

Существуют два принципиально разных подхода к преобразованию виртуальных адресов в физические: статический и динамический.

При **статическом** подходе замена виртуальных адресов на физические выполняется один раз для каждого процесса во время начальной загрузки программы при помощи специального модуля – **перемещающего загрузчика**. При этом внутри программы меняются все адресно-зависимые указатели.

При **динамическом** подходе программа загружается в память в неизменном виде в виртуальных адресах, а при выполнении программы при каждом обращении к оперативной памяти выполняется преобразование виртуального адреса в физический на основании смещения адресов для данного процесса, хранящегося в специальном регистре операционной системы.

Второй способ является более гибким, т. к. программа жестко не привязана к первоначально выделенному ей участку памяти. Но первый вариант более экономичен, т. к. преобразование адресов выполняется только один раз при загрузке программы.

Совокупность виртуальных адресов процесса называется **виртуальным адресным пространством**, который одинаков у всех процессов системы.

Например, при использовании 32-разрядных виртуальных адресов этот диапазон 00000000 16 … FFFFFFFF 16 (232=4 Гбайт при адресации к ячейкам памяти размером 1 байт). Совпадение виртуальных адресов переменных и команд различных процессов не приводит к конфликтам, т. к. операционная система отображает их на разные физические адреса. При этом ОС отображает либо все виртуальное адресное пространство, либо только определенную его часть.

Необходимо различать **максимально возможное** адресное пространство процесса (потенциальное, определяемое разрядностью схем адресации) и **назначенное (выделенное)** процессу виртуальное адресное пространство – действительно нужное процессу для работы. В ходе выполнения процессу при необходимости может быть увеличено выделенное виртуальное адресное пространство.

Сегодня типична ситуация, когда объем виртуального адресного пространства превышает доступный объем оперативной памяти. В этом случае для хранения данных виртуального адресного пространства процесса, не помещающихся в оперативную память, используется внешняя память на жестком диске. Именно на этом принципе основана **виртуальная память** – современный механизм, используемый в ОС для управления памятью.

Следует отметить, что виртуально адресное пространство и виртуальная память – различные механизмы и они не обязательно реализуются в операционной системе одновременно.

Содержимое назначенного процессу виртуального адресного пространства, т. е. коды команд, исходные и промежуточные данные, а также результаты вычислений, представляют собой **образ процесса**.

Обычно виртуальное адресное пространство процесса делится на две непрерывные части: системную, необходимую операционной системе, и пользовательскую. Системная часть виртуального адресного пространства является идентичной для всех процессов. Поэтому при смене активного процесса заменяется только пользовательская часть. Причем системная часть виртуальной памяти в ОС любого типа включает область, подвергаемую вытеснению на диск, и область, на которое вытеснение не распространяется.

# **Классификация алгоритмов распределения памяти**

На рис. 2.2 приведена классификация алгоритмов распределения памяти, в которой все они разделены на два класса – в одних используется перемещение сегментов процесса между оперативной памятью и диском, в других – не используется.



Рис. 2.2. Классификация алгоритмов распределения памяти

## **Распределение памяти фиксированными разделами**

При применении этого алгоритма память разбивается на несколько фиксированных (не одинаковых) областей, называемых разделами. Очередной процесс, поступивший на выполнение, помещается либо в общую очередь, либо в очередь к некоторому разделу, подходящему по размерам.

При очевидном преимуществе – простоте реализации, данный метод имеет существенный недостаток – жесткость: число одновременно выполняемых процессов фиксировано и не учитываются их реальные потребности в памяти. Такой алгоритм управления памяти применялся в ранних мультипрограммных ОС, а сейчас – в системах реального времени.

## **Распределение памяти динамическими разделами**

В этом алгоритме память заранее не делится на разделы. Каждому вновь поступающему приложению на этапе создания процесса выделяется вся необходимая ему память непрерывным разделом. После завершения процесса выделенный ему раздел памяти освобождается, и на это место может быть загружен новый процесс. Однако маловероятно, что необходимая новому процессу память точно соответствует какому-то освободившемуся разделу.

В результате этот метод имеет серьезный недостаток – фрагментацию памяти: наличие большого числа несмежных участков свободной памяти маленького размера, недостаточного ни для одного создаваемого процесса.

## **Перемещаемые разделы**

Одним из методов борьбы с фрагментацией является перемещение всех занятых участков памяти в сторону старших или младших адресов так, чтобы вся свободная память образовала единую область. Эта процедура называется сжатием и на нее требуется достаточно много времени.

Так как программы перемещаются по оперативной памяти в ходе своего выполнения, то при применении этого алгоритма распределения памяти невозможно использовать настройку адресов с помощью перемещающего загрузчика. Здесь можно использовать только динамическое преобразование адресов.

## **Свопинг и виртуальная память**

В мультипрограммном режиме помимо активного процесса, который исполняется процессором, имеются приостановленные процессы, находящиеся либо в состоянии ожидания, либо в состоянии готовности и стоящие в очереди к процессору. Образы таких неактивных процессов могут быть временно выгружены на диск. Когда подходит очередь выполнения выгруженного процесса, его образ возвращается с диска в оперативную память. Если в памяти нет свободного места, то на диск сбрасывается один или несколько неактивных процессов.

Подмена оперативной памяти дисковой памятью позволяет повысить уровень мультипрограммирования – объем оперативной памяти теперь не столь жестко ограничивает количество одновременно выполняемых процессов. Суммарный объем памяти, занимаемой образами этих процессов, может существенно превосходить размер физической оперативной памяти компьютера.

**Виртуальным** называется ресурс, который пользователю или пользовательской программе представляется обладающим свойствами, которыми он в действительности не обладает.

Виртуализация памяти может быть осуществлена на основе двух различных подходов:

·  **свопинг** – образы процессов выгружаются на диск и возвращаются в оперативную память **целиком**, т. е. образы процессов в каждый момент времени существуют в одном экземпляре – в оперативной памяти или на диске;

·  **виртуальная память** – между оперативной памятью и диском перемещаются **части** (сегменты, страницы и т. п.) образов процессов, т. е. на диске постоянно хранятся полные образы всех процессов, а в оперативной памяти – их части.

Свопинг является частным случаем виртуальной памяти и более простым в реализации способом совместного использования оперативной памяти и диска. Однако ему присуща избыточность, т. к. при подкачке с диска не весь программный код образа процесса требуется для исполнения в текущем кванте его времени. Перемещение избыточных данных замедляет работу системы и приводит к неэффективному использованию памяти. Кроме того, невозможно загрузить на выполнение процесс, адресное пространство которого превышает имеющуюся свободную память. Поэтому свопинг в современных операционных системах почти не используется, а применяется механизм виртуальной памяти.

# **Страничное распределение памяти**

При применении этого алгоритма виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые **виртуальными страницами**. В общем случае размер виртуального адресного пространства процесса не кратен размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью.

Вся оперативная память компьютера также делится на части такого же размера, называемые **физическими страницами** (блоками, кадрами), равными степени двойки: 512, 1024, 4096 байт и т. д.

При создании процесса операционная система загружает в оперативную память несколько его виртуальных страниц (начальные страницы кодового сегмента и сегмента данных).

Копия всего виртуального адресного пространства каждого процесса постоянно находится на диске.

Для каждого процесса операционная система создает **таблицу страниц,** содержащую записи обо всех виртуальных страницах процесса. Пример страничного распределения памяти для двух процессов приведен на рис. 2.3.

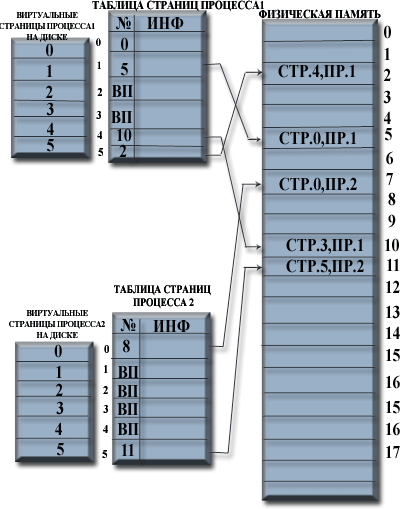


Рис. 3. Пример страничного распределения памяти   
для двух процессов

Каждая запись страницы (**дескриптор страницы**) включает следующую информацию о соответствующей виртуальной странице процесса:

·  **признак присутствия** (1/0) виртуальной страницы в оперативной памяти;

·  **номер физической страницы**, в которую загружена данная виртуальная страница;

·  **признак модификации** страницы (1/0) – изменялась страница, находящаяся в памяти, или нет;

·  **признак обращения** к странице (1/0), который устанавливается 1 при каждом обращении по адресу, относящемуся к данной странице.

Информация из таблиц страниц используется для решения вопроса о необходимости перемещения той или иной страницы между памятью и диском, а также для преобразования виртуального адреса в физический.

Сами таблицы размещаются в оперативной памяти, а адрес их размещения включается в контекст соответствующего процесса. При активации очередного процесса операционная система загружает адрес его таблицы страниц в специальный регистр процессора.

При каждом обращении к памяти выполняется поиск номера виртуальной страницы, содержащей требуемый адрес, затем по этому номеру определяется нужный элемент таблицы страниц, и из него извлекается описывающая страницу информация.

По признаку присутствия определяется, надо или нет загружать страницу. Если нужная страница в данный момент выгружена на диск, то выполняется так называемое **страничное прерывание** – выполняющийся процесс переводится в состояние ожидания (загрузки страницы), а в это время активируется другой процесс, находившийся в очереди готовых процессов.

По номеру физической страницы выполняется преобразование виртуального адреса в физический – простота такого преобразования определяется размером страниц, кратным степени двойки (в операционных системах для процессоров Pentium компании Intel размер страниц обычно 4096 байт = 4 Кбайт).

Любой виртуальный адрес процесса состоит из двух частей: старшие разряды соответствуют номеру виртуальной страницы, а младшие – номеру ячейки в странице. Замена виртуального адреса на физический состоит в замещении старших разрядов – номер виртуальной страницы меняется на номер соответствующей физической страницы памяти.

Размер страницы влияет на производительность системы и эффективность использования памяти. Чем меньше размер страницы, тем меньше фиктивная ее часть и неиспользуемая часть кода процесса, но учащается процесс смены страниц в оперативной памяти.

Размер страниц влияет и на количество записей в таблице страниц. Учитывая, что в современных процессорах максимальный объем адресного пространства процесса, как правило, не меньше 4 Гбайт, то при размере страницы 4 Кбайт и длине записи 4 байта для хранения таблицы страниц может потребоваться до 4 Мбайт памяти.

Так как у объемных процессов таблица страниц занимает несколько страниц физической памяти, то часть из них (менее интенсивно используемых) может быть вытеснена на диск. Для этого общее виртуальное пространство страниц процесса разбивается на разделы и для каждого раздела создается собственная таблица страниц, которая занимает в памяти одну страницу (для процессоров Pentium – на 1024 записи). Информация о страницах с таблицами находится в таблице разделов, которая постоянно расположена в оперативной памяти.

Страничное распределение памяти может быть реализовано в упрощенном варианте без выгрузки страниц на диск. Все виртуальные страницы процессов находятся в памяти. В этом случае нет преимуществ, которые дает виртуальная память, зато можно успешно бороться с фрагментацией памяти, т. к. все страницы имеют фиксированные размеры, а виртуальные страницы жестко не привязаны к физическим страницам.

# **Сегментное распределение памяти**

При этом методе виртуальное адресное пространство процесса делится на части – **сегменты,** размер которых определяется с учетом смыслового значения содержащейся в них информации. Отдельный сегмент может представлять собой подпрограмму, массив данных и т. п. «Осмысленность» сегментов упрощает их защиту.

Деление виртуального адресного пространства на сегменты осуществляется компилятором на основе указаний программиста или по умолчанию, в соответствии с принятыми в системе соглашениями.

Сегменты не упорядочиваются друг относительно друга, так что общего для сегментов линейного виртуального адреса не существует. Виртуальный адрес задается парой чисел: номером сегмента и линейным виртуальным адресом внутри сегмента.

Максимальный размер каждого сегмента при 32-разрядной организации процессора равен 4 Гбайт.

При загрузке процесса в оперативную память помещается только часть его сегментов, полная копия виртуального адресного пространства находится в дисковой памяти.

На этапе создания процесса во время загрузки его образа в оперативную память создается **таблица сегментов** процесса (аналогичная таблице страниц), в которой для каждого сегмента указываются:

·  базовый физический адрес сегмента в оперативной памяти;

·  размер сегмента;

·  правила доступа к сегменту;

·  признаки модификации, присутствия и обращения к данному сегменту, а также некоторая другая информация.

**Достоинства сегментного распределения памяти:**

·  если виртуальные адресные пространства нескольких процессов включают один и тот же сегмент, то в таблицах сегментов этих процессов делаются ссылки на один и тот же участок оперативной памяти, в который данный сегмент загружается в единственном экземпляре, и который в этом случае называется **разделяемой памятью;**

·  возможно задание [дифференцированных](https://pandia.ru/text/category/differentciya/) прав доступа процесса к его сегментам, например, только чтение или запись.

**Недостатки:**

·  более громоздкий механизм преобразования виртуальных адресов процесса в физические. При страничной организации страницы имеют одинаковый размер, кратный степени двойки. Поэтому ОС заносит в таблицы страниц не полный адрес физической памяти, а только номер физической страницы, который одновременно представляет собой старшие разряды физического адреса любой ячейки этой страницы при преобразовании адресов. При сегментной организации сегменты могут начинаться с любого физического адреса памяти, поэтому в таблице сегментов необходимо задавать полный начальный физический адрес;

·  избыточность, т. к. единицей перемещения между памятью и диском является сегмент, который в общем случае больше страницы;

·  фрагментация памяти, которая возникает из-за непредсказуемых размеров сегментов.

# **Сегментно-страничное распределение памяти.**

В этом методе реализуются достоинства страничного и сегментного методов распределения памяти.

Как и при сегментной организации памяти, виртуальные адресные пространства разделены на сегменты. Это позволяет определить разные права доступа к разным частям кодов и данных программ.

Однако в большинстве современных реализаций все виртуальные сегменты образуют одно непрерывное линейное виртуальное адресное пространство процесса.

Перемещение данных между памятью и диском осуществляется не сегментами, а страницами. Для этого каждый сегмент и физическая память делятся на страницы одинакового размера, что позволяет эффективно использовать память, сократив до минимума фрагментацию.

Преобразование виртуального адреса в физический адрес происходит в два этапа.

На первом этапе работают механизмы сегментации. Однако в таблице сегментов в поле базового адреса указывается не начальный физический адрес сегмента в оперативной памяти, а начальный линейный виртуальный адрес сегмента в пространстве виртуальных адресов процесса. Поэтому исходный виртуальный адрес, заданный в виде пары (номер сегмента, смещение) преобразуется в линейный виртуальный адрес.

На втором этапе работает страничный механизм, при помощи которого полученный линейный виртуальный адрес преобразуется в искомый физический адрес.

# **Кэширование данных**

## **Иерархия запоминающих устройств в компьютере**

Память компьютера представляет собой иерархию запоминающих устройств, отличающихся средним временем доступа к данным и стоимостью хранения одного бита данных (рис. 2.4).

Внешняя память (дисковая) компьютеров в настоящее время достигла объема десятков…сотен гигабайт, но скорость доступа к данным является невысокой.

Оперативная память компьютера обычно является динамической (DRAM), которая имеет среднее время доступа 10…20 нс. Для хранения данных, к которым необходимо обеспечить быстрый доступ, используется статическая оперативная память (SRAM), не требующая специальных циклов регенерации, а поэтому более быстродействующая.

Известно – чем более быстродействующая память, тем дороже ее производство. Однако пользователю хотелось бы иметь и недорогую, и быстродействующую память. Кэширование данных представляет некий компромисс в решении этой проблемы.

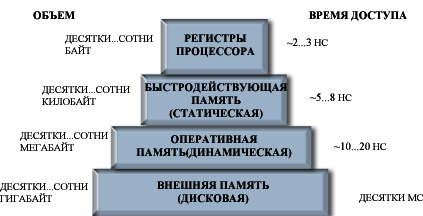


Рис. 2.4. Иерархия запоминающих устройств компьютера

## **Кэш-память**

**Кэш-память**, или просто кэш (cache), – это способ совместного функционирования двух типов запоминающих устройств (ЗУ), отличающихся временем доступа и стоимостью хранения данных, который за счет динамического копирования в быстрое ЗУ наиболее часто используемых данных из медленного ЗУ позволяет уменьшить среднее время доступа к данным и экономить более дорогую быстродействующую память.

Кэш-память прозрачна для программ и пользователей. С ней работает только операционная система.

Кэш-памятью называют не только способ организации работы двух типов запоминающих устройств, но и одно из них – быстрое ЗУ.

В современных компьютерах применяют кэширование, как оперативной памяти так и диска. Рассмотрим процесс кэширования на примере кэширования оперативной памяти, которую часто называют основной памятью.

В процессе работы системы в кэш-память заносятся данные, считываемые системой из оперативной памяти при выполнении каких-либо задач. Каждая запись в кэш-памяти включает в себя:

·  значение элемента данных;

·  адрес, который этот элемент данных имеет в основной памяти;

·  дополнительную информацию, которая используется для реализации алгоритма замещения данных в кэше и обычно включает признак модификации (1/0) и признак действительности данных (1/0).

При каждом обращении к основной памяти просматривается содержимое кэш-памяти с целью определения, не находятся ли там нужные данные. Кэш-память не является адресуемой, поэтому поиск осуществляется по содержимому. Далее возможна одна из двух ситуаций:

·  если данные обнаруживаются в кэш-памяти (**кэш-попадание**), то они считываются из нее;

·  если нужные данные отсутствуют в кэш-памяти (**кэш-промах**), то они считываются из основной памяти и одновременно с этим копируются в кэш-память.

Понятно, что эффективность кэширования зависит от вероятности кэш-попаданий, обусловленной разными факторами: объемом кэша, объемом кэшируемой памяти, алгоритмом замещения данных в кэше и другими особенностями компьютера и вычислительного процесса. Тем не менее, в большинстве систем процент кэш-попаданий не менее 90 %.

Достигается высокая степень попаданий за счет учета пространственной и временной локальности данных.

**Временная локальность**: если произошло обращение по некоторому адресу, то следующее обращение по тому же адресу с большей вероятностью произойдет в ближайшее время.

**Пространственная локальность**: если произошло обращение по некоторому адресу, то с высокой степенью вероятности в ближайшее время произойдет обращение к соседним адресам.

В начале работы системы, когда кэш-память пуста, почти каждый запрос к оперативной памяти сопровождается кэш-промахом и копированием данных в кэш. Затем, по мере накопления кэша, в полном соответствии со свойствами временной локальности возрастает вероятность обращения к данным, которые уже были использованы ранее и имеются в кэше.

Для использования свойства пространственной локальности данных в кэш-память считывается не один информационный элемент, к которому произошло обращение, а целый блок данных, расположенных в непосредственной близости к данным этого элемента.

В процессе работы содержимое кэш-памяти постоянно обновляется, данные из нее вытесняются. Алгоритм обновления данных в кэш-памяти существенно влияет на ее эффективность. Наличие в компьютере двух копий данных – в основной памяти и в кэше – порождает проблему их согласования. Если происходит запись в основную память по некоторому адресу, а содержимое этой ячейки находится в кэше, то в результате соответствующая запись в кэше становится недостоверной.

Возможны два решения проблемы: сквозная и обратная запись.

**Сквозная запись.** При каждом запросе к основной памяти, в том числе и при записи, просматривается кэш. Если данные в кэше отсутствуют, то запись выполняется только в основную память. Если же данные находятся в кэше, то запись выполняется одновременно в кэш и основную память. При вытеснении данных из кэша они просто теряются.

**Обратная запись.** Аналогично при возникновении запроса к памяти выполняется просмотр кэша. И если данных там нет, то запись выполняется только в основную память. В противном случае производится запись только в кэш-память, при этом в описателе данных кэша делается специальная отметка (признак модификации), которая указывает на то, что при вытеснении этих данных из кэша необходимо переписать их в основную память, чтобы обновить устаревшее содержимое основной памяти.

При выполнении запросов к оперативной памяти во многих вычислительных системах используется двухуровневое кэширование. Кэш первого уровня имеет меньший объем и более высокое быстродействие, чем кэш второго уровня. Кэш второго уровня играет роль основной памяти по отношению к кэшу первого уровня.

## **Отображение основной памяти на кэш**

Принцип прозрачности требует, чтобы правило отображения основной памяти на кэш-память не зависело от работы программ и пользователей. Для кэширования используются две основные схемы отображения: случайное отображение и детерминированное отображение.

При **случайном отображении** элемент оперативной памяти вместе со своим адресом может быть размещен в произвольном месте кэш-памяти. При каждом запросе к памяти выполняется поиск в кэше по адресу искомого элемента методом полного перебора. Так как такой поиск требует времени, то с целью ускорения его реализуют аппаратно, что удорожает кэш-память.

В такой кэш-памяти вытеснение старых записей происходит лишь при ее полном заполнении. Обычно вытесняются записи, к которым меньше всего не было обращений.

**Детерминированный (прямой) способ отображения** предполагает, что любой элемент основной памяти всегда отображается в одно и то же место кэш-памяти. Между номерами строк кэш-памяти и адресами основной памяти устанавливается соотношение «один ко многим»: одному номеру строки соответствует несколько адресов оперативной памяти.

Очевидно, что поиск данных в таком кэше сокращается по времени, однако вытеснение данных из кэша может происходить, когда в ней достаточно свободного места, т. к. каждая строка кэша предназначена строго для определенных адресов основной памяти.

Во многих современных процессорах кэш-память строится на основе сочетания этих двух подходов, что позволяет совместить их достоинства.

**При смешанном подходе** произвольный адрес оперативной памяти отображается не на один адрес кэш-памяти и не на любой адрес, а на некоторую группу адресов. Все группы пронумерованы. Поиск в кэше осуществляется вначале по номеру группы (детерминированно), а затем в пределах группы путем просмотра адресов записей. Выгрузка из группы выполняется лишь при ее переполнении по критерию меньшего обращения.

Таким образом, в данном способе комбинируется прямое отображение на группу и случайное отображение в пределах группы.