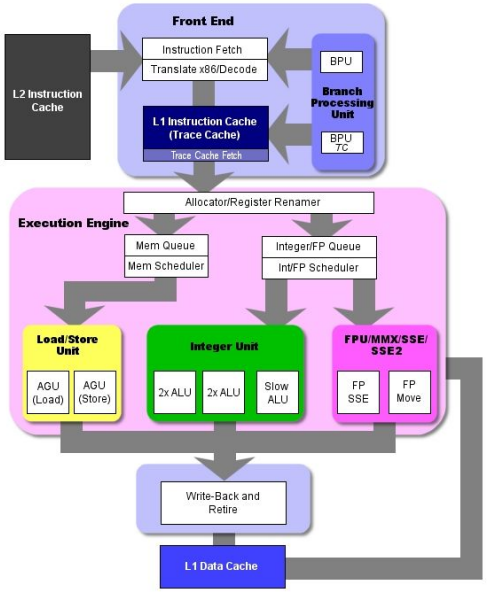
Кэширование памяти



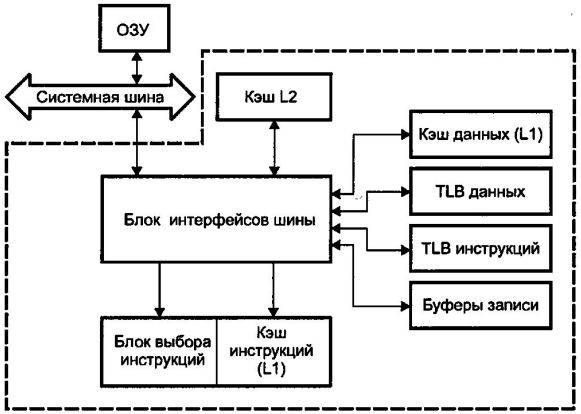
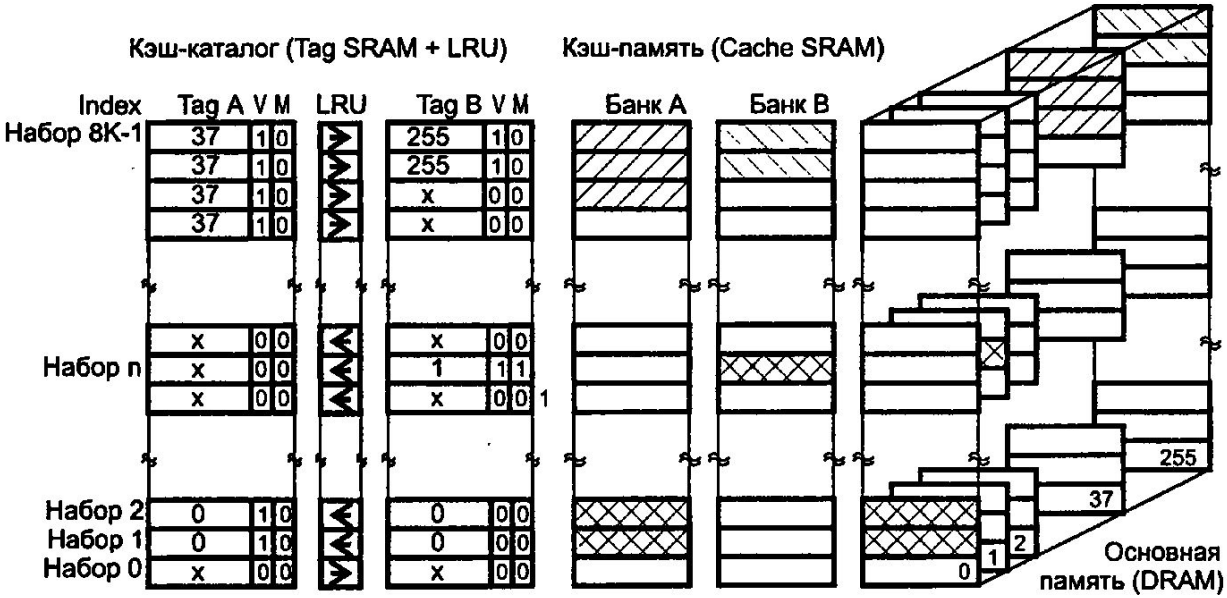


Рис. 6.4. Кэширование в процессорах Intel



Outline

Кэш память :

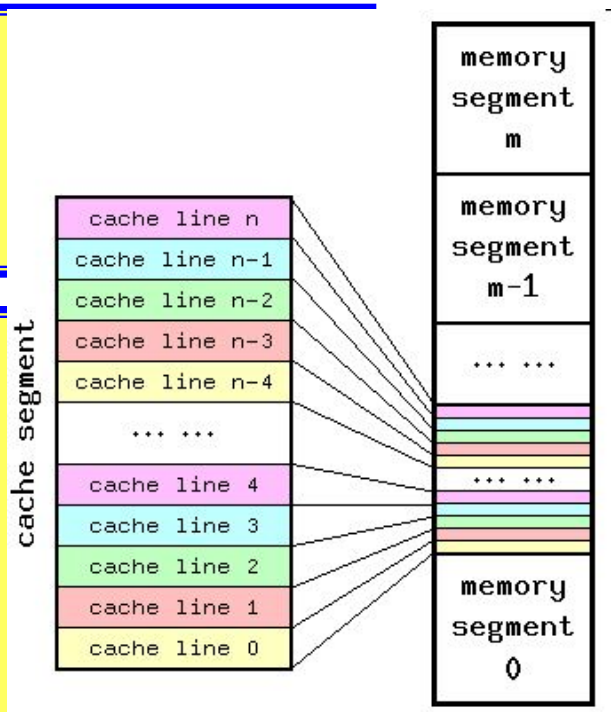
- историческая справка;
- общие понятия;
- назначение.

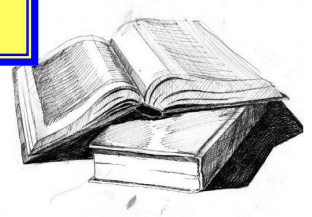
Функционрование:

- взаимодействие с прикладными программами;
- принцип работы кэш-памяти;
- алгоритмы замещения.

Отображение на кэш:

- тег;
- проблема вытеснения;
- факторы присутствия;
- пространственная и временная локальности данных;
- строка кэша;
- проблема согласования данных;
- политики записи в кэш.





Резюме к лекции и список используемой литературы

Исторический экскурс

Впервые слово «кэш» в компьютерном контексте было использовано в 1967 году во время подготовки статьи для публикации в журнале «IBM Systems Journal».

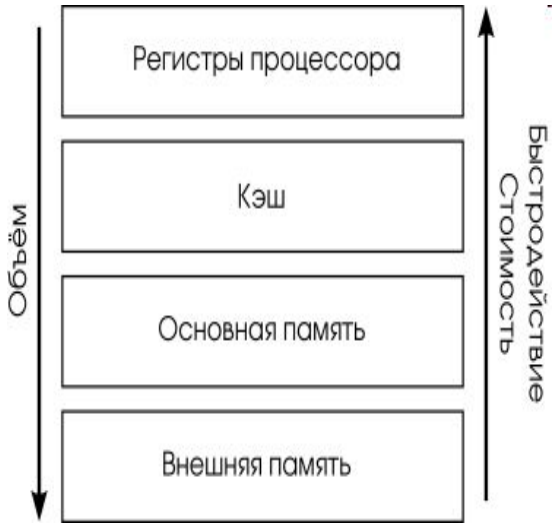
Статья касалась усовершенствования памяти в разрабатываемой модели 85 из серии IBM System/360.

Редактор журнала Лайл Джонсон попросил придумать более описательный термин, нежели «высокоскоростной буфер», но из-за отсутствия идей сам предложил слово «***кэш***».

Статья была опубликована в начале 1968 года, авторы были премированы IBM, их работа получила распространение и впоследствии была улучшена, а слово «кэш» вскоре стало использоваться в компьютерной литературе как общепринятый термин.

Общие понятия

Кэш (англ. cache - «тайник, заначка») — промежуточный буфер с быстрым доступом, содержащий информацию, которая может быть запрошена с P_{max} наибольшей вероятностью.



Основная память компьютеров реализуется на относительно медленной динамической памяти (DRAM), обращение к ней приводит к простою процессора — появляются такты ожидания (wait states). Статическая память (SRAM), построенная, как и процессор, на триггерных ячейках, по своей природе способна догнать современные процессоры по быстродействию и сделать ненужными такты ожидания (или хотя бы сократить их количество).

Разумным компромиссом для построения экономичных и производительных систем стал иерархический способ организации RAM.

Идея заключается в сочетании основной памяти большого объема на DRAM с относительно небольшой кэш-памятью на быстродействующих микросхемах SRAM.

Общие понятия. Назначение

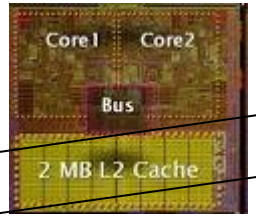
Т.о., кэш процессора - это быстродействующая память небольшого объёма, используемая для уменьшения (в среднем) времени доступа процессора к более медленной RAM.

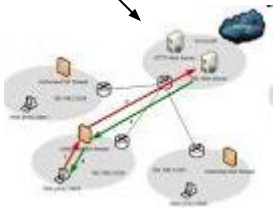
Назначение:

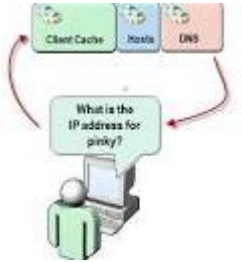
- Кэш хранит копию части данных RAM.
- Уменьшение времени доступа происходит из-за того, что большинство данных, требуемых CPU, оказываются в кэше, и количество обращений к RAM снижается.
- Кэш особенно актуален в современных системах, в которых велик разрыв между скоростью работы процессора и скоростью работы RAM.

Кэширование применяется в:

- CPU,
- HDD,
- браузерах,
- веб-серверах ,
- службах DNS (**Domain Name System** — система доменных имён)
- и WINS(*Windows Internet Name Service* - служба сопоставления NetBIOS-имён компьютеров с IP-адресами узлов).

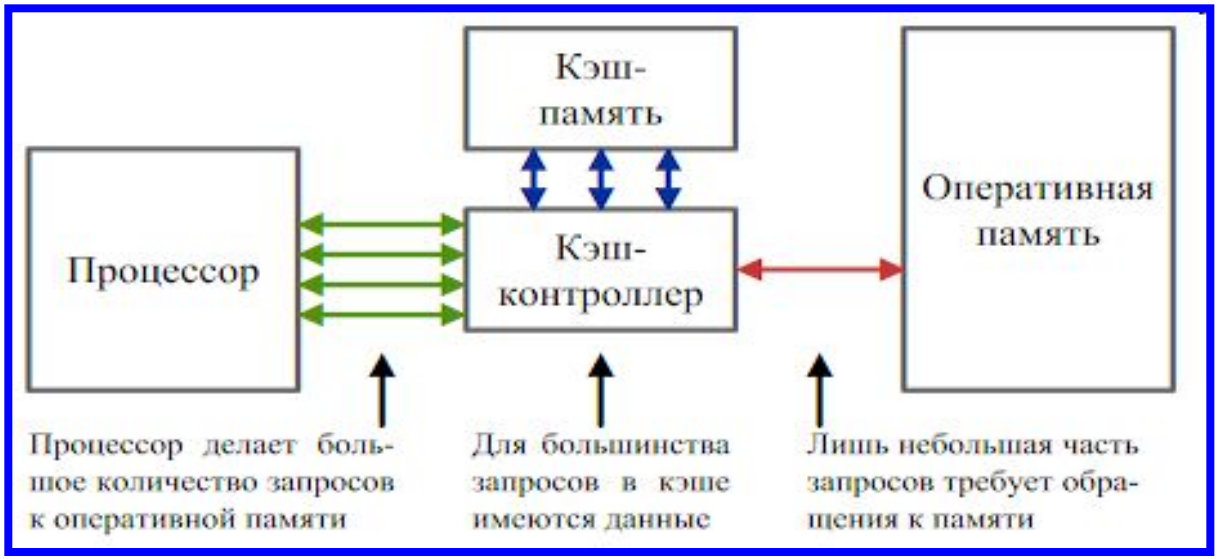






Функционирование

Кэш состоит из собственно кэш-памяти, и кэш-контроллера



Кэш-контроллер управляет кэш-памятью: загружает в неё нужные данные из оперативной памяти - RAM, и возвращает, когда нужно, модифицированные процессором данные в RAM.

Важно понимать, что кэш всегда «полон», т.к. оставлять часть кэш-памяти «пустой» было бы совершенно нерационально.

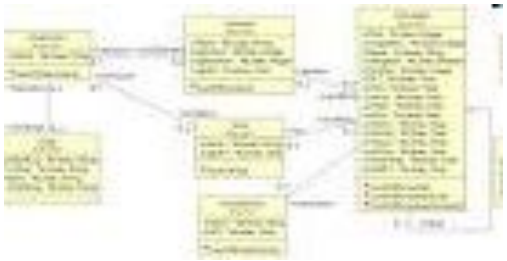
Новые данные попадают в кэш только путём вытеснения (замещения) каких-либо старых данных.

Взаимодействие с прикладными программами

Обычно кэш прозрачен для прикладных программ.

Это означает, что программы работают с памятью, не заботясь о существовании кэша: кэш «перехватывает» запросы к RAM, предоставляя программе требуемые данные.

Однако, современные системы позволяют приложению «подсказать» кэшу правильное поведение, например, при помощи команд предварительной загрузки данных в кэш и записи данных в память, минуя кэш.

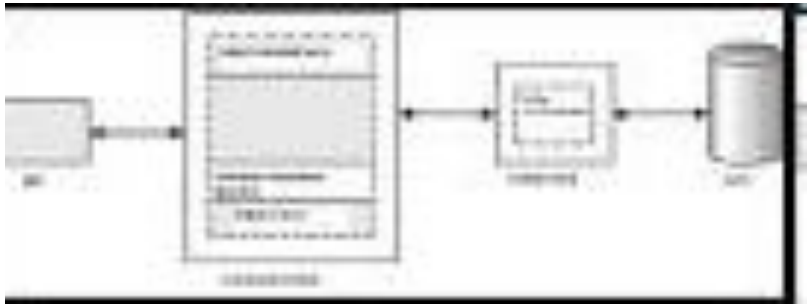


Бывают также вычислительные системы, в которых кэш полностью управляется программой:

- программа может независимо работать с кэш-памятью и RAM

Взаимодействие с прикладными программами

Виртуальную память также можно считать одним из вариантов реализации принципа кэширования данных, при котором RAM выступает в роли кэша по отношению к внешней памяти — HDD.

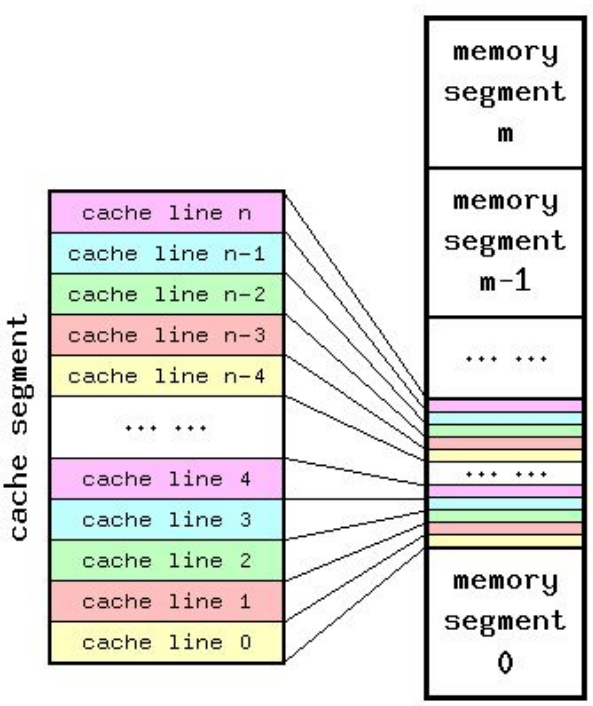


В этом случае кэширование используется не для того, чтобы уменьшить $t_{доступа}$ (время доступа к данным), а для того, чтобы заставить диск частично подменить RAM за счет перемещения временно неиспользуемого кода и данных на диск с целью освобождения места для активных процессов.

В результате наиболее интенсивно используемые данные «оседают» в RAM, остальная же информация хранится в более объемной и менее дорогостоящей внешней памяти (HDD).

Принцип работы кэш-памяти

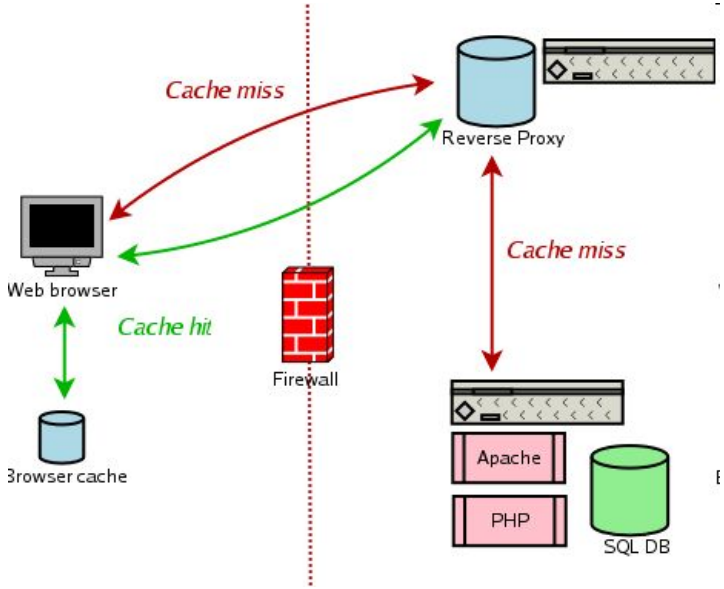
Содержимое кэш-памяти представляет собой совокупность записей обо всех загруженных в нее элементах данных из основной памяти.



Каждая запись об элементе данных включает в себя:

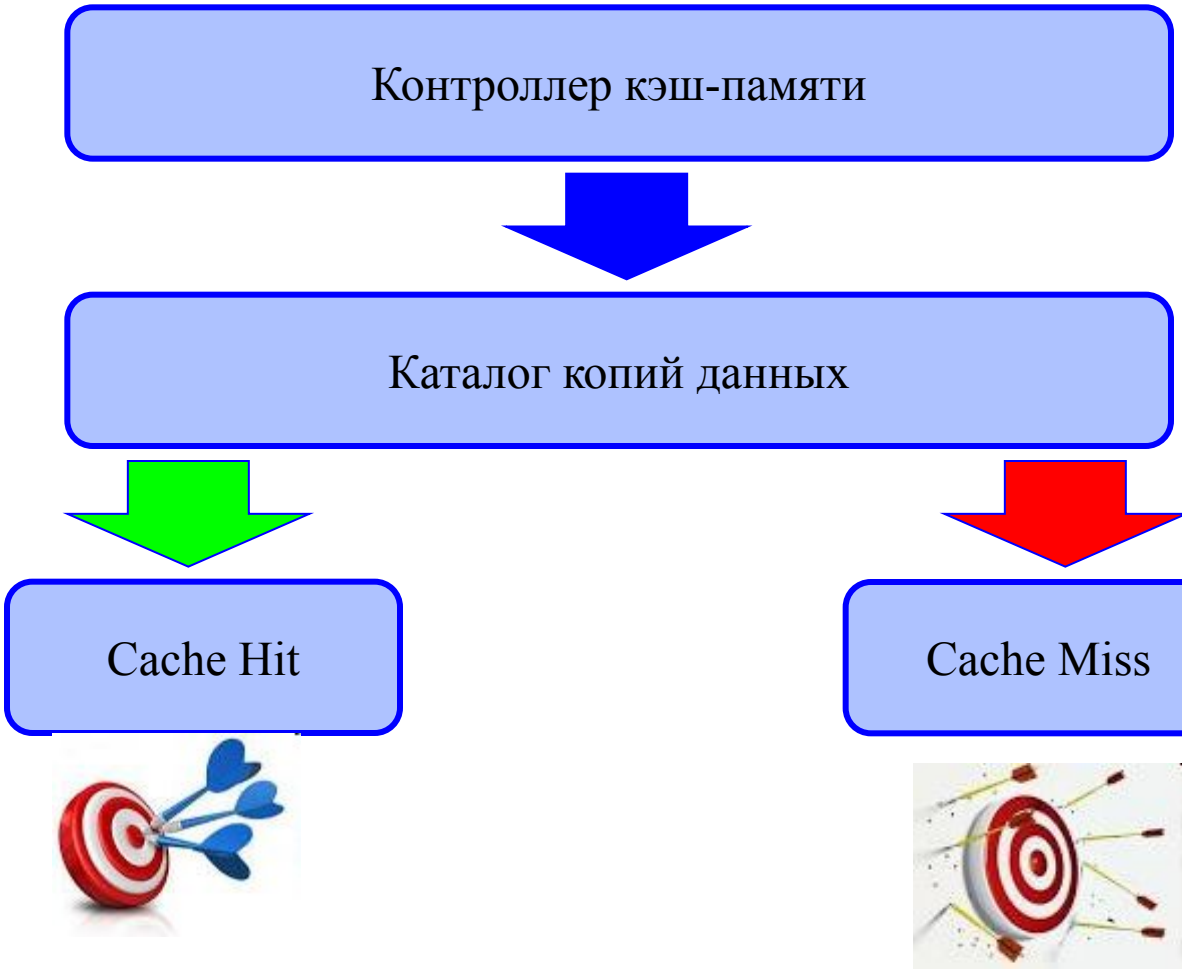
- значение элемента данных;
- адрес, который этот элемент данных имеет в основной памяти;
- дополнительную информацию, которая используется для реализации алгоритма замещения данных в кэше и обычно включает признак модификации и признак действительности данных.

При каждом обращении к памяти контроллер кэш-памяти по каталогу проверяет, есть ли действительная копия затребованных данных в кэше.



Поведение кэш-контроллера

Если она там есть, то это случай кэш-попадания (cache hit) и данные берутся из кэш-памяти.
Если действительной копии там нет, это случай кэш-промаха (cache miss) и данные берутся из основной памяти. В соответствии с алгоритмом кэширования блок данных, считанный из основной памяти, при определенных условиях заместит один из блоков кэша.



От интеллектуальности алгоритма замещения зависит процент попаданий и, следовательно, эффективность кэширования.

Алгоритмы замещения

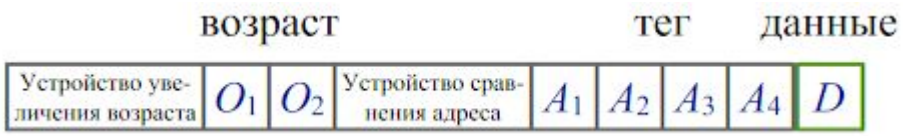
● LRU (Least Recently Used, т.е. "наименее новый из запрашиваемых") — запись идёт в строку, находящаяся в которой информация запрашивалась в последний раз наиболее давно;

● LRR (Least Recently Replaced, т.е. "наименее новый из записанных"), также известный как FIFO (First In, First Out, то есть "первым пришёл, первым ушёл") — вытесняется строка, находящаяся в которой информация характеризуется наиболее старой записью;

● LFU (Least Frequently Used, т.е. "наименее часто запрашиваемый") — для записи выбирается строка, находящаяся в которой информация запрашивалась наименее часто;

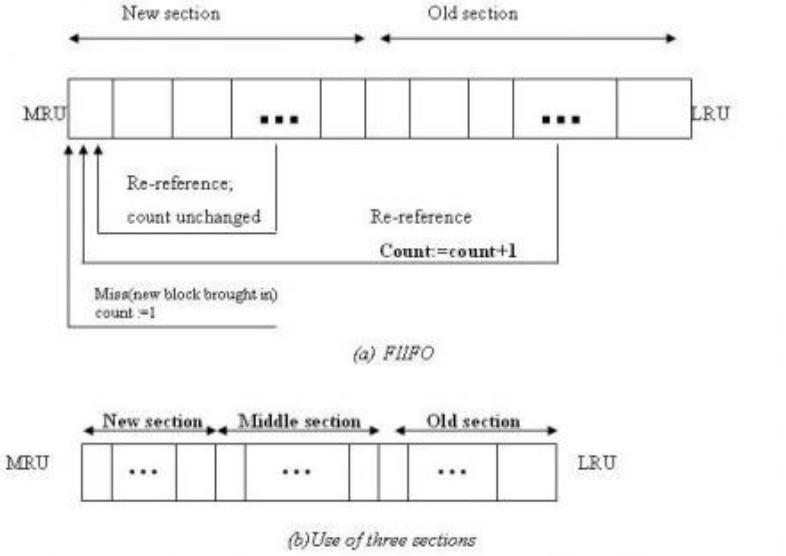
● Random — запись идёт в произвольно выбираемую строку.

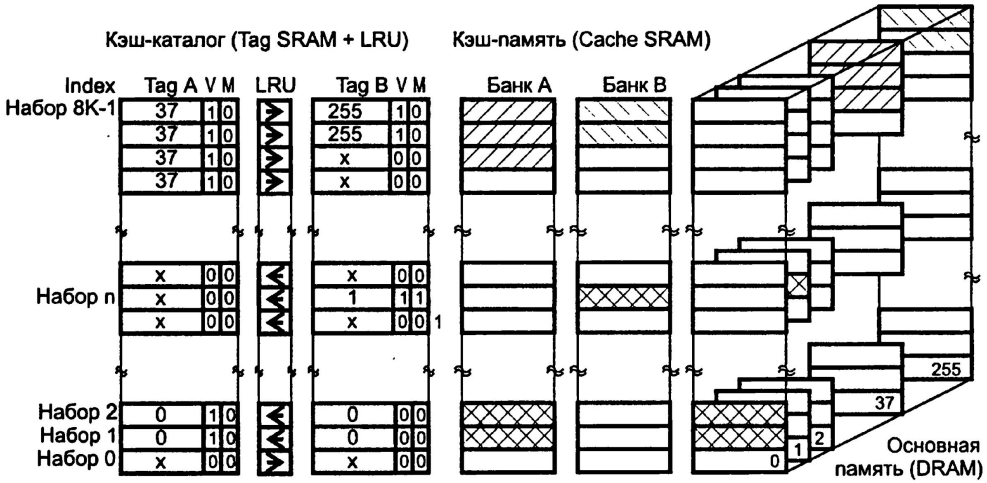
Как можно предположить, более сложные по сравнению с Random алгоритмы являются более эффективными, но используют более сложную обслуживающую логику.



Алгоритмы замещения

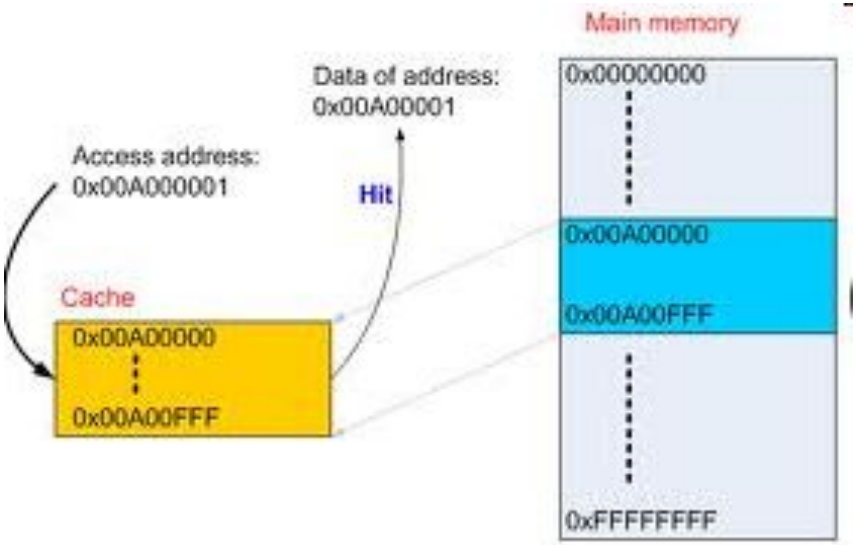
- LFU предполагает, что каждая строка кэш-памяти снабжена соответствующим счётчиком, обновляемым при каждом удачном запросе.
- LRU и LRR используют вместо него счётчик времени в той или иной форме, причём LRU требует его обновления при каждом удачном запросе.
- Random не предполагает каких-либо строчных счётчиков и вполне может быть основан на одном генераторе случайных чисел или чём-то подобном.





Отображение на кэш

На рисунке изображены основная и кэш память.



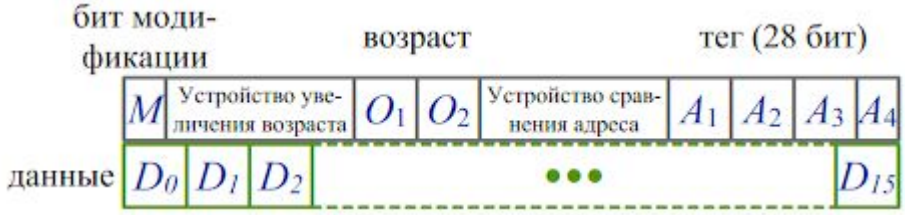
Каждая строка — группа ячеек памяти содержит данные, организованные в кэш-линии.
Размер каждой кэш-линии может различаться в разных процессорах, но для большинства x86-процессоров он составляет 64 байта.

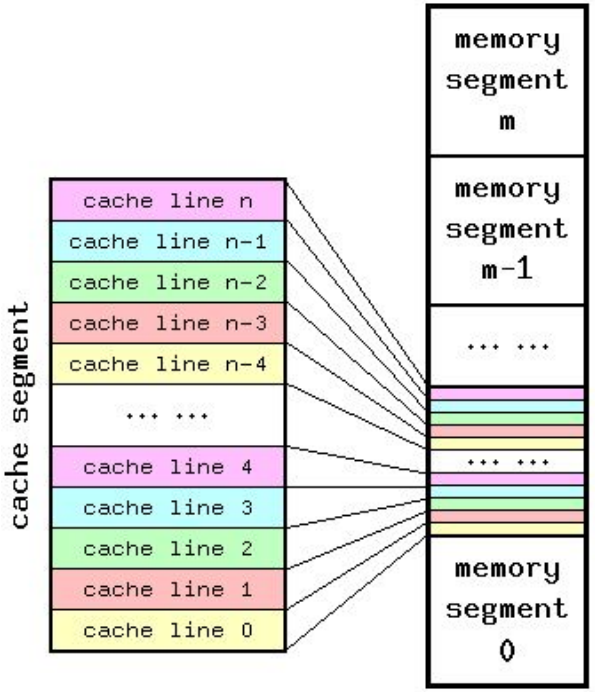
Размер кэш-линии обычно больше размера данных, к которому возможен доступ из одной машинной команды (типичные размеры от 1 до 16 байт).

Каждая группа данных в памяти размером в 1 кэш-линию имеет порядковый номер. Для основной памяти этот номер является адресом памяти с отброшенными младшими битами. В кэше каждой кэш-линии дополнительно ставится в соответствие тег, который является адресом продублированных в этой кэш-линии данных в основной памяти.

Тег

Объяснение кэш-попаданий и кэш-промахов с точки позиции термина тег представляем в следующем формате.





При доступе процессора в память сначала производится проверка, хранит ли кэш запрашиваемые из памяти данные. Для этого производится сравнение адреса запроса со значениями всех тегов кэша, в которых эти данные могут хранится.

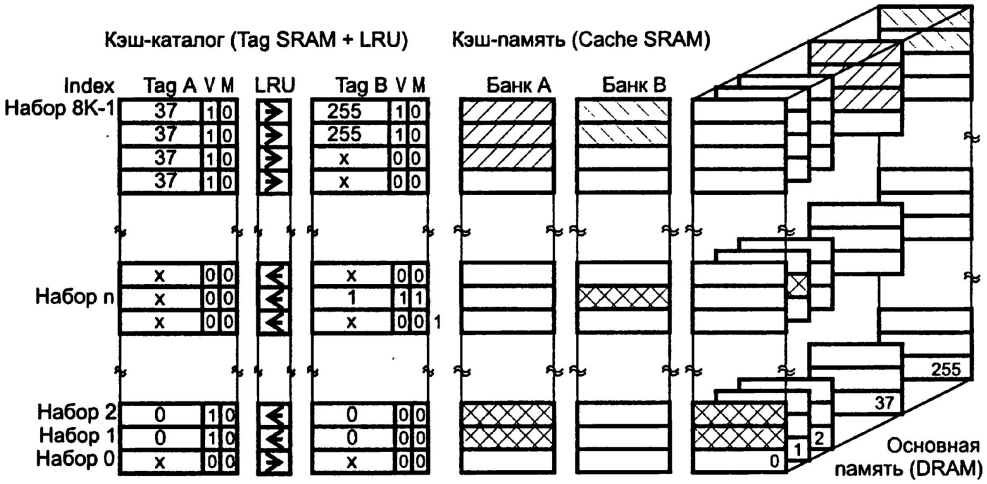
$$\text{HIT RATE} = \frac{Cache\ hits}{Cache\ request}$$

Попадание в кэш позволяет процессору немедленно произвести чтение или запись данных в кэш-линии с совпавшем тегом. Отношение количества попаданий в кэш к общему количеству запросов к памяти называют рейтингом попаданий (hit rate),

оно является мерой эффективности кэша для выбранного алгоритма или программы.

Проблема вытеснения

Основной проблемой алгоритма является предсказание, какая линейка вероятнее всего не потребуется для последующих операций.
Качественные предсказания сложны, и аппаратные кэши используют простые правила, такие как LRU.

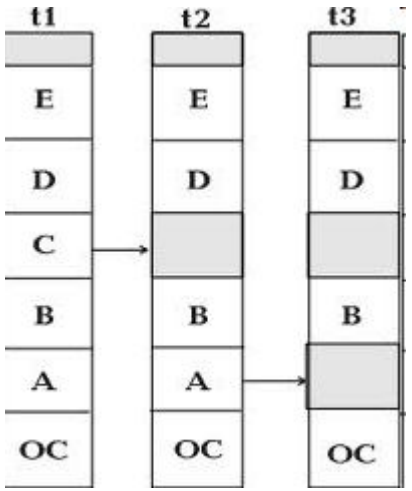


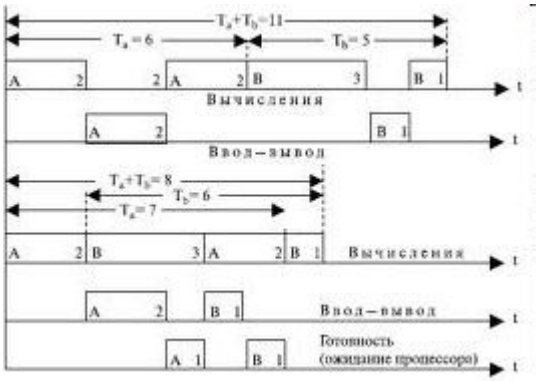
Пометка некоторых областей памяти как некэшируемых (англ. non cacheable) улучшает производительность за счет запрета кэширования редко используемых данных. Промахи для такой памяти не создают копию данных в кэше.

Факторы присутствия

Вероятность обнаружения данных в кэше зависит от разных факторов:

- объем кэша (V_{cache}) - чем больше объём кэша, тем бóльшую часть требуемых программе данных он может в себе содержать, тем реже будут происходить обращения к RAM, и тем выше будет общее быстродействие системы.
- объем кэшируемой памяти(V_{cached}),
- алгоритм замещения данных в кэше,
- особенности выполняемой программы и время ее работы - кэш оказывается эффективным потому, что большинство программ обращаются к памяти не случайным образом, а закономерно. Чем лучше кэш-контроллер может «предсказать» обращения приложения к памяти, тем выше эффективность.
- уровень мультипрограммирования – количества одновременно выполняемых задач на том же объеме памяти;
- и других особенностей вычислительного процесса.





Локальность

Высокое значение вероятности нахождения данных в кэш-памяти объясняется наличием у данных объективных свойств: *временная локальность* и *пространственная локальность*.

Свойства данных

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

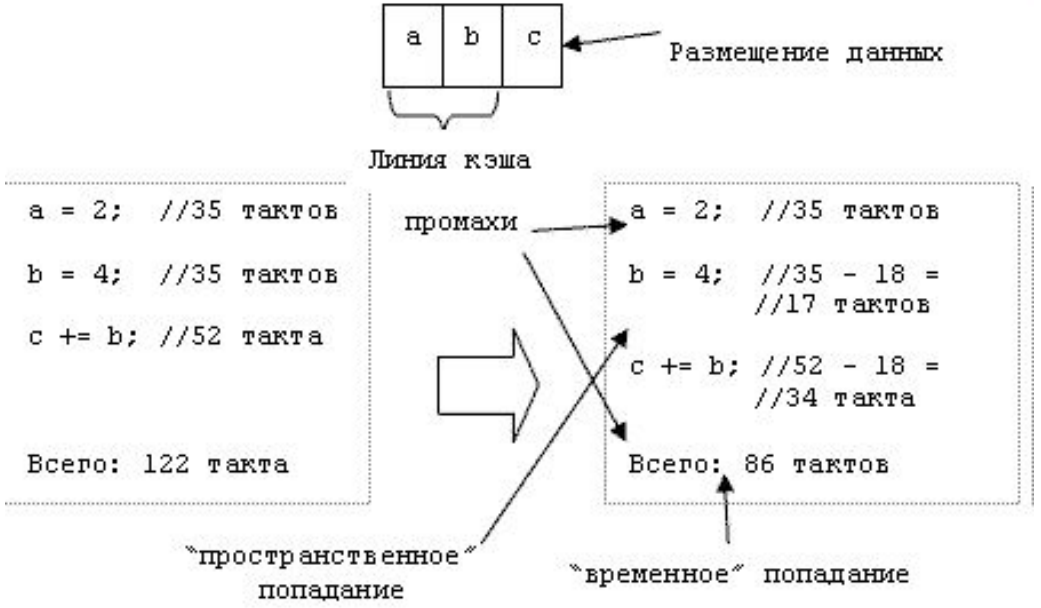
Пространственная локальность
если произошло обращение
по некоторому адресу,
то с высокой степенью
вероятности
в ближайшее время
произойдет обращение
к соседним адресам.

Временная локальность

Именно основываясь на свойстве временной локальности, данные, только что считанные из основной памяти, размещают в запоминающем устройстве быстрого доступа, предполагая, что скоро они опять понадобятся.

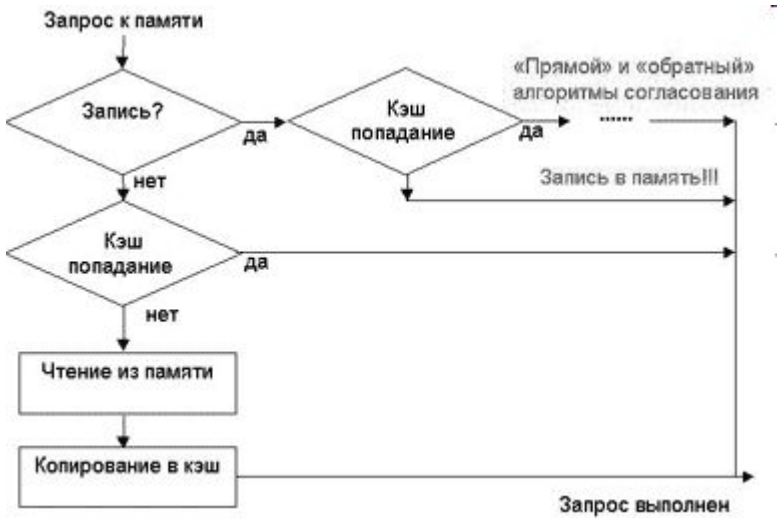
В начале: кэш пуст, =>, каждый запрос к основной памяти выполняется «по полной программе»: просмотр кэша, констатация кэш-промаха, чтение данных из основной памяти, передача результата источнику запроса, копирование данных в кэш.

Затем, по мере заполнения кэша, в полном соответствии со свойством временной локальности возрастает вероятность обращения к данным, которые уже были использованы на предыдущем этапе работы системы, т.е. к данным, которые содержатся в кэше и могут быть считаны значительно быстрее, чем из основной памяти.



Пространственная локальность

Свойство пространственной локальности также используется для увеличения вероятности кэш-попадания: как правило, в кэш-память считывается не один информационный элемент, к которому произошло обращение, а целый блок данных, расположенных в основной памяти в непосредственной близости с данным элементом.

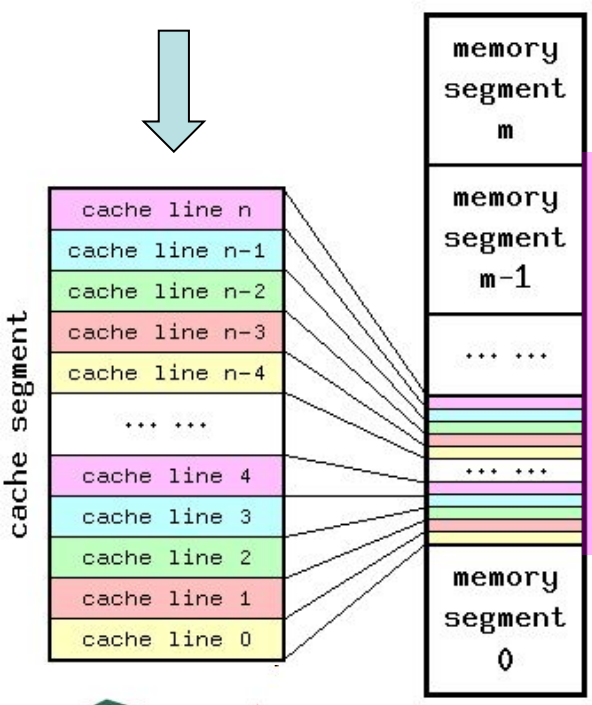


Поскольку при выполнении программы очень высока вероятность, что команды выбираются из памяти последовательно одна за другой из соседних ячеек, то имеет смысл загружать в кэш-память целый фрагмент программы.

Кэш-память: Размер строки, тэга и индекса

Любая кэш-память подразделяется на так называемые строки (lines).

ЗАЧЕМ??



Было бы **крайне нерационально** наделять каждый байт в кэш-памяти адресным полем, указывающим на его местонахождение в RAM

ПОЧЕМУ??

потому что это привело бы к очень тяжёлым последствиям с точки зрения производственных затрат.

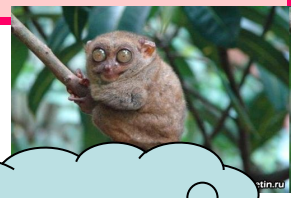
Непонятно?!

- Если рассмотреть 32-битное линейное физическое адресное пространство, позволяющее непосредственно поддерживать до 4Гб оперативной памяти, каждый байт в кэш-памяти должен обслуживаться **4 адресными байтами!**
- Кроме того, такая кэш-память была бы очень плоха с точки зрения производительности

Поэтому куда удобнее адресовать некоторые группы из соседствующих байт, которые и будут **формировать строки кэш-памяти**. На практике широко используются строки по 32 или 64 байта, хотя их размер может достигать даже 1024 байт. Естественно, размер строки должен быть эквивалентен 2 в некоторой целой положительной степени

Кэш-память: Размер строки, тэга и индекса

Понятно! Однако, даже если требование по размеру выполнено, не каждая группа из соседствующих байт может быть кэширована по причине дополнительного ограничения, известного как адресное выравнивание (address alignment)



Другими словами, группа соседствующих байт может помещена в строку кэш-памяти ⇔ если её начальный адрес выровнен по границе, равной размеру строки.
Например, 32-байтная строка может быть заполнена информацией из RAM, находящейся по hex-(dec-) адресам 00-1F (00-31), 20-3F (32-63), 40-5F (64-95) и т. д.

Кроме иных преимуществ, это простое правило позволяет сократить число адресных бит в расчёте на одну строку. Если точнее, то на 5 в вышеприведённом примере, т. е. $\log_2(32) = 5$.

Наконец, строка кэш-памяти может быть либо полностью заполненной действительной информацией, либо полностью пустой, что равносильно быть заполненной недействительной информацией.

Промежуточные варианты **не поддерживаются**. Из этого правила есть **одно исключение**: если у двух строк кэш-памяти имеется одно общее адресное поле, тогда их можно рассматривать как подстроки одной строки, которые могут функционировать относительно **независимо** одна от другой.

Каждое адресное поле состоит из двух основных частей:

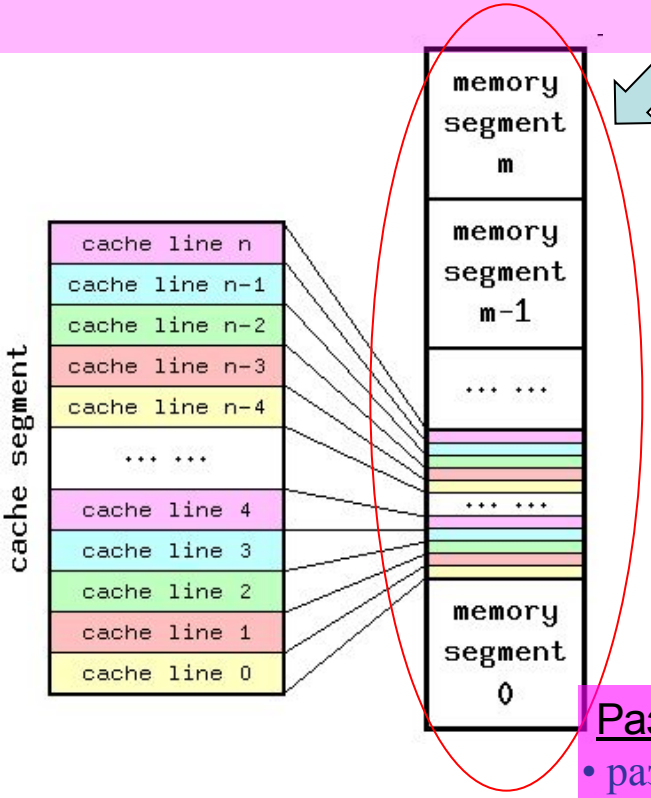
динамическая (tag),
которая содержит старшие биты адреса:
может быть изменена в процессе работы

статическая (index),
которая содержит младшие биты адреса:
значение зафиксировано

Nota Bene: не путать с динамическими и статическими ячейками памяти, это другое.

Кэш-память: Размер строки, тэга и индекса

Для объяснения их функционального назначения необходимо ввести понятие логической сегментации RAM



Допустим,что:

- имеется некоторое физическое пространство RAM, состоящее из M сегментов памяти одинакового размера, каждый из которых равен по размеру сегменту кэш-памяти.
- Каждый сегмент памяти состоит из N строк одинакового размера, каждая из которых равна по размеру строке кэш-памяти.
- Т.о., чтобы получить адрес какой-либо строки памяти, необходимо сначала определить номер её сегмента памяти, затем номер данной строки в этом сегменте и объединить оба номера.
- Для полноты картины осталось лишь подставить тэг вместо номера сегмента и индекс вместо номера строки.

Размер тэга строки кэш-памяти зависит от 3 основных факторов:

- размера кэш-памяти;
- ассоциативности кэш-памяти;
- кэшируемого размера оперативной памяти.

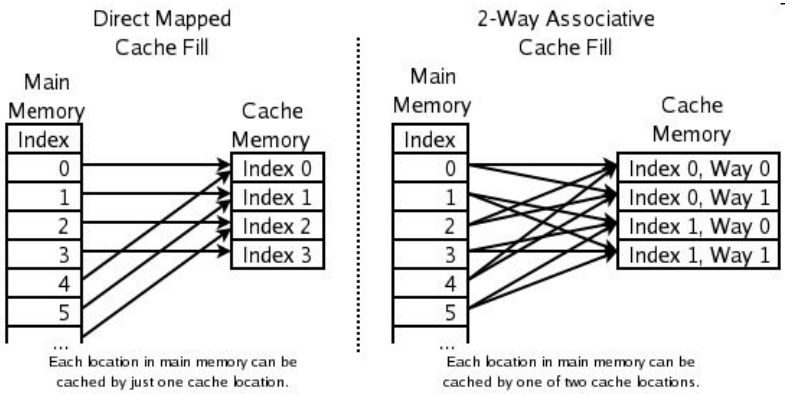
Этот размер рассчитывается по следующей формуле:

$$S_{tag} = \log_2\left(\frac{S_{memory} \times A}{S_{cache}}\right)$$

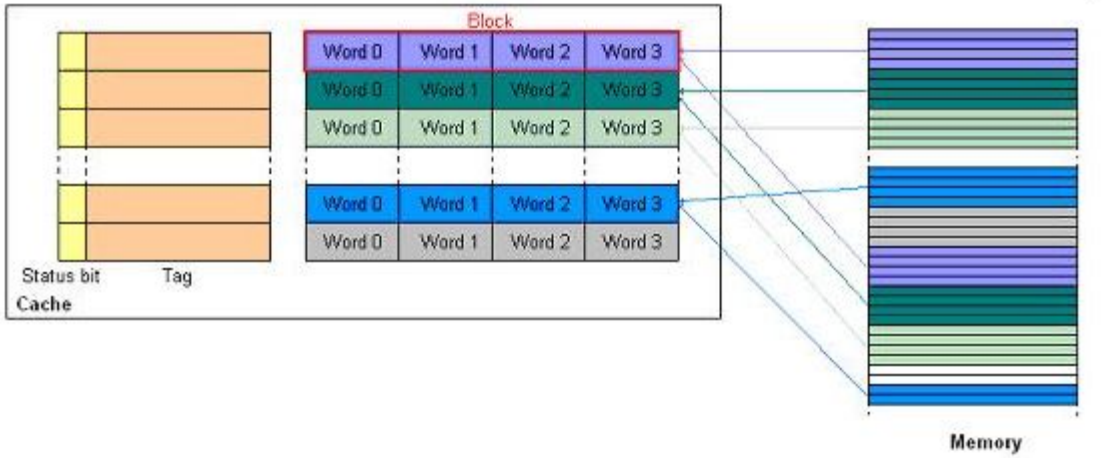
Stag — размер одного тэга кэш-памяти, в битах;
Smemory — максимальный кэшируемый размер RAM, в байтах;
Scache — размер кэш-памяти, в байтах;
A — ассоциативность кэш-памяти, в каналах.

Проблема согласования данных

Наличие в компьютере двух копий данных — в основной памяти и в кэше — порождает проблему согласования данных.



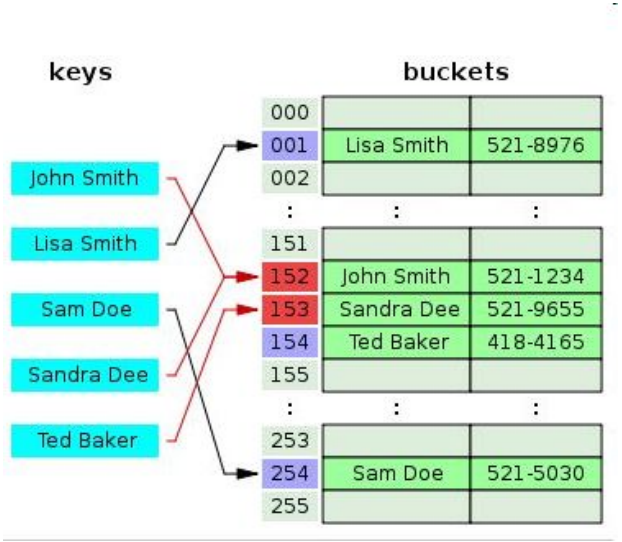
Рассмотрим два подхода к решению этой проблемы:
сквозная запись WT (Write Through) и обратная запись WB (Write Back).

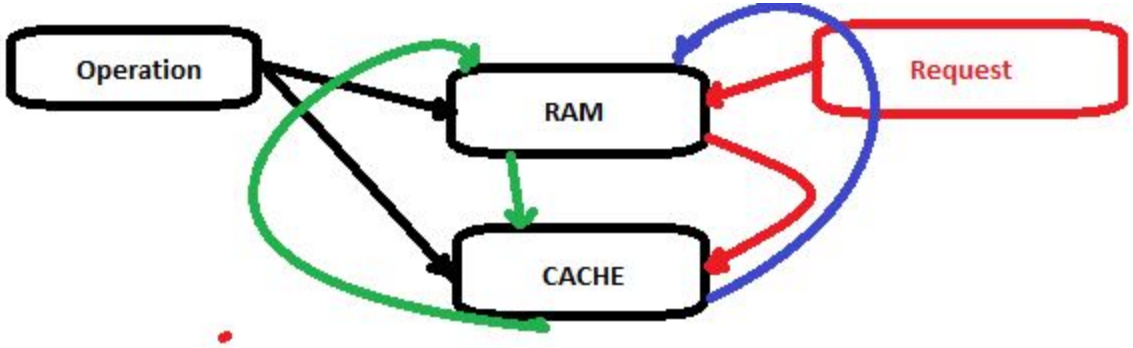


Политика записи в кэш

Политика WT

предусматривает одновременное выполнение каждой операции записи (даже однобайтной), попадающей в кэшированный блок, в строку кэша и в основную память. При этом процессору при каждой операции записи придется выполнять относительно длительную запись в основную память. При каждом запросе к основной памяти, в том числе и при записи, просматривается кэш. Если данные по запрашиваемому адресу отсутствуют, то запись выполняется только в основную память. Если же данные, к которым выполняется обращение, находятся в кэше, то запись выполняется одновременно в кэш и основную память.

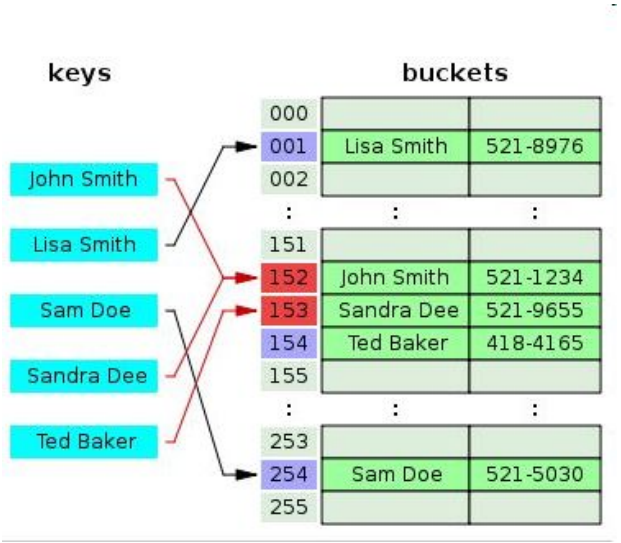




Политика записи в кэш

Политика WT

Алгоритм достаточно прост в реализации и легко обеспечивает целостность данных за счет постоянного совпадения копий данных в кэше и основной памяти.
Для него не нужно хранить признаки присутствия и модифицированности — вполне достаточно только информации тега (при этом считается, что любая строка всегда отражает какой-либо блок, а какой именно — указывает тег).
Но эта простота оборачивается низкой эффективностью записи.
Существуют варианты этого алгоритма с применением отложенной буферизованной записи,при которой данные в основную память переписываются через FIFO-буфер во время свободных тактов шины.



Политика записи в кэш

Политика WB

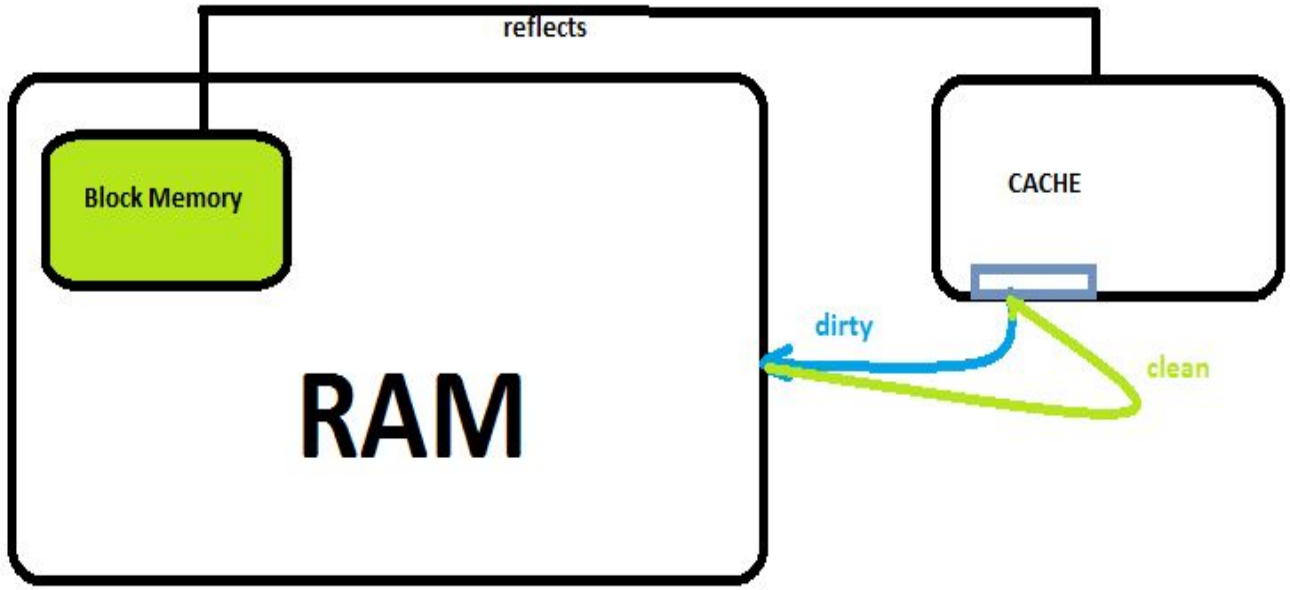
позволяет уменьшить количество операций записи на шине основной памяти.
Если блок памяти, в который должна производиться запись, отображен в кэше,
то физическая запись сначала будет произведена в эту действительную строку кэша,
которая отмечается как грязная (dirty), или модифицированная, т.е. требующая
выгрузки в основную память. Только после этой выгрузки (записи в основную память)
строка станет чистой (clean), и ее можно будет использовать для кэширования других
блоков без потери целостности данных.

В основную память данные переписываются только целой строкой.
Эта выгрузка контроллером может откладываться до наступления крайней необходимости (обращение к кэшированной памяти другим абонентом, замещение в кэше новыми данными) или выполняться в свободное время после модификации всей строки. При возникновении запроса к памяти выполняется просмотр кэша, и если запрашиваемых данных там нет, то запись выполняется только в основную память.
В противном же случае запись производится только в кэш-память, при этом в описателе данных делается специальная отметка (признак модификации), которая указывает на то, что при вытеснении этих данных из кэша необходимо переписать их в основную память, чтобы актуализировать устаревшее содержимое основной памяти.

Политика записи в кэш

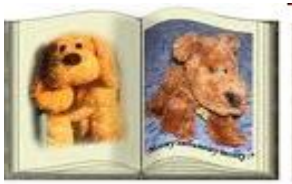
Политика WB

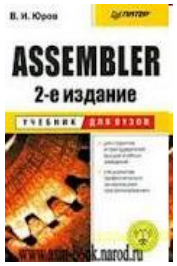
Данный алгоритм сложнее в реализации, но существенно эффективнее, чем WT. Поддержка системной платой кэширования с обратной записью требует обработки дополнительных интерфейсных сигналов для выгрузки модифицированных строк в основную память, если к этой области производится обращение со стороны таких контроллеров шины, как другие процессоры, графические адаптеры, контроллеры дисков, сетевые адаптеры и т. п.

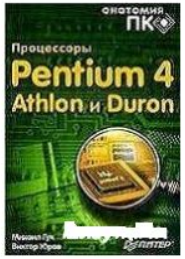


Используемая литература:

- Книга «*Assembler*. Учебник для ВУЗов», автор Юров
- Книга «Процессоры *Pentium4, Athlon* и *Duron*», авторы Михаил Гук, Виктор Юров
- Книга «Модернизация и ремонт ПК»,автор Скотт Мюллер
- Книга «Архитектура ЭВМ», автор Танненбаум
- Книга «Организация ЭВМ», автор Хамахер







ЭВУ
SUPER!!!