КЭШ

Материалы по дисциплине «Микропроцессорные системы»
Специальность «Компьютерные системы и комплексы»
Составитель: Торгашин Р.Г

ГБПОУ ВО "Борисоглебский техникум промышленных и информационных технологий"

2016 год

Общие принципы работы КЭШ – памяти

Что такое КЭШ

Кэш-память (КП), или просто кэш, представляет собой организованную в виде ассоциативного запоминающего устройства (АЗУ) быстродействующую буферную память ограниченного объема, которая располагается между регистрами процессора и относительно медленной основной памятью и хранит наиболее часто используемую информацию совместно с ее признаками (тегами), в качестве которых выступает часть адресного кода.

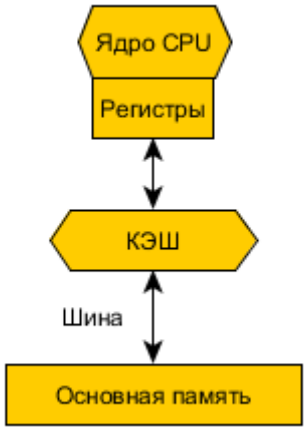


Рисунок 1: Место КЭШа в вычислительной системе

В процессе работы отдельные блоки информации копируются из основной памяти в кэш-память. При обращении процессора за командой или данными сначала проверяется их наличие в КП. Если необходимая информация находится в кэше, она быстро извлекается. Это — **кэш-попадание**. Если необходимая информация в КП отсутствует (**кэш-промах**), то она выбирается из основной памяти, передается в микропроцессор и одновременно заносится в кэш-память.

Повышение быстродействия вычислительной системы достигается в том случае, когда кэш-попадания происходят намного чаще, чем кэш-промахи.

Основные параметры определяющие производительность кэша:

- Размер блока (строки)
- Время попадания (hit time)
- Потери при промахе (miss penalty)
  - Время доступа - access time
  - Время пересылки - transfer time
- Доля промахов (miss rate)
- Размер кэш-памяти

Как же определить наиболее часто используемую информацию?

Неужели сначала кто-то анализирует ход выполнения программы, определяет, какие команды и данные чаще используются, а потом, при следующем запуске программы, эти данные переписываются в кэш-память, и уже тогда программа выполняется эффективно?

Конечно, нет, хотя в некоторых современных микропроцессорах имеются механизмы, которые позволяют по результатам тестового выполнения программы определить, какие ее блоки размещать в кэш-памяти. Но в основном микропроцессор в процессе работы сам отбирает информацию, которая чаще всего используется.

Механизм сохранения информации в кэш-памяти

При включении микропроцессора в работу вся информация в его кэш-памяти недостоверна. При обращении за любой информацией микропроцессор, как отмечалось ранее, сначала проверяет ее наличие в кэш-памяти. Для этого сформированный им физический адрес сравнивается с адресами ячеек памяти, которые были ранее кэшированы из ОЗУ в КП.

При первом запросе информации в кэше, конечно нет. Тогда информация считывается из внешней памяти, передается процессору и записывается в кэш.

Очевидно, что если бы в кэш заносилась только запрошенная информация, то при последующем запросе вновь произошел бы кэш-промах. Кэш попадания начались бы только после того, как в кэше накопился большой фрагмент программы с циклами и повторами.

Чтобы увеличить вероятность кэш-попаданий при последующих обращениях, из оперативной памяти записываются **строки данных**, которые больше запрошенного блока данных.

Рисунок 2: Структура строки кэш памяти

Увеличение процента кэш-попаданий происходит потому, что в большинстве случаев программа обращается к ячейкам памяти расположенным рядом с используемыми. Такую стратегию называют **принципом локальности ссылок.**

Чем длиннее строка тем больше вероятность попадания. Однако слишком большая строка будет долго загружаться из внешней памяти. Поэтому при проектировании КЭШ памяти стараются найти оптимальную длину строки.

После того, как в кэше накопится большой объем информации. Увеличится вероятность попаданий. Но в целом это будет зависеть от выполняемой программы — если в коде использовано большое число циклов — прирост производительности будет выше. Поэтому роль КЭШа в увеличении производительности зависит от типа решаемых системой задач.

Типы КЭШ памяти

С прямым отображением

Кэш памятью с прямым отображением называется кэш в котором у каждой строки внешней памяти есть только одно фиксированное место, где она может находится.

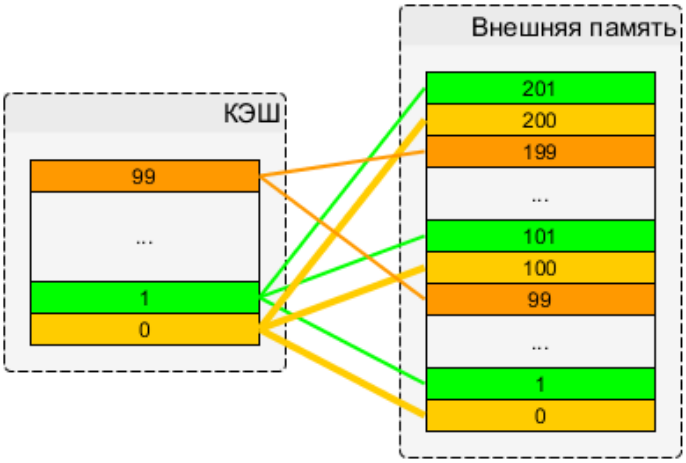


Рисунок 3: КЭШ с прямым отображением

Например, пусть в КЭШе 100 строк а в внешней памяти 1000. Тогда строки памяти с номерами 1, 100, 200… будут отображаться в ячейку 0. Строки с номерами 1, 101, 201.. в ячейку 1 и т. д.

Такая организация обеспечивает быстрый поиск. Однако емкость КЭШа используется не оптимально. Часть может быть не заполнена.

Полностью ассоциативная

Полностью ассоциативная - кэш память в которой каждая строка внешней памяти может располагаться в любом месте кэш памяти.

В такой памяти максимально используется объем. Однако поиск затруднен

Множественно - ассоциативная

Множественно — ассоциативная — кэш память, в которой каждая строка внешней памяти может находится по ограниченному множеству мест в кэше.

Диаграммы иллюстрирующие типы кэш памяти приведены в Приложении А

Стратегии замещения

При замещении информации в кэш памяти на новую используется несколько стратегий замещения:

LRU – замещается строка к которой дольше всего не было обращений;

FIFO – замещается самая давняя по пребыванию в кэш памяти строка;

Random – замещение происходит случайным образом.

Random существенно экономит аппаратные ресурсы и в некоторых случаях обеспечивает более эффективное использование кэш памяти.

Размер кэш-памяти, Кбайт	Организация кэш-памяти					
	2-канальная ассоциативная		4-канальная ассоциативная		8-канальная ассоциативная	
	LRU	Random	LRU	Random	LRU	Random
16	5,18	5,69	4,67	5,29	4,39	4,96
64	1,88	2,01	1,54	1,66	1,39	1,53
256	1,15	1,17	1,13	1,13	1,12	1,12

Рисунок 4: Вероятность кэш-промаха для кэш памяти с различной организацией

Из приведенной на рисунке таблицы видно, что:

С увеличением емкости уменьшается вероятность кэш-промаха. Но даже при 16Кб кэша 95% обращений не доходят до внешней памяти

чем больше ассоциативность — тем больше вероятность кэш попадания

LRU обеспечивает более высокую вероятность попадания по сравнению в Random но незначительно.

Стратегии записи

Для обеспечения соответствия данных в КЭШ и во внешней памяти нужно вносить изменения в те области внешней памяти, для которых данные в кэше изменились.

Существуют две основных стратегии реализации записи изменений:

Сквозная запись (write-through) — данные во внешней памяти обновляются одновременно с обновлением данных в КЭШ. Это снижает производительность. Так как если в строке кэша данные меняются несколько раз подряд — предыдущие записи в внешнюю память становятся бессмысленными. Однако при таком способе содержимое КЭШ и внешней памяти всегда совпадает. А это важно для мультипроцессорных систем с общей ОЗУ.

Обратная запись (write-back)

Строка во внешней памяти модифицируется только при вытеснении соответствующей строки из кэша, или по запросу поддержания согласованности.

Буферизованная сквозная запись (buffered write through) Третий, промежуточный способ, при котором строка накапливаются в специальном буфере. Передача осуществляется при вытеснении строки, заполнении буфера или при запросе поддержания согласованности.

Организация внутреннего кэша микропроцессора

Внутренний кэш 32-разрядного универсального микропроцессора является общим для хранения команд и данных. Кэш-память обычно реализуется в виде ассоциативного ЗУ, в котором для каждой строки сохраняются дополнительные сведения, называемые тегом, или признаком, в качестве которого выступает адресный код или его часть. Когда подается адрес, с ним одновременно сравниваются все теги.

Внутренняя кэш-память в микропроцессоре i486 реализовала сквозную запись. Начиная с Pentium, используется сквозная или обратная запись. Во внешней КП применяется любой способ записи или их комбинация.

Внутренняя кэш-память МП i486 имела емкость 8Кбайт и была организована в виде 4-канальной ассоциативной памяти. Это означает, что данные из какой-либо строки ОЗУ могли храниться в любой из 4 строк кэш-памяти.

КЭШ-память состоит из следующих блоков:

Блок данных - предназначен для хранения данных; Также в нем хранится индекс строки. Строки с одинаковыми индексами называются множествами.

Блок тегов - в котором хранятся биты, задающие адрес кэшированного блока во внешней памяти. Тегом строки обычно считается адрес в оперативной памяти первого содержащегося в ней байта. Также в блоке содержится бит значимости, определяющий действительность строки (если бит значимости 0, данная строка считается недействительной и не вызовет кэш-попадания).

В **LRU** содержатся биты, управляющие механизмом LRU.

Заметим, что возможна реализация, в которой биты достоверности относятся к тому же блоку что и биты LRU

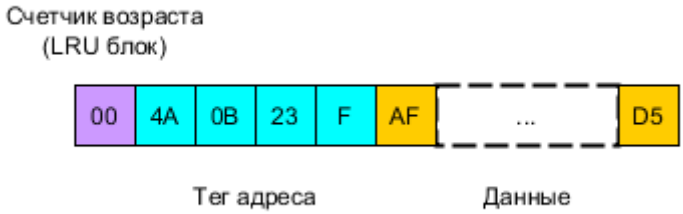


Рисунок 5: Пример кэш-строки

Поиск информации в кэш-памяти производится по алгоритму:

1. Физический адрес по которому происходит обращение разбивается на тег, индекс и номер байта. Поле индекса определяет множество строк в блоке данных;
2. В выбранном множестве сравниваются теги запрошенной информации и теги строк множества. Сравнение выполняется только для достоверных строк.
3. Если теги и индексы в запросе совпали с одной из строк то произошло кэш-попадание
   a. Считывается найденная строка
4. Если попадания не произошло то произошел кэш-промах
   a. Выполняется запрос данных из внешней памяти

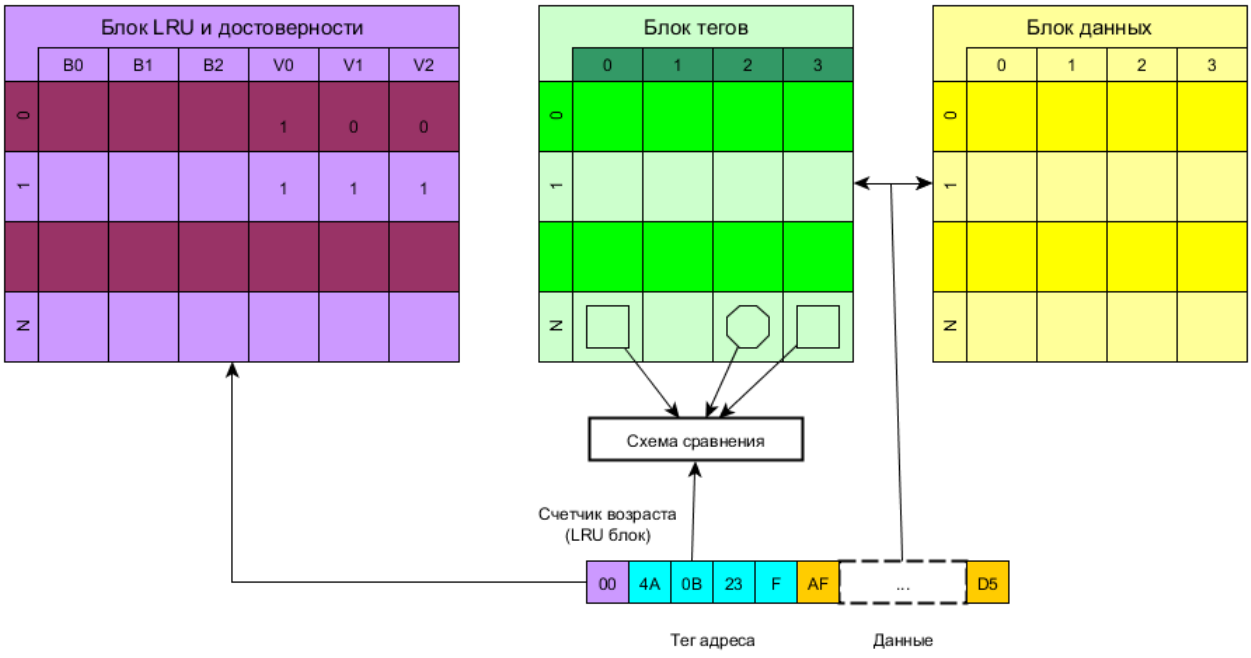


Рисунок 6: Структура внутреннего кэша процессора Intel 80486

Обеспечение согласованности кэш-памяти микропроцессоров в мультипроцессорных системах.

В настоящее время распространена ситуация, когда несколько микропроцессоров используют одну оперативную память. Например это многоядерные процессоры, в которых каждое ядро в данном контексте — микропроцессор.

В этом случае могут возникнуть проблемы, связанные с рассогласованием содержимого кэш-памяти одного из микропроцессоров с общей оперативной памятью нескольких МП.

Предположим, что МП А считал некоторую строку данных из ОЗУ в свою внутреннюю КП и изменил данные в этой строке в процессе работы. Как было сказано ранее существует два основных механизма обновления оперативной памяти:

1) сквозная запись, которая подразумевает, что, как только изменилась информация во внутренней кэш-памяти, эта же информация копируется в то же место оперативной памяти;

2) обратная запись, при которой микропроцессор после изменения информации во внутреннем кэше отражает это изменение в оперативной памяти не сразу, а лишь в тот момент, когда происходит вытеснение данной строки из кэш-памяти в оперативную.

Иными словами, существуют определенные моменты времени, когда информация, предположим, по адресу 2000 имеет разные значения в разных запоминающих устройствах микропроцессорной системы: МП А ее обновил, а в оперативной памяти осталось старое значение. Если в этот момент другой микропроцессор (МП В), использующий ту же оперативную память, обратится по адресу 2000 в ОЗУ, то он прочитает оттуда старую информацию, которая к этому времени уже не актуальна.

Для обеспечения согласованности (*когерентности*) памяти в мультипроцессорных системах используются аппаратные механизмы, позволяющие решить эту проблему. Такие механизмы называются **протоколами когерентности кэш-памяти**. Эти протоколы призваны гарантировать, что любое считывание элемента данных возвращает последнее по времени записанное в него значение.

Существует два класса протоколов когерентности:

• **протоколы на основе справочника (directory based)**: информация о состоянии блока физической памяти содержится только в одном месте, называемом справочником (физически справочник может быть распределен по узлам системы);

• **протоколы наблюдения (snooping)**: каждый кэш, который содержит копию данных некоторого блока физической памяти, имеет также соответствующую копию служебной информации о его состоянии.

Централизованная система записей отсутствует. Обычно кэши расположены на общей шине, и контроллеры всех кэшей наблюдают за шиной (просматривают ее), чтобы определять, какие обращения по адресам в пределах этого блока происходят со стороны других микропроцессоров.

В мультипроцессорных системах с общей памятью наибольшей популярностью пользуются протоколы наблюдения, поскольку для опроса состояния кэшей они могут использовать уже существующее физическое соединение — шину памяти.

Для поддержания когерентности применяется два основных метода. Один метод заключается в том, чтобы гарантировать, что процессор должен получить исключительные права доступа к элементу данных перед выполнением записи в этот элемент данных. Этот тип протоколов называется протоколом **записи с аннулированием (write ivalidate protocol)**, поскольку при выполнении записи он аннулирует другие копии. Это наиболее часто используемый протокол как в схемах на основе справочников, так и в схемах наблюдения. Исключительное право доступа гарантирует, что во время выполнения записи не существует

никаких других копий элемента данных, в которые можно писать или из которых можно читать: все другие кэшированные копии элемента данных аннулированы.

Альтернативой протоколу записи с аннулированием является обновление всех копий элемента данных в случае записи в этот элемент данных. Этот тип протокола называется протоколом **записи с обновлением (write update protocol)**, или протоколом **записи с трансляцией (write broadcast protocol)**.

Эти две схемы во многом похожи на схемы работы кэш-памяти со сквозной и обратной записью. Ключевым моментом реализации в многопроцессорных системах с небольшим числом процессоров как схемы записи с аннулированием, так и схемы записи с обновлением данных является использование механизма шины для выполнения этих операций. Для выполнения операции обновления или аннулирования процессор просто захватывает шину и транслирует по ней адрес, по которому должно производиться обновление или аннулирование данных. Все процессоры непрерывно наблюдают за шиной, контролируя появляющиеся на ней адреса. Процессоры проверяют, не находится ли в их кэш-памяти адрес, появившийся на шине. Если это обнаруживается, то соответствующие данные в кэше либо аннулируются, либо обновляются в зависимости от используемого протокола.

Один из наиболее распространенных протоколов, обеспечивающих соглассогласованную работу кэш-памяти нескольких микропроцессоров и основной памяти в мультимикропроцессорных системах - MESI[1].

Многоуровневая организация кэша

Все современные процессоры имеют как минимум двухуровневую структуру кэш-памяти, а большинство процессоров трехуровневую кэш-память. При этом различают кэш первого уровня (обозначается L1), кэш второго уровня (L2) и кэш третьего уровня (L3). Причем кэши всех уровней размещены на кристалле процессора.

Казалось бы, зачем нужно делать так много кэшей? Не проще ли создать один большой кэш? Оказывается, не проще. Проблема заключается в том, что чем больше размер кэша, тем ниже его скорость. То есть можно сделать один большой, но медленный кэш, а можно — несколько маленьких, но быстрых кэшей, и второй вариант оказывается более предпочтительным.

Кроме того, кэши разных уровней в процессоре выполняют различные задачи. Так, самый быстрый и маленький кэш первого уровня L1 всегда делится на кэш данных (L1D) и кэш команд или инструкций (L1I). Это так называемая гарвардская архитектура процессора (*да, современные реализации Фон-Немановской архитектуры содержит элементы Гарвардской*). Кэш L1 всегда принадлежит только конкретному ядру многоядерного процессора.

Кэш второго уровня L2 является уже унифицированным (содержит и данные и команды). Кэш L2 всегда больше, чем кэш L1, но медленнее его. В случае многоядерных процессоров кэш L2 часто принадлежит конкретному ядру процессора.

А вот кэш L3 является самым большим и медленным и разделяется между всеми ядрами процессора.

Понятно, что в случае, когда в процессоре имеется многоуровневая система кэш -памяти, необходимо организовать взаимодействие между кэшами разных уровней.

Для начала рассмотрим двухуровневую систему кэша. Такая кэш-память строится на базе одной из двух архитектур: включающей, которую также называют инклюзивной (inclusive), и исключающей, именуемой эксклюзивной (exclusive). То есть кэш L2 всегда построен либо по включающей, либо по исключающей архитектуре по отношению к кэшу L1

1 Работа протокола рассмотрена в книге «Микропроцессорные системы: Учебник / В.В. Гуров» Глава 4.4

(отметим, что при наличии кэша L3 кэши L2 и L1 могут быть и не включающими, и не исключающими по отношению друг к другу).

Кэш L2, построенный по включающей архитектуре, всегда дублирует содержимое кэша L1, а потому эффективная емкость кэш - амяти равна емкости кэша L2.

Кэш L2, построенный по исключающей архитектуре, никогда не дублирует содержимое кэша L1, а потому эффективная емкость кэш-памяти равна суммарной емкости кэшей L1 и L2.

Пусть кэш имеет включающую архитектуру. Рассмотрим, каким образом происходит запись данных из оперативной памяти в такой кэш. Если в такой системе кэш - памяти при полностью заполненном кэше L2 процессор пытается загрузить еще одну кэш-строку, то произойдет следующее. Обнаружив, что все кэш-строки заняты, кэш L2 избавляется от наименее ценной из них, стремясь при этом найти слово, которое еще не было модифицирована, поскольку в противном случае его еще придется выгружать в оперативную память.

Затем кэш L2 передает полученные из памяти данные кэшу L1. Если кэш первого уровня также заполнен, ему приходится избавляться от одной из кэш-строк по сценарию, описанному выше.

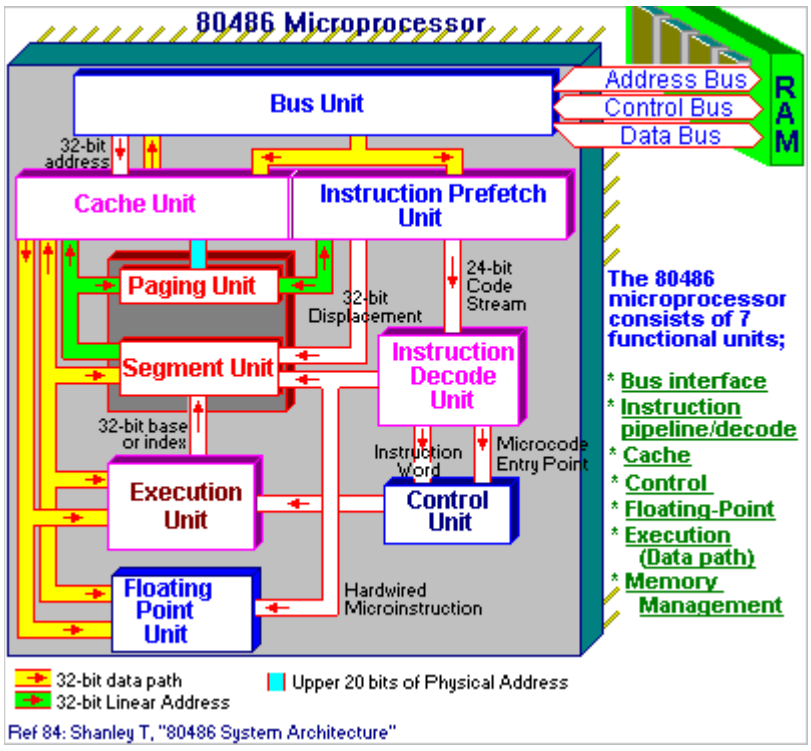
Таким образом, загруженная порция данных присутствует и в кэше L1, и в кэше L2.

Отметим, что процессоры Intel Pentium II и Pentium III имели двухуровневый кэш, построенный по включающей архитектуре.

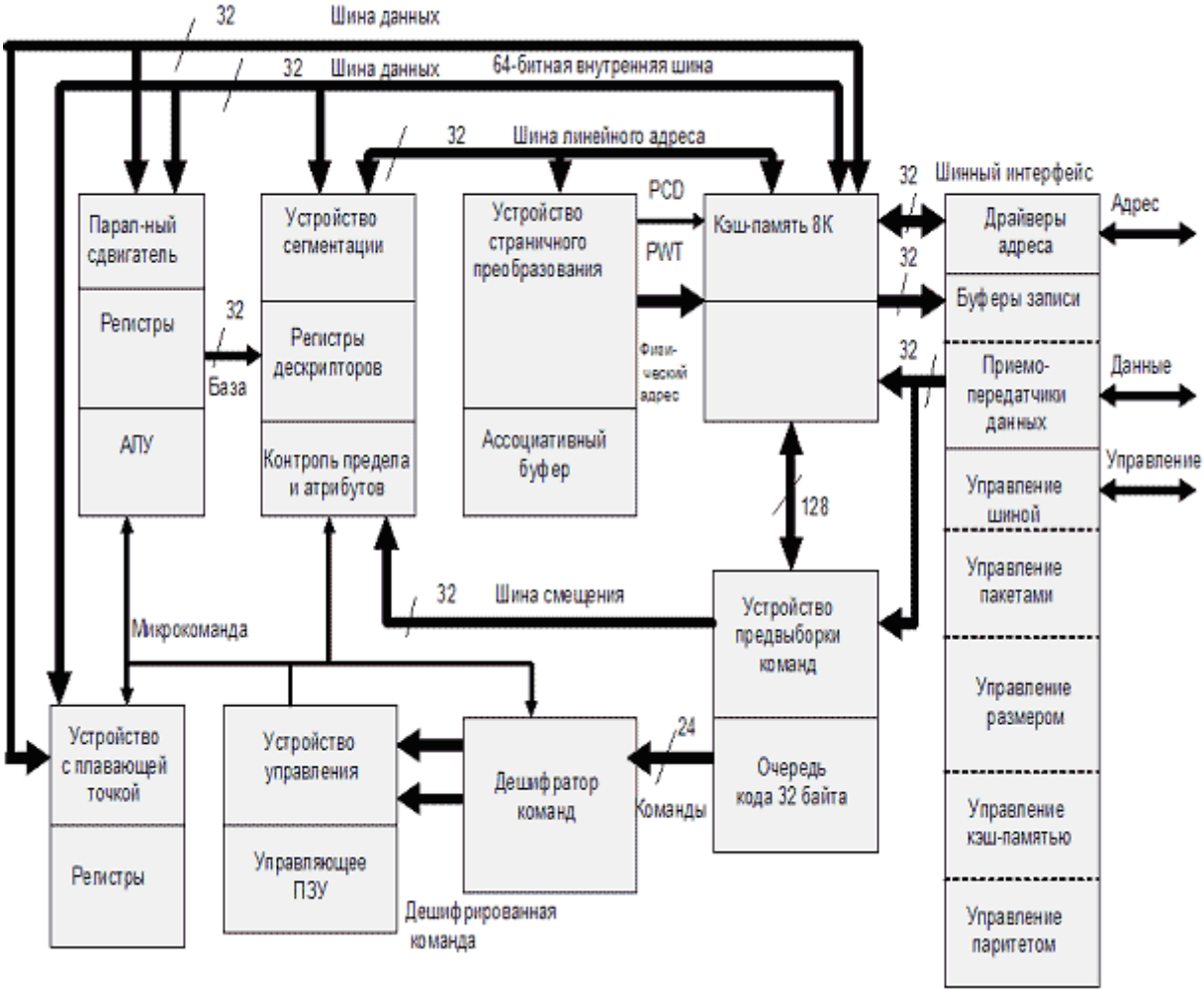
В случае кэша, построенного по исключающей архитектуре, кэш L1 никогда не уничтожает кэш-строки при нехватке места. Даже если кэш-строки не были модифицированы, они вытесняются в кэш L2 на то место, где находилась только что переданная кэшу L1 кэш-строка. То есть кэши L1 и L2 как бы обмениваются друг с другом своими кэш-строками, благодаря чему кэш-память используется весьма эффективно.[2]

КЭШ в архитектуре реальных процессоров

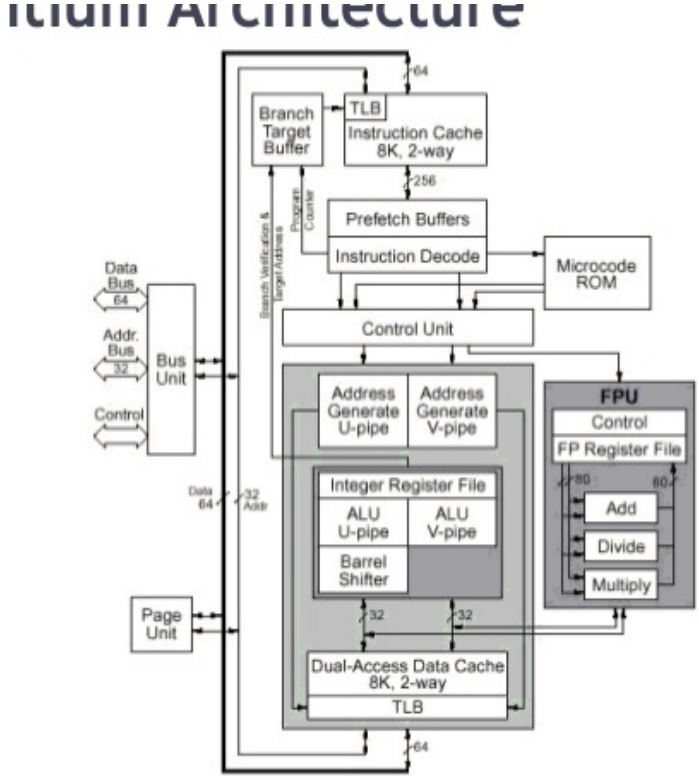
Процессор 80486

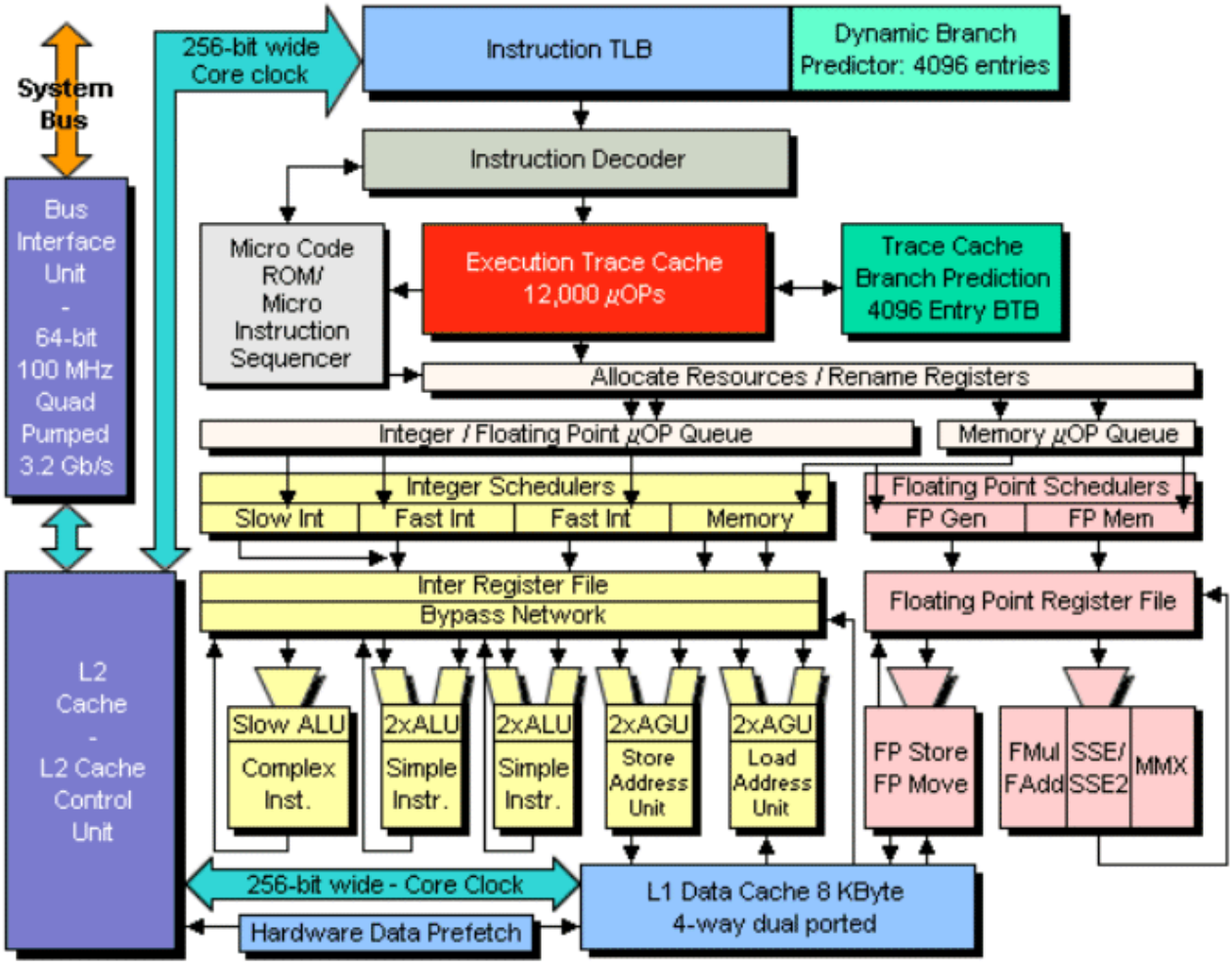


2 Сергей Пахомов Что такое кэш процессора, и как он работает - http://compress.ru/article.aspx?id=23541

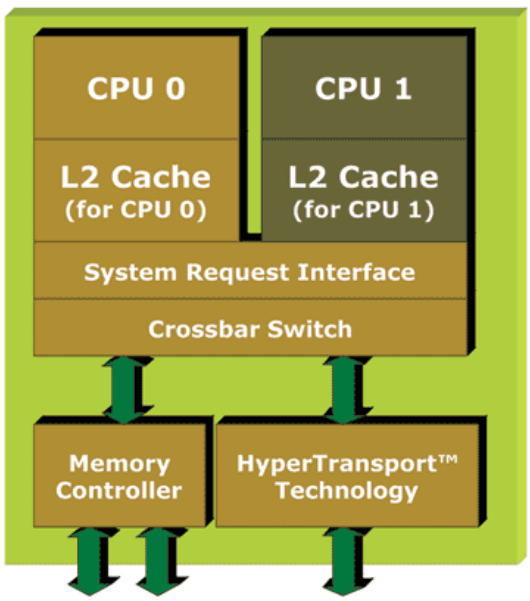


Процессор 80586(Pentium)

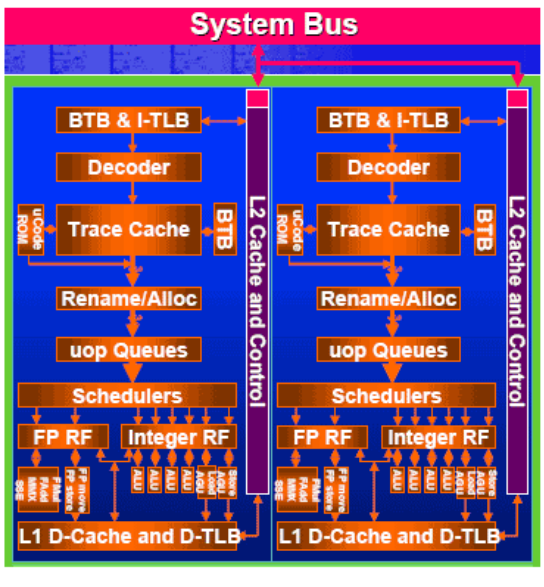




Процессоры Athlon64x2 и PentiumD

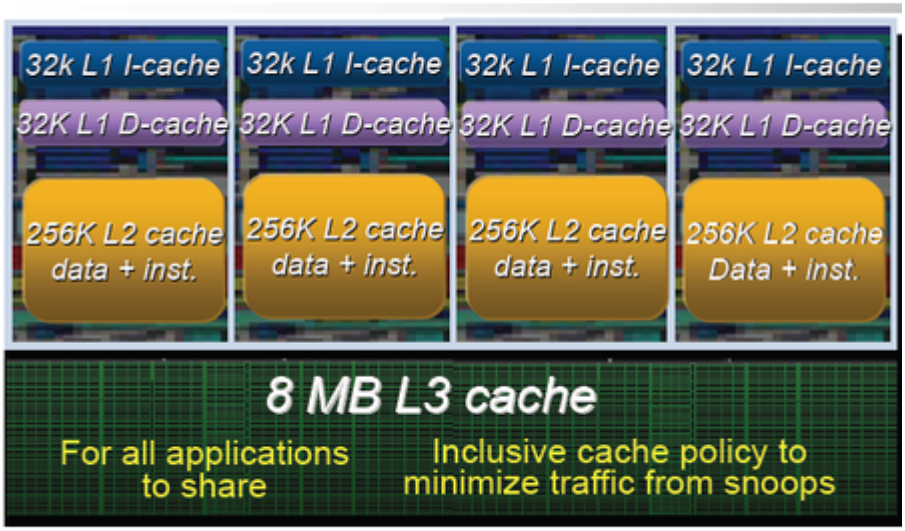


AMD Athlon 64 X2

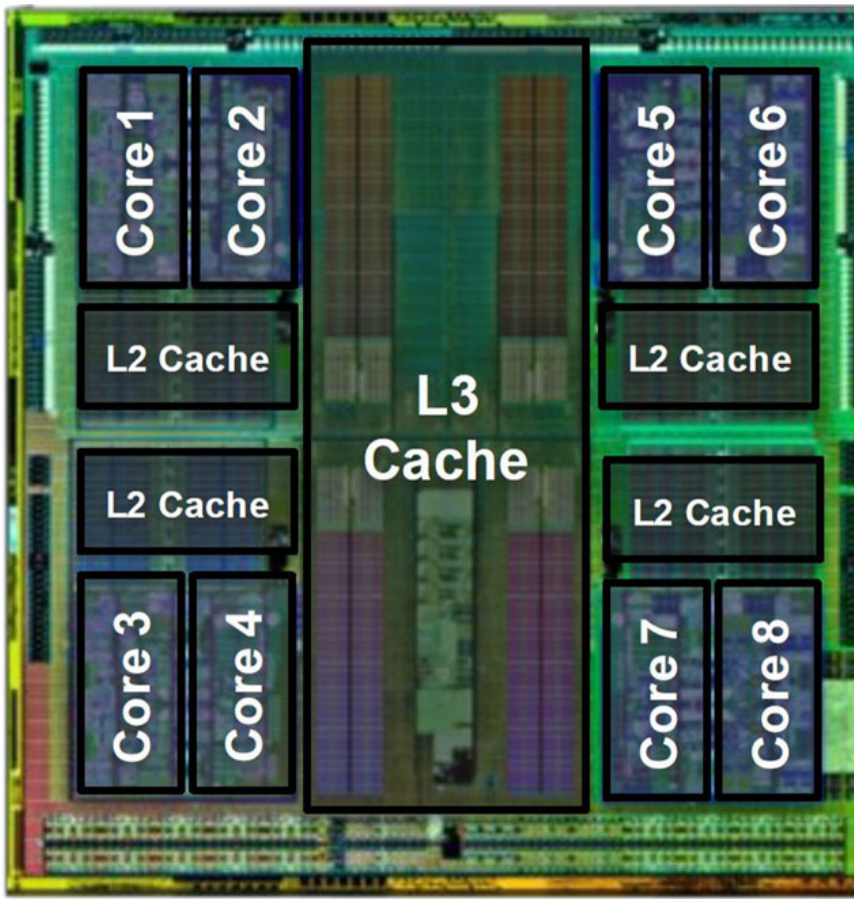


Intel Pentium D 9xx

Процессор Core i7(Nehalem)



Процессор AMD FX(Bulldozer)



Источники

Микропроцессорные системы: Учебник / В.В. Гуров. - М.: НИЦ ИНФРА-М, 2016. - 336 с.: 60x90 1/16. - (Высшее образование: Бакалавриат) (Переплёт) ISBN 978-5-16-009950-7

Сергей Пахомов Что такое кэш процессора, и как он работает - http://compress.ru/article.aspx?id=23541

Wikipedia Кэш процессора - https://ru.wikipedia.org/wiki/Кэш_процессора

Приложение А

Типы кэш памяти

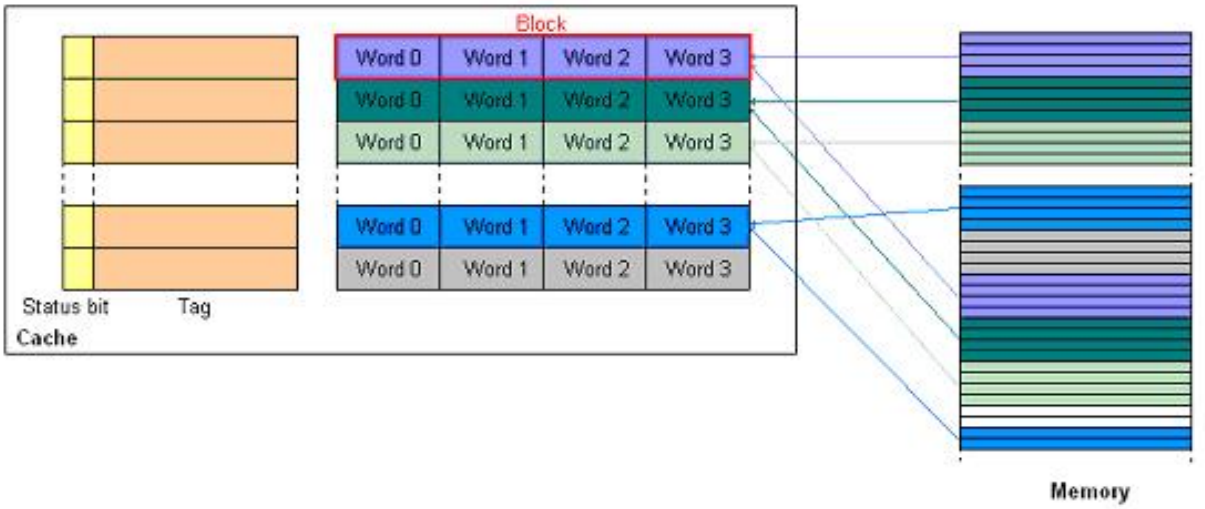


Рисунок 7: Кэш прямого отображения

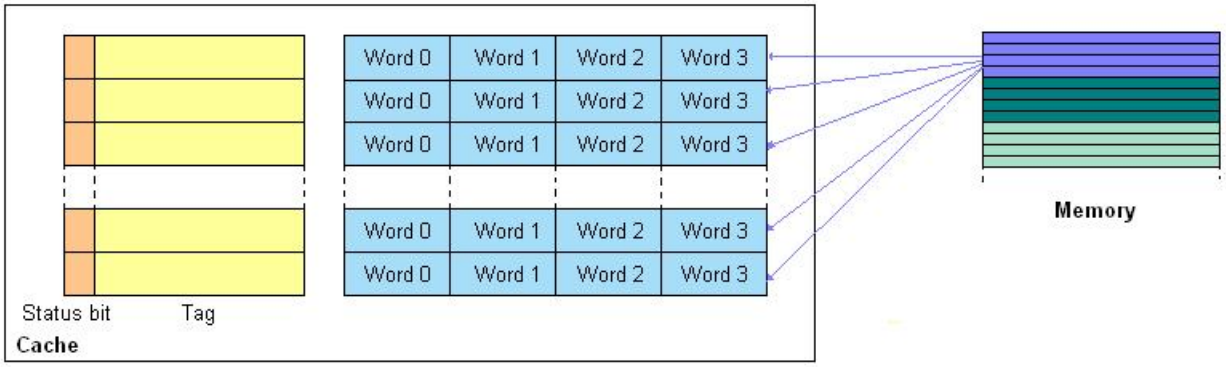


Рисунок 8: Полностью ассоциативный кэш

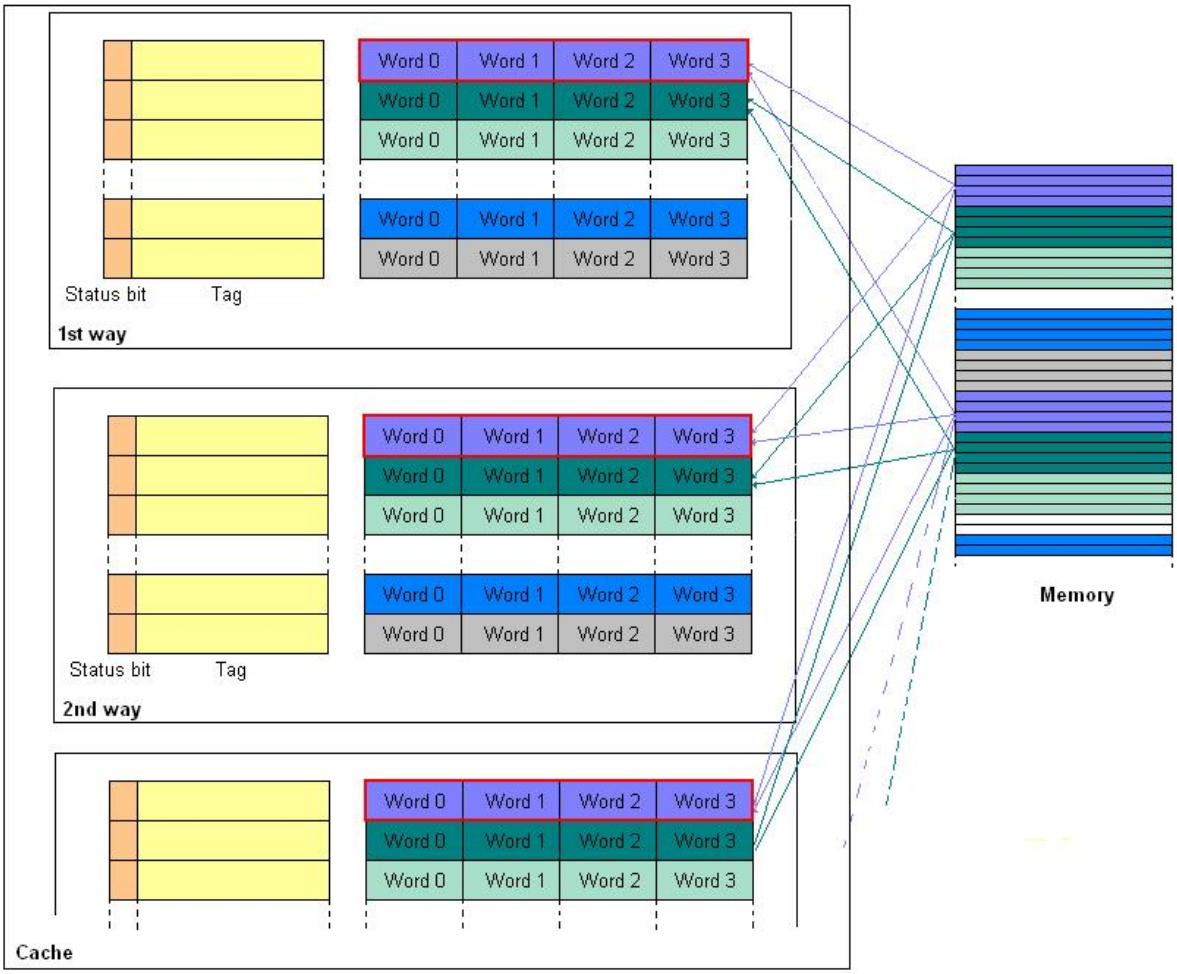


Рисунок 9: n-канальный ассоциативный кэш