ВВЕДЕНИЕ

Главная проблема, связанная с памятью, состоит в том, что скорость процессоров растёт быстрее, чем скорость памяти. В настоящее время оперативная память типа DRAM (Dynamic Random Access Memory, динамическая память с произвольным доступом) медленнее процессора от 10 до 100 раз. Увеличивающийся разрыв в скорости между процессором и оперативной памятью требует все более и более изощренных подсистем памяти, чтобы попытаться приблизить скорость работы памяти к скорости процессора.

Процессор работает с памятью через интерфейс памяти. (рис. 8.1.)

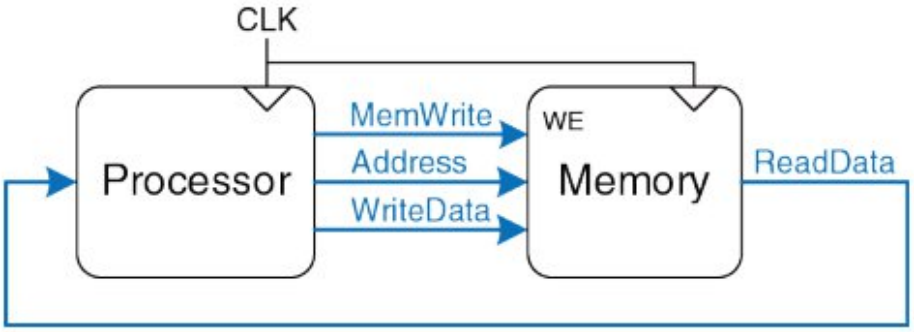


Рис. 8.1 Интерфейс к памяти

Компьютерная память в основном построена на базе динамической (DRAM) и статической (SRAM) памяти. В идеале память должна быть быстрой, большой и дешёвой. Однако на практике любой тип памяти имеет только два из этих свойств; память либо медленная, либо дорогая, либо маленького объема.

Оперативная память компьютера обычно строится на микросхемах динамической памяти (DRAM). На рисунке показан график увеличения скорости работы оперативной памяти и процессоров с 1980 года по настоящее время. (рис. 8.2) Как видно из графика время доступа к данным в DRAM на порядок или два медленнее, чем длительность такта процессора (десятки наносекунд против долей наносекунды).

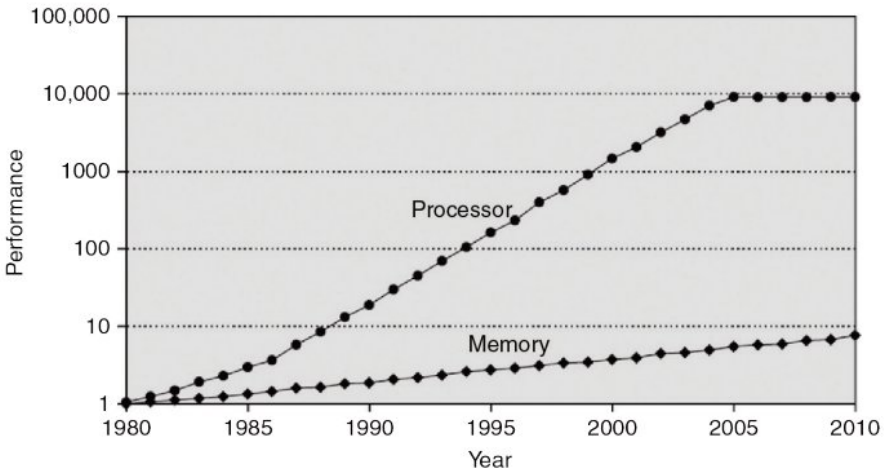


Рис. 8.2 Разрыв в производительности процессоров и памяти.
График из книги Hennessy and Patterson, Computer Architecture:
A Quantitative Approach, 5th ed., Morgan Kaufmann, 2012, с разрешения авторов

Чтобы справиться с этой проблемой, компьютеры хранят наиболее часто используемые команды и данные в быстрой, но небольшой по объему памяти, называемой кэш-памятью или просто кэшем (англ.: cache). Кэш-память обычно сделана с использованием статической памяти (SRAM).

Процессор сначала обращается к кэш-памяти (первый уровень иерархии). Если происходит кэш-промах, то он обращается к оперативной памяти (второй уровень иерархии).

Третий уровень в иерархии памяти – жёсткий диск. Он используется для хранения данных, которые не помещаются в оперативной памяти. Жесткий диск обеспечивает иллюзию наличия бóльшего объема памяти, чем реально доступно в оперативной памяти. Это называется виртуальной памятью. Типичная иерархия памяти показана на схеме. (рис.8.3)

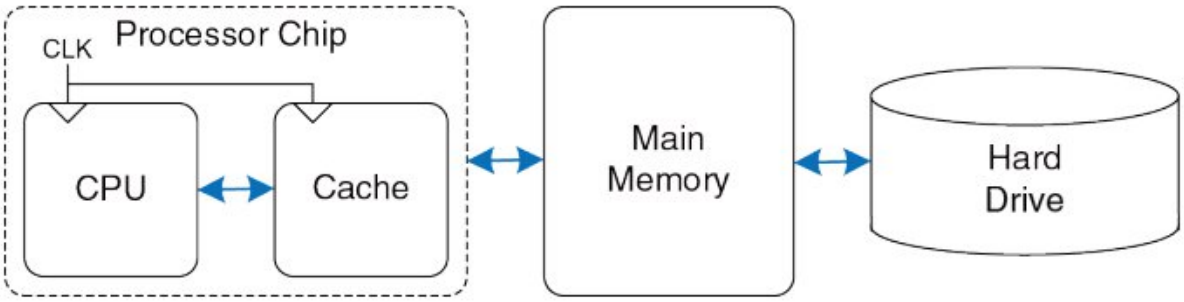
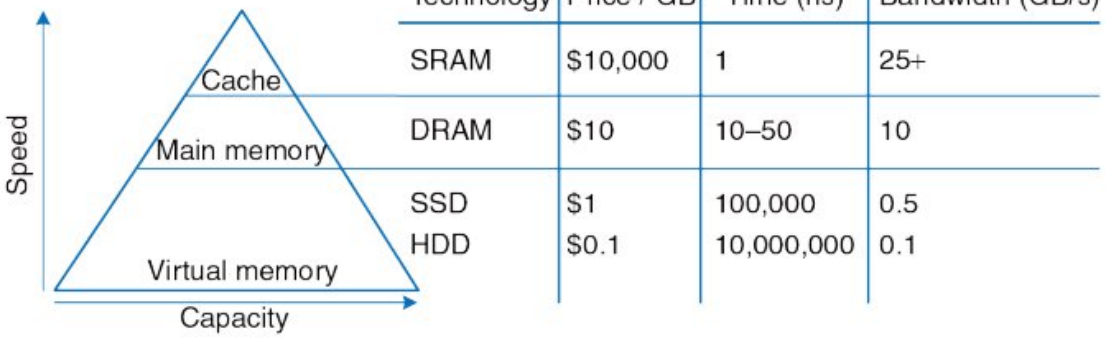


Рис. 8.3 Типичная иерархия памяти

Рис. 8.4 иллюстрирует соотношение емкости и скорости в многоуровневой иерархии памяти компьютерной системы и показывает типичную стоимость, время доступа и пропускную способность для технологий памяти по состоянию на 2012 год.



Technology	Price / GB	Access Time (ns)	Bandwidth (GB/s)
SRAM	$10,000	1	25+
DRAM	$10	10–50	10
SSD	$1	100,000	0.5
HDD	$0.1	10,000,000	0.1

Рис. 8.4 Компоненты иерархии памяти и их характеристики на 2012 год

АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ СИСТЕМ ПАМЯТИ

Чтобы оценить соотношение цены и производительности у разных вариантов систем памяти, разработчикам (и покупателям) компьютеров нужны количественные способы измерения производительности. Мерами измерения производительности систем памяти являются процент попаданий (hit rate) или промахов (miss rate), а также среднее время доступа. Процент попаданий и промахов вычисляется по следующим формулам. (8.1)

$$Miss\ Rate = \frac{Number\ of\ misses}{Number\ of\ total\ memory\ accesses} = 1 - Hit\ Rate$$

$$Hit\ Rate = \frac{Number\ of\ hits}{Number\ of\ total\ memory\ accesses} = 1 - Miss\ Rate \quad (8.1)$$

Среднее время доступа (англ.: Average Memory Access Time, AMAT) – это среднее время, которое процессор тратит, ожидая доступа к памяти при выполнении команд загрузки или сохранения данных. Среднее время доступа вычисляется по следующей формуле, где где tcache, tMM, и tVM – это времена доступа к кэшу, оперативной памяти и диску соответственно, а MRcache и MRMM – это процент промахов кэша и оперативной памяти. (8.2)

$$AMAT = t_{cache} + MR_{cache}(t_{MM} + MR_{MM}t_{VM}) \quad (8.2)$$

КЭШ-ПАМЯТЬ

Кэш содержит часто используемые данные из памяти. Число слов данных, которое он может хранить, называется ёмкостью кэша (англ.: capacity).

КАКИЕ ДАННЫЕ ХРАНЯТСЯ В КЭШ-ПАМЯТИ?

Далее будет упоминаться пространственная и временная локальность:

1) Пространственная локальность. Если произошло обращение по некоторому адресу, то с высокой степенью вероятности в ближайшее время произойдет обращение к соседним адресам.
2) Временная локальность. Если произошло обращение по некоторому адресу, то следующее обращение по этому же адресу с большой вероятностью произойдет в ближайшее время.

Идеальный кэш должен предугадывать, какие данные понадобятся процессору, и выбирать их из оперативной памяти заранее таким образом, чтобы кэш имел нулевой процент промахов. Но поскольку точно предсказать будущее невозможно, то кэш должен угадывать, какие данные понадобятся, основываясь на предыдущих обращениях в память. В частности, кэш использует временну́ю и пространственную локальность, чтобы уменьшить процент промахов в кэш.

Из пространственной локальности следует, что когда кэш читает одно слово данных из памяти, он заодно читает и несколько соседних строк. Эта группа слов называется строкой кэша (англ.: cache line), также иногда используют термин «блок кэша» (англ.: cache block). Число слов в строке b называется длиной строки (line size или block size). Кэш емкость C содержит B = C/b строк.

КАК НАЙТИ ДАННЫЕ В КЭШ-ПАМЯТИ?

Кэш-память классифицируется по числу строк в наборе.

Выделяют кэш-память прямого отображения, многосекционный наборно-ассоциативный кэш и полностью ассоциативный кэш.

Рассмотрим части адреса при отображении в кэш. Два младших бита адреса называются байтовым смещением (byte offset), поскольку они указывают на номер байта внутри слова. Следующие три бита называются номером набора (set bits), так как они указывают на номер набора, в который отображается этот адрес. Оставшиеся 27 бит тега указывают на адрес слова, которое в текущий момент находится в этом наборе кэша. (рис.8.6)

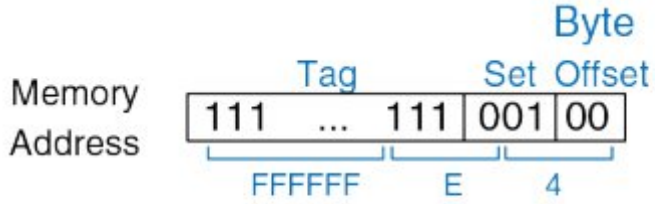


Рис. 8.6 Части адреса при отображении в кэш

В кэш-памяти прямого отображения каждый набор содержит только одну строку (блок данных), так что у него S (количество наборов) = B (количество строк) наборов и строк.

Например, мы имеем кэш ёмкостью 8 строк. В кэше 8 наборов, каждый из которых содержит по одной строке, длина которых равна одному слову. Младшие два бита адреса всегда равны нулю, потому что все адреса выровнены на границу слова. Следующие log28 = 3 бита адресуют один из восьми наборов, в который будет отображен этот адрес памяти. (рис.8.5)

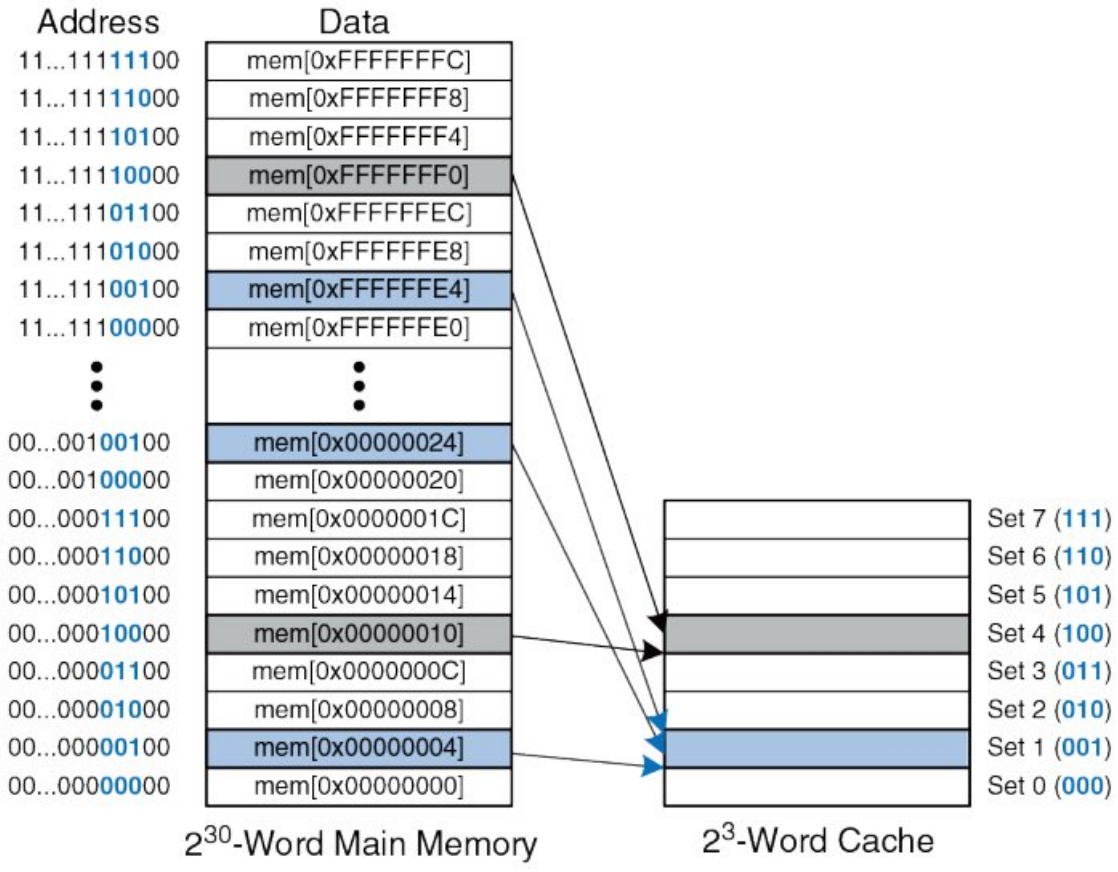


Рис. 8.5 Отображение оперативной памяти на кэш прямого отображения

Так как в один набор кэша отображается множество адресов, то кэш должен отслеживать адреса данных, находящихся в каждом из наборов в текущий момент времени.

Минус кэша прямого отображения заключается в том, что когда два обращения к памяти по разным адресам отображаются в одну и ту же строку кэша, то возникает конфликт и данные, загруженные во время последнего обращения, вытесняют (англ.: evict) из кэша данные, загруженные во время предыдущего обращения. В кэше прямого отображения в каждом наборе есть только одна строка, так что два адреса, отображаемые в одну строку, всегда вызывают конфликт.

Итак, один набор содержит одну строку, и так как в этой одной строке содержится несколько адресов, то вероятность промаха велика.

Далее рассмотрим многосекционный наборно-ассоциативный кэш.

N-секционный наборно-ассоциативный кэш (англ.: N-way set associative cache) уменьшает число конфликтов путем расширения набора до N строк. Каждый адрес памяти по-прежнему отображается в строго определенный набор, но теперь он может быть отображен в любую из N строк этого набора. Можно сказать, что кэш прямого отображения – это односекционный наборно-ассоциативный кэш. Число N называют степенью ассоциативности кэша.

Наборно-ассоциативные кэши, как правило, имеют меньший процент промахов, чем кэши прямого отображения той же ёмкости, так как в них происходит меньше конфликтов. Однако, они обычно медленнее и дороже в реализации, так как необходимо использовать дополнительные компараторы и выходной мультиплексор. Кроме того, в таких кэшах возникает вопрос о том, какую именно секцию замещать, когда все они заняты.

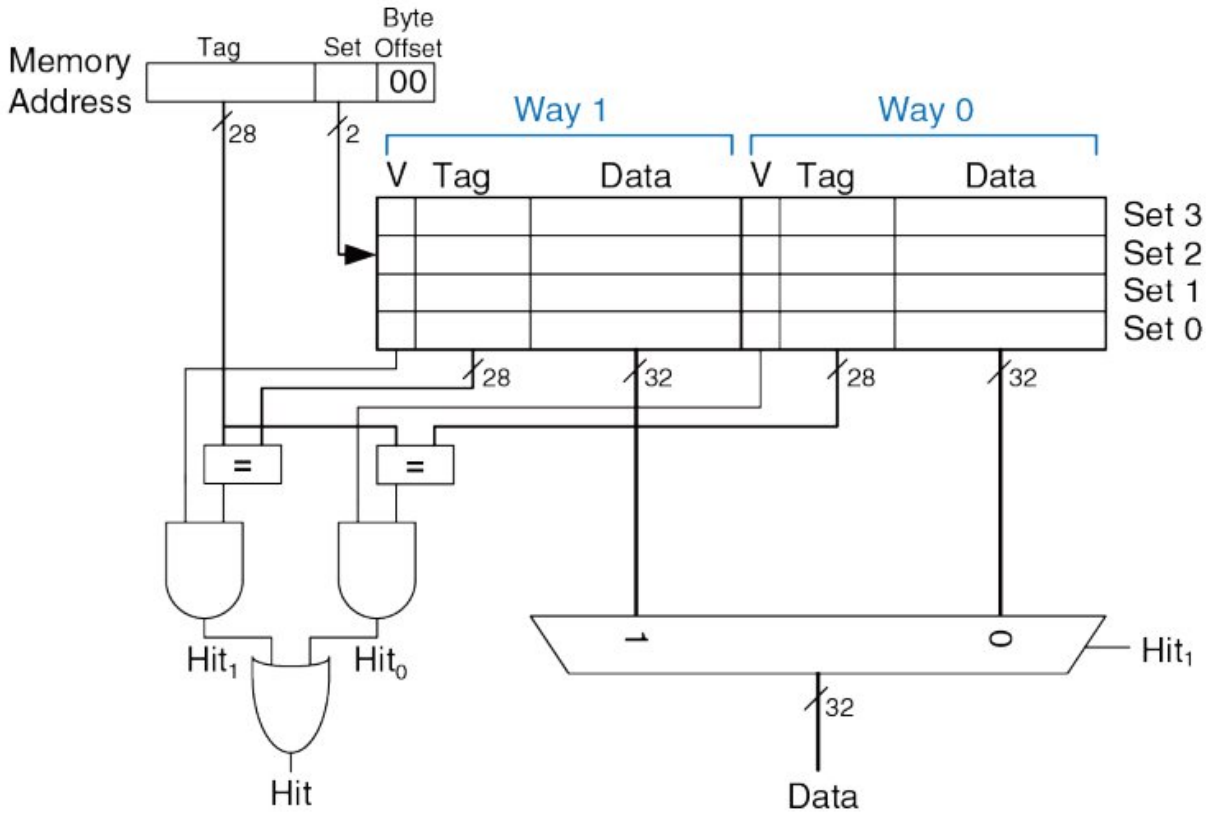


Рис. 8.9 Двухсекционный наборно-ассоциативный кэш

Полностью ассоциативный кэш (англ.: fully associative cache) состоит из одного набора с B секциями, где B – число строк (блоков данных). Адрес памяти может быть отображен в строку любой из этих секций. Можно сказать, что полностью ассоциативный кэш – это B-секционный наборно-ассоциативный кэш с одним набором. Полностью ассоциативные кэши обеспечивают при прочих равных условиях минимально возможное количество конфликтов, но требуют еще больше аппаратуры для дополнительных сравнений тегов.

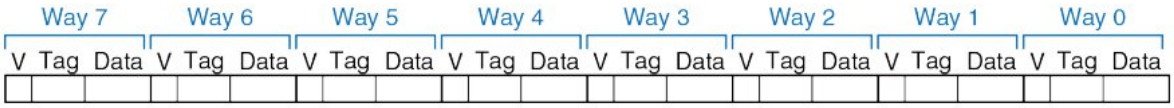


Рис. 8.11 Полностью ассоциативный кэш с восемью строками

Вывод: виды кэш-памяти отличаются лишь числом строк в наборе. Чем больше количество строк, тем меньше вероятность промаха, но скорость реализации также становится меньше, а цена увеличивается.

ДЛИНА СТРОКИ

В предыдущих примерах мы использовали преимущества исключительно временной локальности данных, так как длина строки (т.е. размер блока данных) была равна одному слову. *Чтобы воспользоваться пространственной локальностью, в кэш-памяти используют бóльшие по размеру строки, содержащие несколько последовательных слов. Преимущества строк с длиной, превышающей одно слово, заключается в том, что когда случается промах кэша и требуется прочитать слово данных из памяти, то в эту строку заодно загружаются и соседние слова. Таким образом, последующие*

обращения с большей вероятностью приведут к попаданию в кэш из-за пространственной локальности данных. Однако, увеличившаяся длина строки означает, что кэш того же размера теперь будет иметь меньшее число самих строк. Это может привести к увеличению числа конфликтов и, соответственно, увеличить вероятность промахов кэша. Более того, потребуется больше времени на чтение данных в строку после промаха, т.к. из памяти необходимо будет прочитать не одно, а несколько слов. Время, требуемое для загрузки данных в строку кэша после промаха, называется ценой промаха (англ.: miss penalty). Если соседние слова данных в строке не будут использованы в дальнейшем, то усилия на их загрузку будут потрачены зря. Тем не менее, большинству реальных программ увеличение длины строки приносит пользу.

В Табл. 8.2 перечислены различные способы организации кэш-памяти. Любой адрес в памяти отображается только в один набор, но соответствующие этому адресу данные могут оказаться в любой из секций этого набора.

N – число строк в наборе

B- число строк (C(ёмкость)/b(длина строки))

Табл. 8.2 Способы организации кэш-памяти

Способ организации	Количество секций (N)	Количество наборов (S)
Прямого отображения	1	*B*
Наборно-ассоциативный	$1 < N < B$	*B*/*N*
Полностью ассоциативный	*B*	1

КАКИЕ ДАННЫЕ ЗАМЕСТИТЬ В КЭШ-ПАМЯТИ?

В кэш-памяти прямого действия когда нужно загрузить новые данные в набор, который уже содержит данные, то строка в наборе просто замещается на новые данные

В наборно-ассоциативной и полностью ассоциативной кэш-памяти нужно решить, какую именно из нескольких строк в наборе вытеснить. Учитывая принцип временной локальности, наилучшим вариантом было бы заменить ту строку, которая дольше всего не использовалась, потому что маловероятно, что она будет использована снова. Именно поэтому большинство кэшей используют стратегию замены редко используемых данных (англ.: least recently used, LRU).

УЛУЧШЕННАЯ КЭШ-ПАМЯТЬ

В современных системах для сокращения времени доступа к памяти используются несколько уровней кэша.

Многоуровневые кэши

Чем больше размер кэша, тем больше вероятность, что интересующие нас данные в нем найдутся, и, следовательно, тем меньше у него будет частота промахов, но с увеличением размера кэша уменьшается его скорость. Поэтому в современных системах используется два уровня кэша. Кэш первого уровня (L1) маленький, но имеет высокую скорость. К нему обращаются в первую очередь. Кэш второго уровня (L2) больше по размеру и поэтому медленнее, к нему обращаются во вторую очередь.

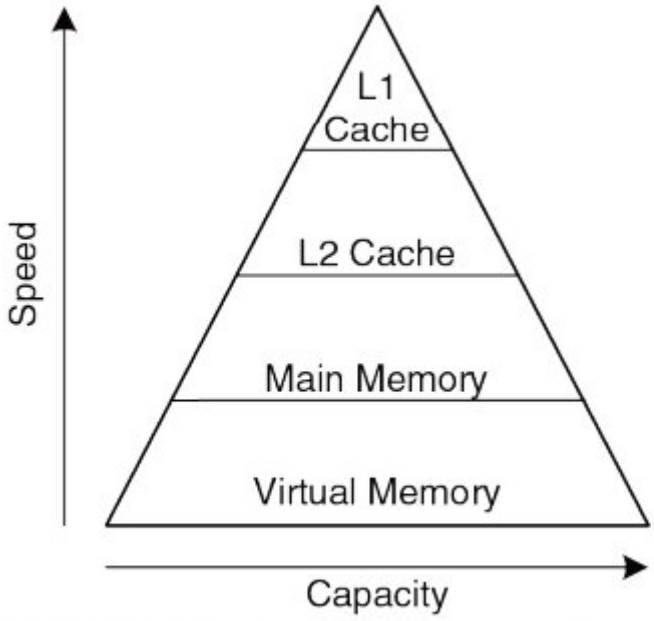


Рис. 8.16 Иерархия памяти с двумя уровнями кэша

Сокращение частоты промахов

Процент промахов кэша можно сократить, изменяя его емкость, длину строки и/или ассоциативность. Промахи кэша делятся на неизбежные промахи (compulsory misses), промахи из-за недостаточной емкости (capacity misses) и промахи из-за конфликтов (conflict misses).

Первое обращение к строке кэша всегда приводит к неизбежному промаху, так как эту строку нужно прочесть из оперативной памяти хотя бы один раз независимо от архитектуры кэша. Промахи из-за недостаточной емкости происходят, когда кэш слишком мал для хранения всех одновременно используемых данных. Промахи из-за конфликтов случаются, если несколько адресов памяти отображаются на один и тот же набор кэша и выталкивают из него данные, которые все еще нужны.

Изменение параметров кэша может повлиять на частоту одного или нескольких типов промахов.

С увеличением емкости кэша частота промахов из-за недостатка емкости сокращается. Увеличение ассоциативности, особенно для кэшей небольшого размера, сокращает количество промахов из-за конфликтов, показанных вдоль верхней части кривой. (рис.8.17)

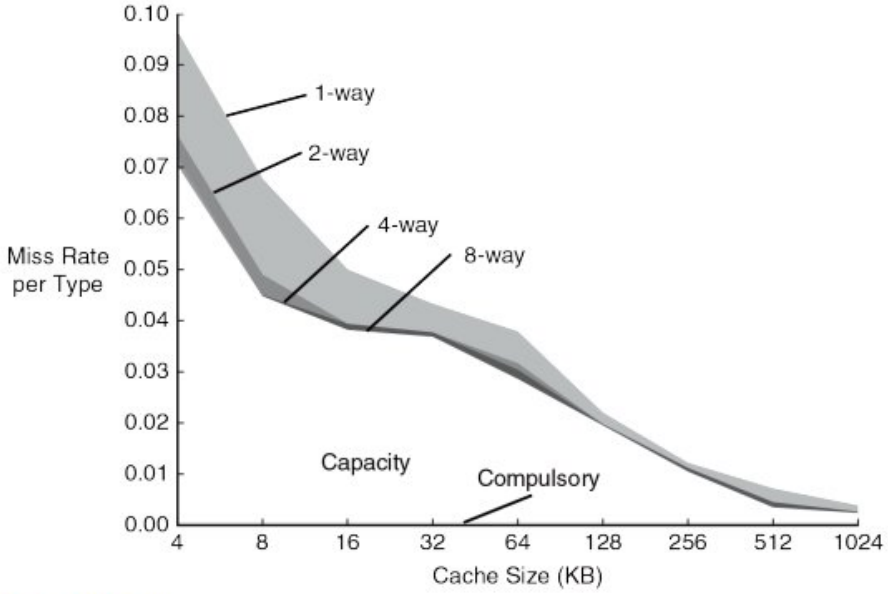


Рис. 8.17 Зависимость частоты промахов от размера и ассоциативности кэша на тестах SPEC2000.
График из книги Hennessy and Patterson, Computer Architecture: A Quantitative Approach, 5th ed., Morgan Kaufmann, 2012, с разрешения авторов

Увеличение размера кэша может сократить промахи из-за конфликтов и промахи из-за недостатка емкости, но никак не повлияет на число неизбежных промахов. Увеличение длины строки может сократить число неизбежных промахов (благодаря локальности данных), но одновременно может увеличить частоту промахов из-за конфликтов, поскольку большее адресов будет отображаться на один и тот же набор, увеличивая вероятность конфликтов. (рис.8.18)

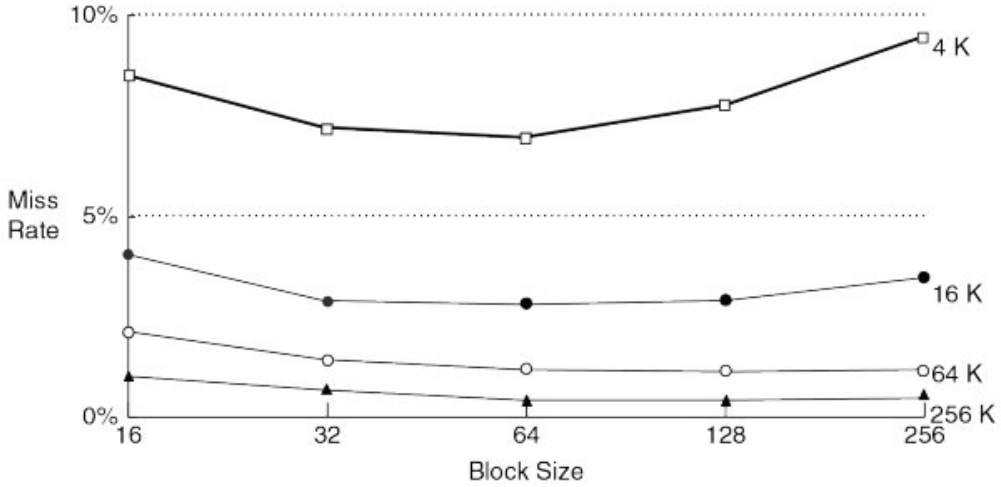


Рис. 8.18 Зависимость частоты промахов от длины строки и размера кэша на тестах SPEC92.
График из книги Hennessy and Patterson, Computer Architecture: A Quantitative Approach, 5th ed., Morgan Kaufmann, 2012, с разрешения авторов

Стратегии записи

Запись в память выполняется примерно так же, как и чтение. При выполнении команды сохранения данных процессор сначала проверяет кэш. В случае промаха кэша соответствующая строка выбирается из оперативной памяти в кэш, а затем в нее

записывается нужное слово. В случае попадания в кэш слово просто записывается в строку.

Кэши делятся на два типа – со сквозной записью (write-through) и с отложенной записью (write-back). Сквозной — сначала полученные сведения оказываются в основной памяти, а потом оттуда дублируются в кэш; отложенный — сначала данные кэшируются, а потом, по истечении определенного срока или при вытеснении, переносятся в основную память.

Эволюция кэш-памяти процессоров MIPS

В Табл. 8.3 прослежена эволюция организации кэш-памяти в процессорах MIPS с 1985 по 2010 год. Основные тенденции – введение нескольких уровней кэша, увеличение емкости и степени ассоциативности.

Табл. 8.3 Эволюция кэш-памяти процессоров MIPS*

Год	Процессор	МГц	Кэш L1	Кэш L2
1985	R2000	16.7	нет	нет
1990	R3000	33	32 Кбайт, прямого отображения	нет
1991	R4000	100	8 Кбайт, прямого отображения	1 Мбайт, прямого отображения
1995	R10000	250	32 Кбайт, двухсекционный	4 Мбайт, двухсекционный
2001	R14000	600	32 Кбайт, двухсекционный	16 Мбайт, двухсекционный
2004	R16000A	800	64 Кбайт, двухсекционный	16 Мбайт, двухсекционный
2010	MIPS32 1074K	1500	32 Кбайт	переменного размера

* Адаптировано из книги D. Sweetman, See MIPS Run, Morgan Kaufmann, 1999.

Виртуальная память

Большинство современных вычислительных систем в качестве нижнего уровня в иерархии памяти используют жесткие диски, представляющие собой магнитные или твердотельные запоминающие устройства (см. Рис. 8.4). По сравнению с идеальной памятью, которая должна быть быстрой, дешевой и большой, жесткий диск имеет большой объем и недорого стоит, однако невероятно медленно работает. Жесткий диск обеспечивает намного больший объем, чем недорогая оперативная память (DRAM). Однако если существенная часть обращений к памяти осуществляется к жесткому диску, скорость работы сильно снижается.

Рис. 8.19 Жесткий диск

Цель включения жесткого диска в иерархию памяти – занедорого создать видимость памяти большого объема, одновременно обеспечивая для большинства обращений к памяти скорость доступа, равную скорости более быстрых типов памяти. Например, компьютер с оперативной памятью на 128 Мбайт может обеспечить видимость наличия 2 Гбайт оперативной памяти, используя для этого жесткий диск. В этом случае большая память, объёмом 2 Гбайта, называется виртуальной памятью, а меньшая память, объёмом 128 Мбайт, называется физической (оперативной) памятью .

Физическая память выступает в роли кэша для виртуальной памяти.

Табл. 8.4 Соответствие терминов кэша и виртуальной памяти

Кэш	Виртуальная память
Строка	Страница
Длина строки	Размер страницы
Смещение относительно начала строки	Смещение относительно начала страницы
Промах	Страничная ошибка
Тег	Номер виртуальной страницы

Процесс преобразования виртуального адреса в физический называется трансляцией адреса.

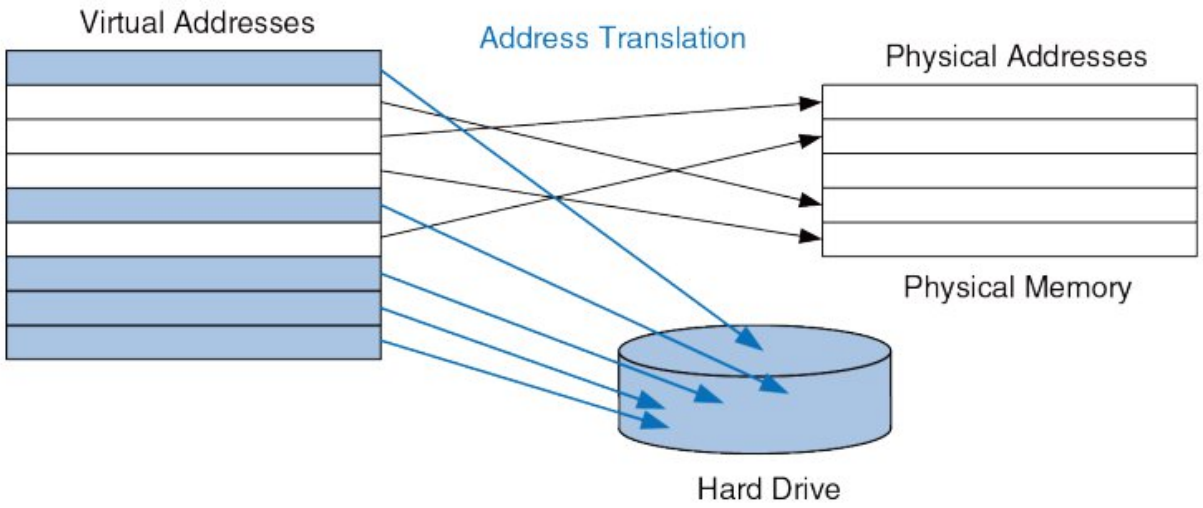


Рис. 8.20 Виртуальные и физические страницы

Чтобы избежать страничных ошибок, вызванных конфликтами, любая виртуальная страница может отображаться на любую физическую страницу. Другими словами, физическая память работает как полностью ассоциативный кэш для виртуальной памяти. В традиционном полностью ассоциативном кэше каждая секция содержит компаратор, который проверяет старшие биты адреса на соответствие тегу, чтобы определить, находятся ли там нужные данные. На практике виртуальная память имеет настолько много физических страниц, что обеспечить компаратором каждую страницу было бы чересчур дорого. Вместо этого в подсистемах виртуальной памяти используется трансляция адреса при помощи таблицы страниц (англ.: page table). Каждая команда загрузки или сохранения данных включает два обращения к физической памяти: обращение к таблице страниц и собственно обращение к данным. Чтобы ускорить трансляцию адреса, используется буфер ассоциативной трансляции (англ.: translation lookaside buffer, TLB), который содержит наиболее часто используемые записи таблицы страниц.

ТРАНСЛЯЦИЯ АДРЕСОВ

В системах с виртуальной памятью программы используют виртуальные адреса и поэтому имеют доступ к памяти большого объема. Компьютер должен транслировать эти виртуальные адреса, чтобы либо найти соответствующий адрес в физической памяти, либо получить страничную ошибку и загрузить данные с жесткого диска.

В любой момент времени физическая память может хранить максимум 1/16 от числа страниц виртуальной памяти. Остальные виртуальные страницы хранятся на жестком диске. Чтобы получить физический адрес из виртуального, необходимо транслировать только номер страницы.

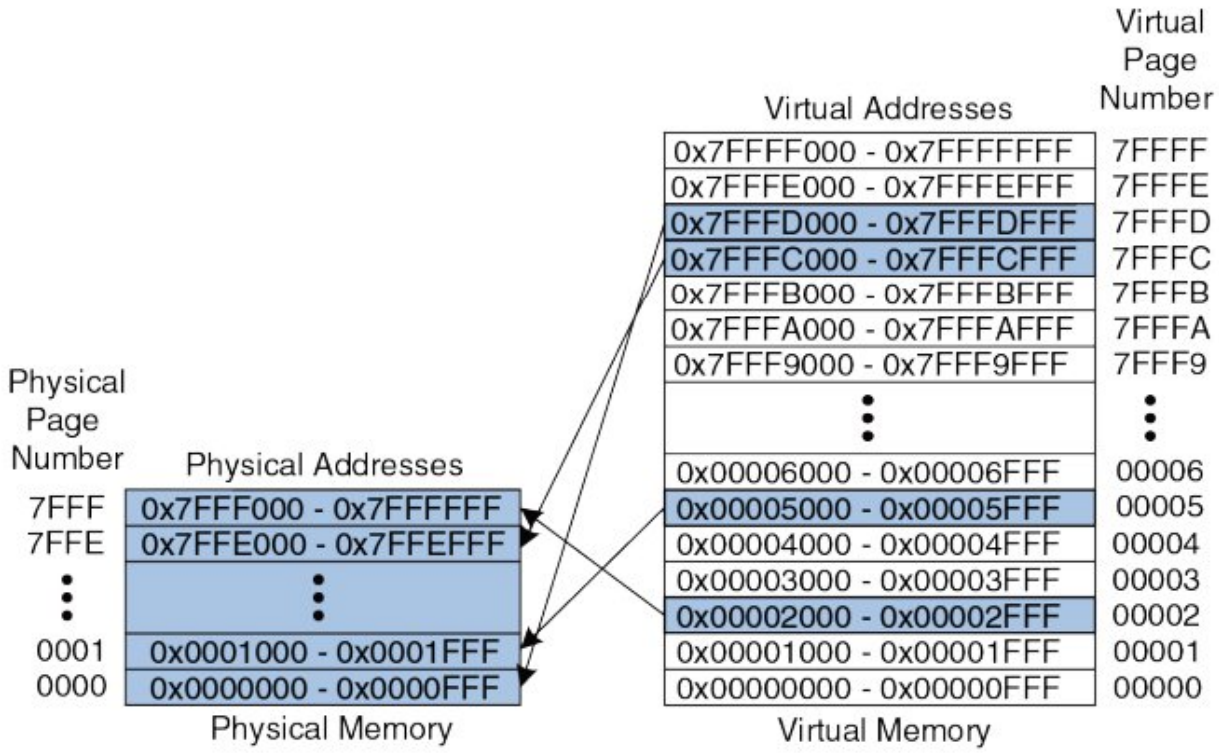


Рис. 8.21 Физические и виртуальные страницы

ТАБЛИЦА СТРАНИЦ

Процессор использует таблицу страниц для трансляции виртуальных адресов в физические. Таблица страниц содержит отдельную запись для каждой виртуальной страницы. Эта запись содержит номер физической страницы и бит достоверности (англ.: valid bit). Если бит достоверности равен 1, виртуальная страница отображается на физическую страницу, номер которой указан в записи. В обратном случае виртуальная страница находится на жестком диске.

Поскольку таблица страниц велика, она хранится в физической памяти.

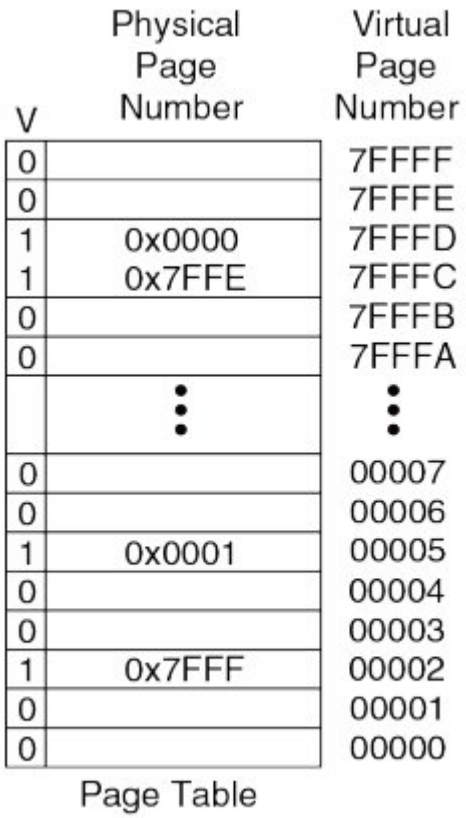


Рис. 8.23 Таблица страниц для случая, показанного на Рис. 8.21

Таблица страниц может храниться в любом месте физической памяти, ее расположение определяется операционной системой. Процессор обычно использует выделенный регистр, называемый регистром таблицы страниц, для хранения ее базового адреса.

БУФЕР АССОЦИАТИВНОЙ ТРАНСЛЯЦИИ

Виртуальная память оказывала бы огромное негативное влияние на производительность, если бы требовалось обращение к таблице страниц при выполнении каждой команды загрузки или сохранения – это удваивало бы время выполнения этих команд. К счастью, обращения к таблице страниц имеют высокую временную локальность.

В целом, процессор может хранить несколько последних записей, прочитанных из таблицы страниц в небольшой кэш-памяти, называемой буфером ассоциативной трансляции (TLB).

Процессор «заглядывает» в TLB в поисках информации о трансляции, прежде чем обратиться к таблице страниц в физической памяти.

Каждая запись в TLB хранит номер виртуальной страницы и соответствующий ей номер физической страницы. Обращение к TLB происходит по номеру виртуальной страницы.

Буфер ассоциативной трансляции разрабатывают таким образом, чтобы он был маленьким и чтобы доступ к нему занимал менее одного такта. Даже при этом доля попаданий в него обычно превышает 99%. TLB уменьшает число обращений к памяти, требуемое для большинства команд загрузки и сохранения, с двух до одного.

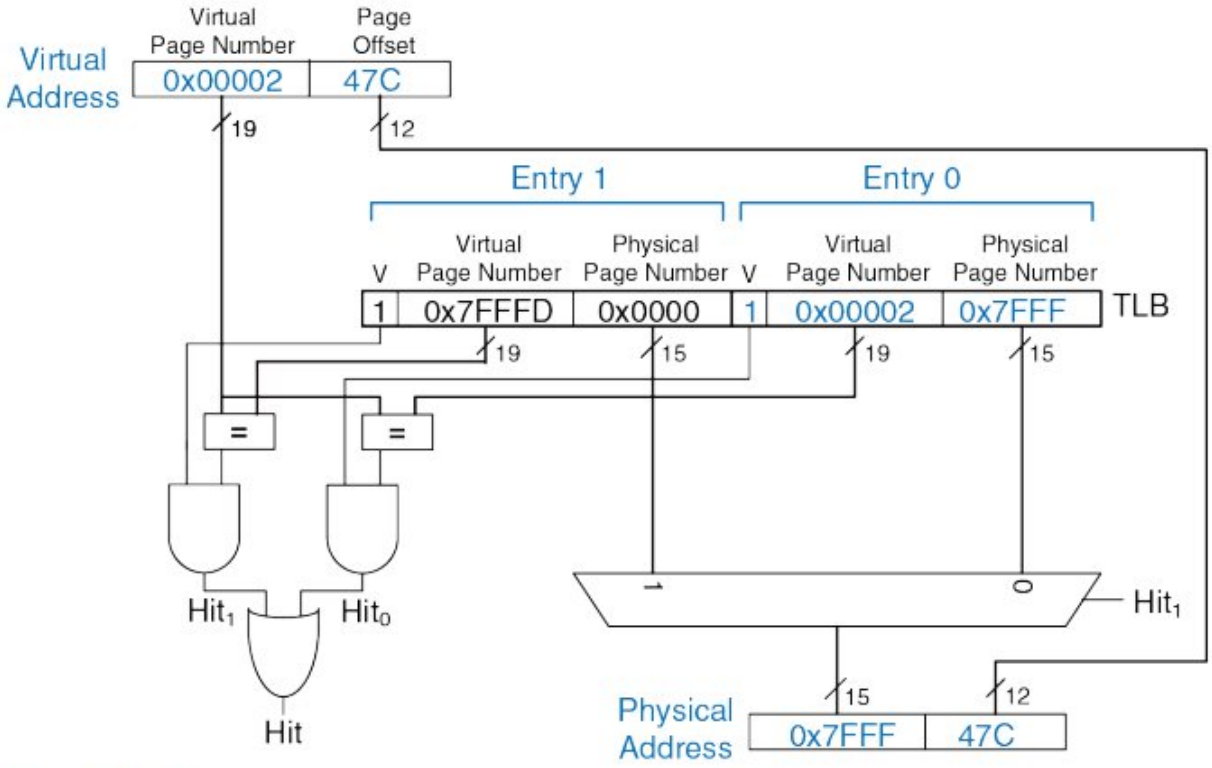


Рис. 8.25 Трансляция адреса с использованием TLB