Кеши.

Кэш— промежуточный буфер с быстрым доступом, содержащий информацию, которая может быть запрошена с наибольшей вероятностью. Доступ к данным в кэше осуществляется быстрее, чем выборка исходных данных из более медленной памяти или удаленного источника, однако её объём существенно ограничен по сравнению с хранилищем исходных данных.

Уровни кэша

Кэш центрального процессора разделён на несколько уровней. Максимальное количество кэшей — четыре. В универсальном процессоре в настоящее время число уровней может достигать трёх. Кэш-память уровня N+1, как правило, больше по размеру и медленнее по скорости доступа и передаче данных, чем кэш-память уровня N.

•Самым быстрым является кэш первого уровня — L1 cache (level 1 cache). По сути, она является неотъемлемой частью процессора, поскольку расположена на одном с ним кристалле и входит в состав функциональных блоков. В современных процессорах обычно L1 разделен на два кэша — кэш команд (инструкций) и кэш данных (Гарвардская архитектура). Большинство процессоров без L1 не могут функционировать. L1 работает на частоте процессора, и, в общем случае, обращение к нему может производиться каждый такт. Зачастую является возможным выполнять несколько операций чтения/записи одновременно.

•Вторым по быстродействию является кэш второго уровня— L2 cache, который обычно, как и L1, расположен на одном кристалле с процессором. В ранних версиях процессоров L2 реализован в виде отдельного набора микросхем памяти на материнской плате. Объём L2 от 128 кбайт до 1−12 Мбайт. В современных многоядерных процессорах кэш второго уровня, находясь на том же кристалле, является памятью раздельного пользования— при общем объёме кэша в n Мбайт на каждое ядро приходится по n/c Мбайта, где c— количество ядер процессора.

•Кэш третьего уровня наименее быстродействующий, но он может быть очень большим— более 24 Мбайт. L3 медленнее предыдущих кэшей, но всё равно значительно быстрее, чем оперативная память. В многопроцессорных системах находится в общем пользовании и предназначен для синхронизации данных различных L2.

•Существует четвёртый уровень кэша, применение которого оправдано только для многопроцессорных высокопроизводительных серверов и мейнфреймов. Обычно он реализован отдельной микросхемой.

Когерентность кэша — свойство кэшей, означающее целостность данных, хранящихся в локальных кэшах для разделяемого ресурса. Когерентность кэшей — частный случайкогерентности памяти.

Когерентность памяти — свойство компьютерных систем, в которых два или более процессора или ядра имеют доступ к общей области памяти.
Воднопроцессорных системах (более строго — в одноядерных) лишь один процессорный узел выполняет всю работу, следовательно, только он один может выполнять чтение и запись определённой области памяти. В результате, когда содержимое ячейки меняется, все последующие операции чтения по данному адресу получат обновлённое значение даже при наличии кэширования.

С другой стороны, в многопроцессорных (многоядерных) системах несколько процессорных узлов работают одновременно, поэтому возможна ситуация параллельного доступа к одной ячейке памяти. При условии, что ни один из них не меняет значение данной ячейки, они могут сводобно пользоваться ей совместно, кэшируя по своему усмотрению. Но как только один из них обновляет значение ячейки, данные в локальных кэшах других узлов могут оказаться устаревшими. Следовательно, необходим механизм уведомления всех узлов об изменении значения в общей памяти; такой механизм называется протоколом когерентности. Если подобный протокол применён, то говорят, что система имеет «**когерентную память**».

Точная природа и смысл механизма когерентности определяется моделью консистентности, реализованной в протоколе. Чтобы писать правильные параллельные программы, программисты должны быть в курсе того, какая именно модель консистентности применена в их системах.

Когерентность определяет поведение чтений и записей в одно и то же место памяти. Кэш называется когерентным, если выполняются следующие условия:

1.Если процессор P записывает значение в переменную X, то при следующем считывании X он должен получить ранее записанное значение, если между записью и чтением X другой процессор не производил запись в X. Это условие связано с сохранением порядка выполнения программы, это должно выполняться и для однопоточной архитектуры.

2.Операция чтения X процессором P_1, следующая после того, как другой процессор P_2 существил запись в X, должна вернуть записанное значение, если другие процессоры не изменяли X между двумя операциями. Это условие определяет понятие когерентной видимости памяти.

3.Записи в одну и ту же ячейку памяти должны быть последовательными. Другими словами, если два процессора записывают в переменную X два значения: A, затем B — не должно случиться так, чтобы при считывании процессор сначала получал значение B, а затем A.

В этих условиях предполагается, что операции чтения и записи происходят мгновенно. Однако этого не происходит на практике из-за задержек памяти и других особенностей архитектуры. Изменения, сделанные процессором P_1, могут быть не видны процессору P_2, если чтение произошло через очень маленький промежуток времени после записи. Модель консистентности памяти определяет, когда записанное значение будет видно при чтении из другого потока.

Механизмы когерентности кэшей

•Когерентность с использованием справочника (*directory*). Информация о состоянии блока физической памяти содержится только в одном месте, называемом справочником (физически справочник может быть распределен по узлам системы).

•Когерентность с использованием отслеживания (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 (разделяемой) шине и контроллеры всех кэшей

наблюдают за шиной (просматривают ее) для определения того, не содержат ли они копию соответствующего блока.

•Перехват (snarfing). Когда из какого-либо одного кэша данные переписываются в оперативную память, контроллеры остальных получают сигнал об этом изменении ("перехватывают" информацию об изменении данных) и, если необходимо, изменяют соответствующие данные в своих кэшах.

Системы распределенной разделяемой памяти используют похожие механизмы для поддержания корректности между блоками памяти в слабосвязанных системах.

Протоколы поддержки когерентности отвечают за поддержание корректности данных между всеми кэшами в системе. Протокол поддерживает когерентность памяти согласно выбранной модели.

Протокол MSI.

В протоколе MSI каждый блок памяти может иметь одно из трех состояний:

MODIFIED – данные в блоке изменены и не согласуются с данными в основной памяти. Блок в этом состоянии должен будет перезаписать данные в основную память.

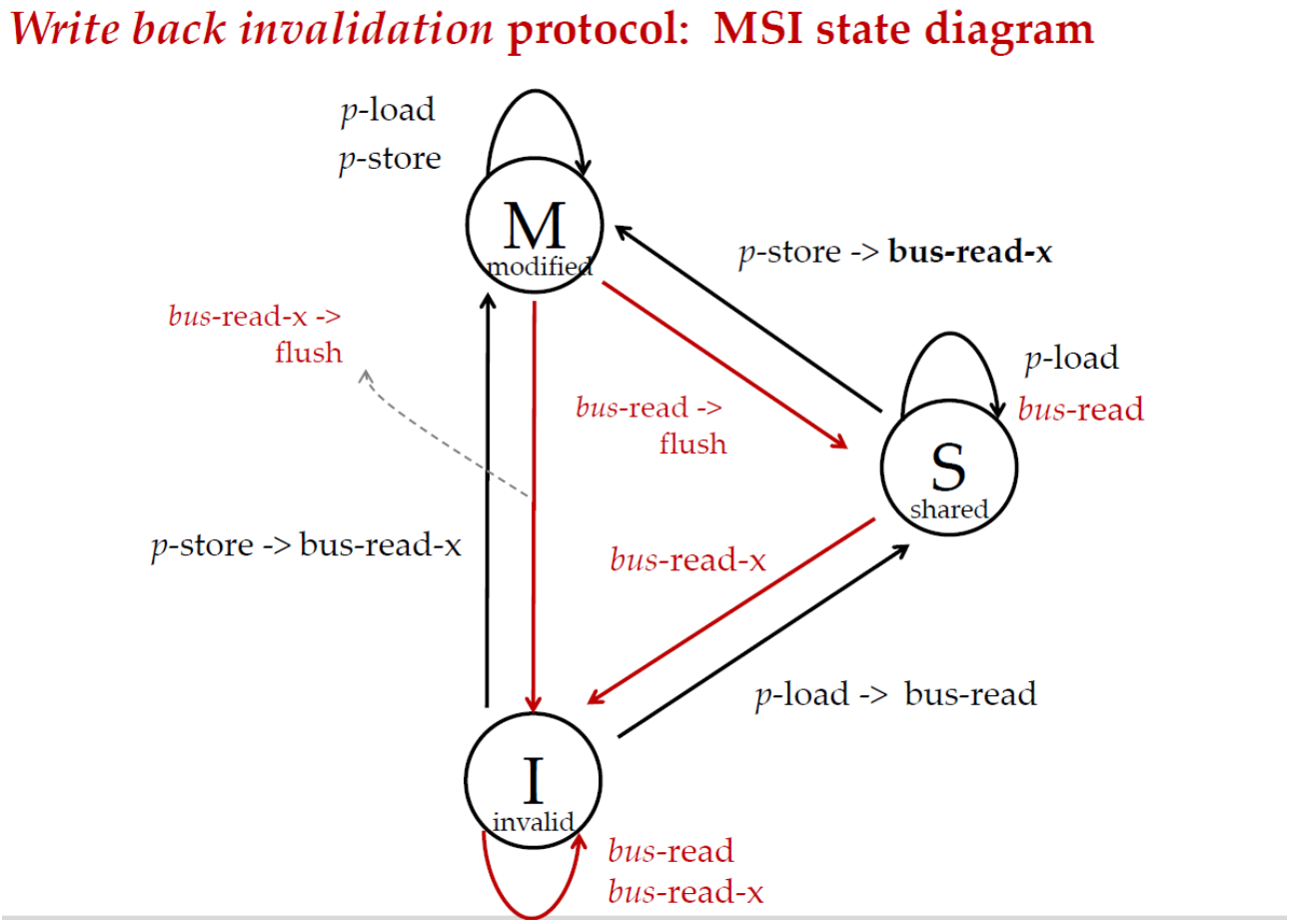
SHARED – данные в блоке не модифицированы. В этом состоянии блок предназначен только для чтения до тех пор, пока остается хотя бы один блок в таком состоянии.

INVALID – данные в этом блоке невалидны и должны быть записаны из других кэшей или основной памяти.

Когда запрос на чтение прибывает в кэш для блока в состоянии "M" или "S" , кэш отдает данные. Если блок в состоянии "I", он должен убедиться, что нет этого блока в состоянии "M" в любом другом кэше. Если другой кэш имеет блок в состоянии "M", то он должен записать данные в основную память и перейти в состояние "S" или "I". После того, как любой блок "M" записывается обратно, кэш получает данные либо из резервного хранилища, или другого кэша с блоком в состоянии «S». Теперь кэш может поставлять данные запрашивающей стороне. После подачи данных, блок кэш-памяти находится в состоянии «S».

Когда запрос записи прибывает в кэш для блока в состоянии "M", кэш изменяет данные локально. Если ячейка находится в состоянии "S", кэш должен уведомить любые другие кэши, которые могут содержать блок в состоянии "S", что они должны инвалидировать ячейку. Затем данные могут быть изменены локально. Если блок находится в состоянии "I", кэш должен уведомить любые другие кэши, которые могут содержать блок в "S" или "M" говорится, что они должны инвалидировать блок. Если блок находится в другом кэше в состоянии "M", то кэш должен либо написать данные в основную память, либо поставить его запрашивающему кэшу. Если в этот момент кэш еще не имеют блок на уровне кэша, блок считывается из основной памяти, прежде чем изменять в памяти кэша. После модификации

данных, блок кэш-памяти находится в состоянии "M".



Для любой данной пары кэшей, разрешенные состояния блоков памяти на одной линии:

	MODIFIED	*SHARED*	*INVALID*
MODIFIED	NO	NO	YES
SHARED	NO	YES	YES
INVALID	YES	YES	YES