Структурная и функциональная организация ЭВМ (Computer Organization and Design)

БГУИР
кафедра ЭВМ

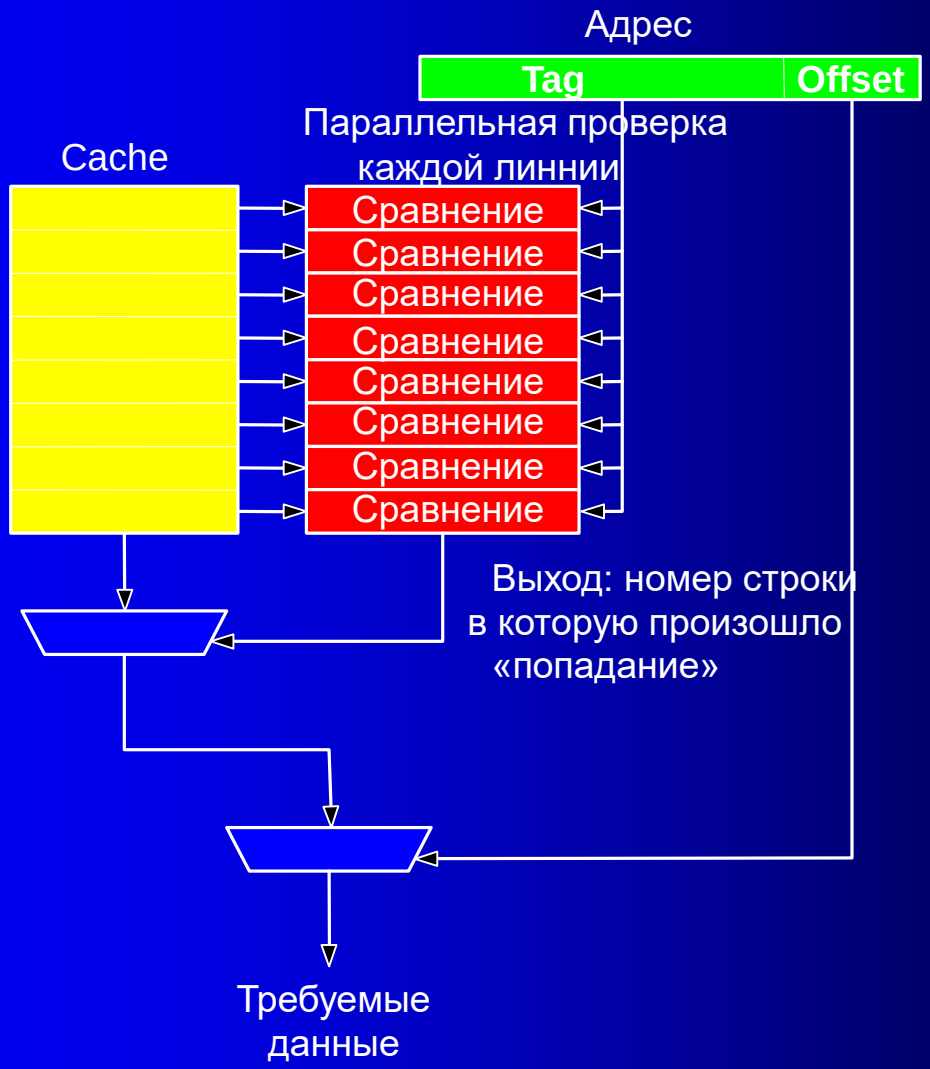
Лекция 22
«Кэш-II»

2019

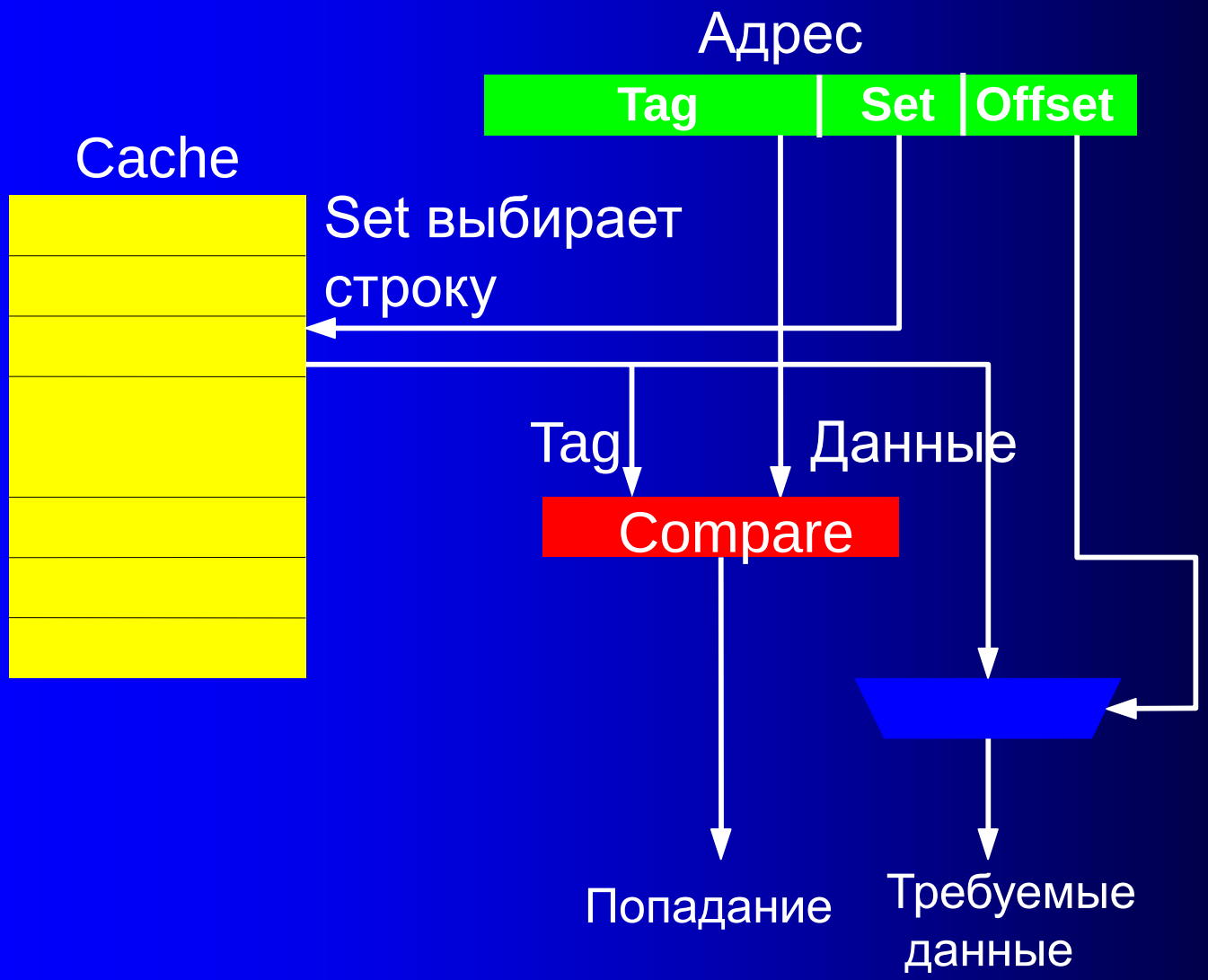
План лекции

1. Синхронизация Кэш-памяти и основной памяти
2. Что и как хранить в кэш?
3. Дисковая кэш
4. Система ввода-вывода

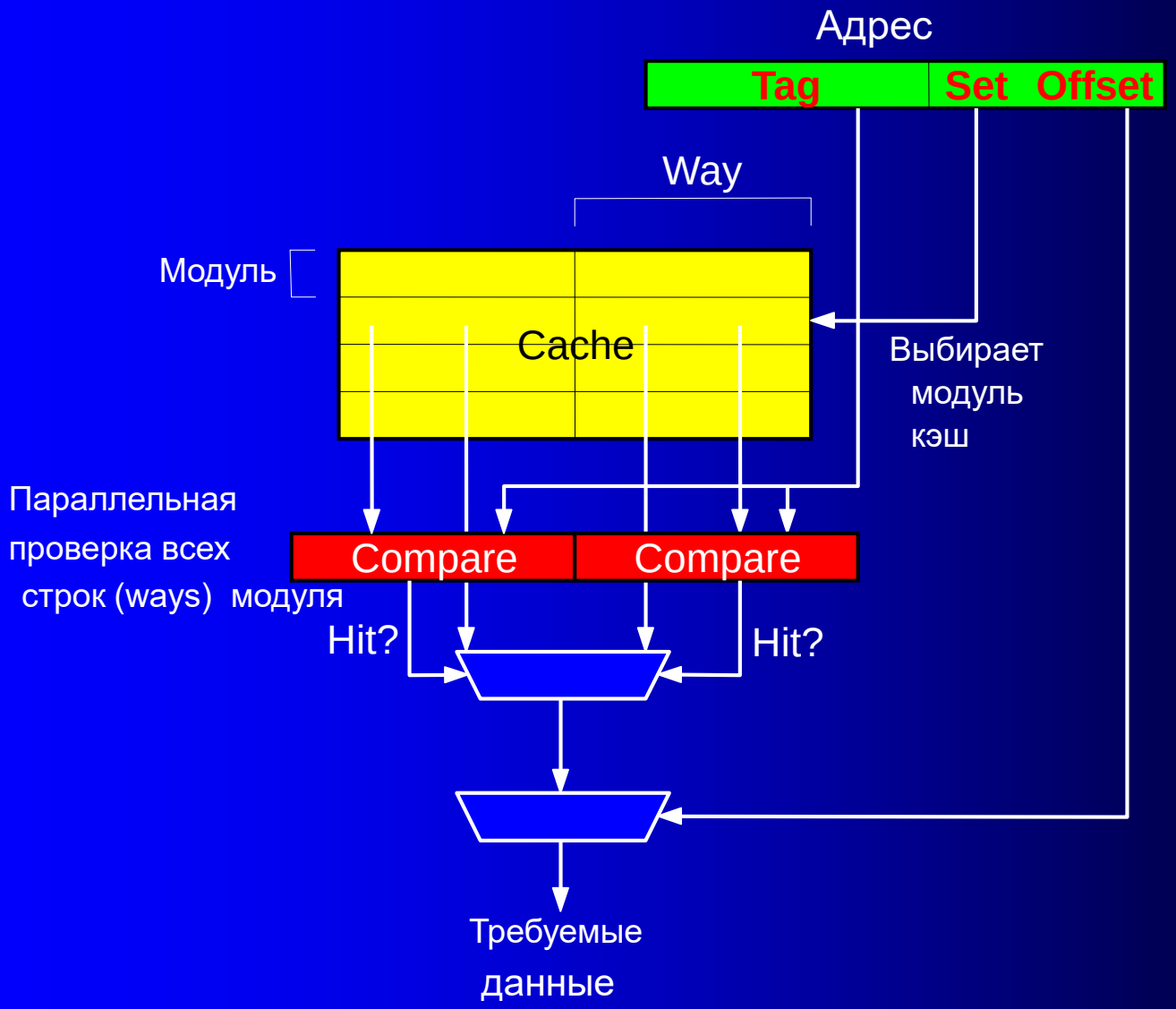
Полностью ассоциативное отображение (Fully-Associative Cache)



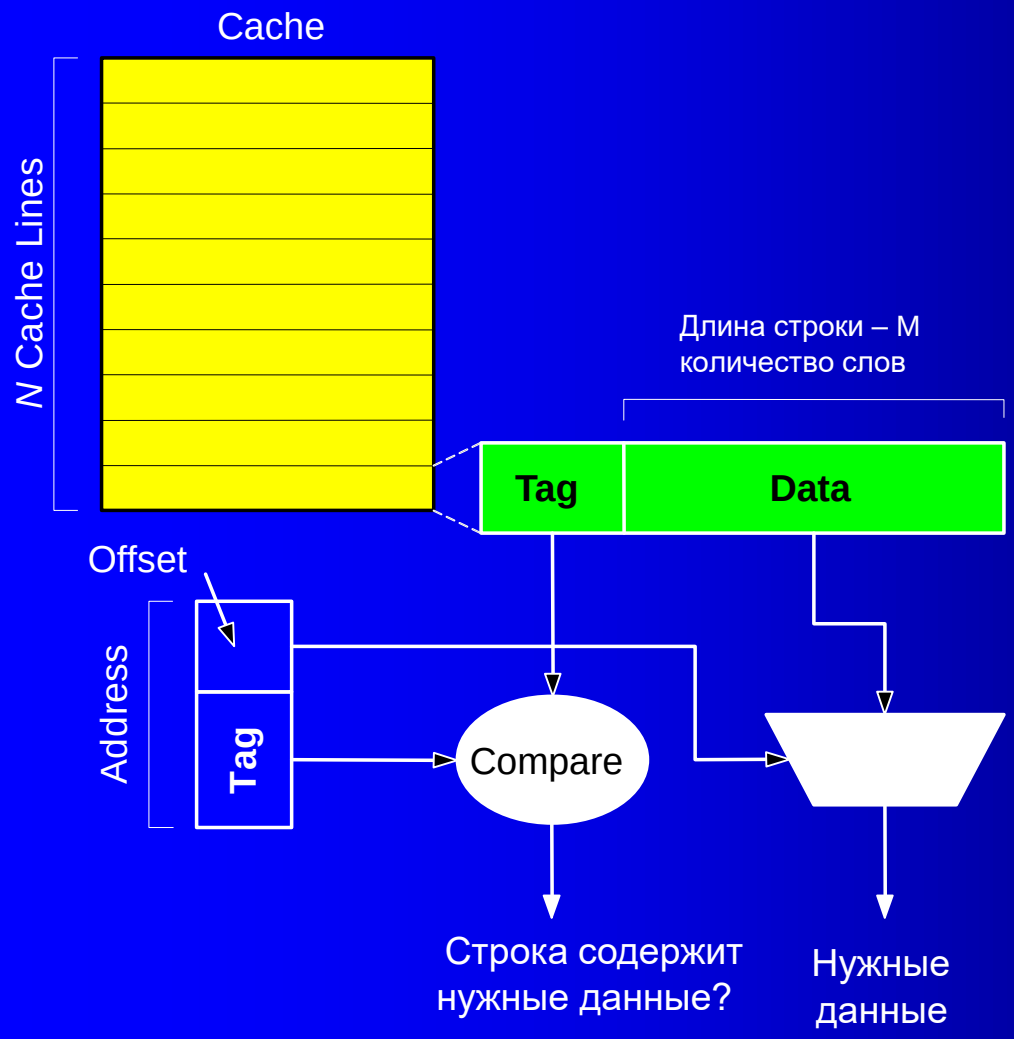
Direct-Mapped Cache



Set-Associative Caches



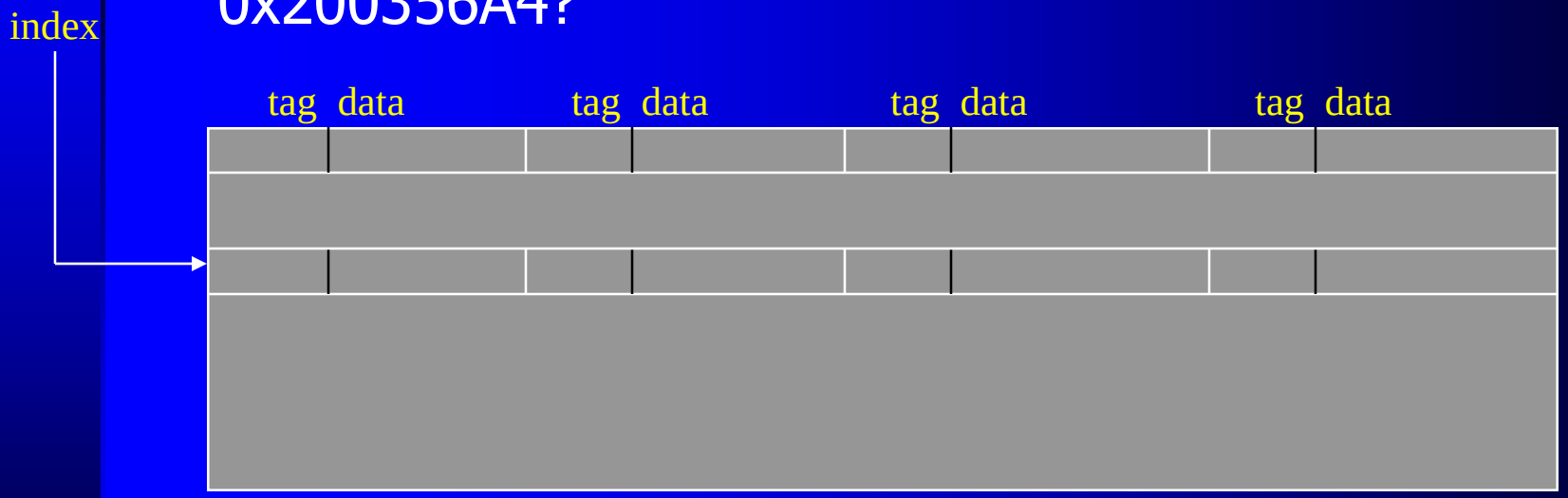
A Typical Cache Operation



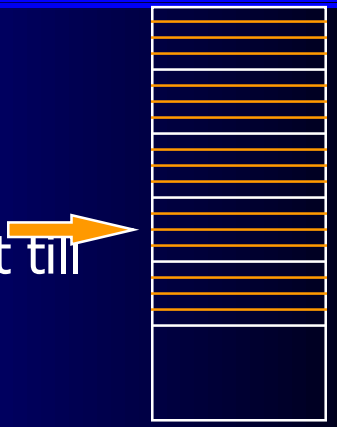
- При доступе к памяти, сперва просмотреть кэш
- Если нужный адрес есть в кэш – то завершить операцию за один цикл
- Если нет – то обращение к основной памяти (many cycles)

Thought Experiment

- 16 KB, 4-way set-associative cache, 32-bit address, byte-addressable memory, 32-byte cache blocks/lines
- how many tag bits?
- Where would you find the word at address 0x200356A4?



Какой блок должен быть замещён в случае промаха?



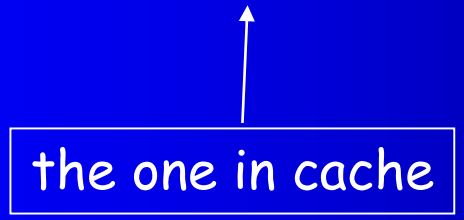
- Direct Mapped – ответ очевиден
- Множественно ассоциативная (Set associative) или полностью ассоциативная:
  - Тот который больше всего не понадобится longest till next use (ideal, impossible)
  - Наиболее долго не использовавшийся least recently used (best practical approximation)
    - Счётчик – обнуление при обращении, иначе +1
    - Очередь, при каждом обращении к строке ссылка на неё в конец очереди. Первая строка в очереди - замещается
  - pseudo-LRU (e.g., NMRU, NRU)
  - FIFO – удаляется строка, дольше всех бывшая в кэш
  - LFU – least frequently used, счётчик +1, строка с минимальным значением – вылетает.
  - случайный (easy)

  - how many bits for LRU?

Синхронизация данных

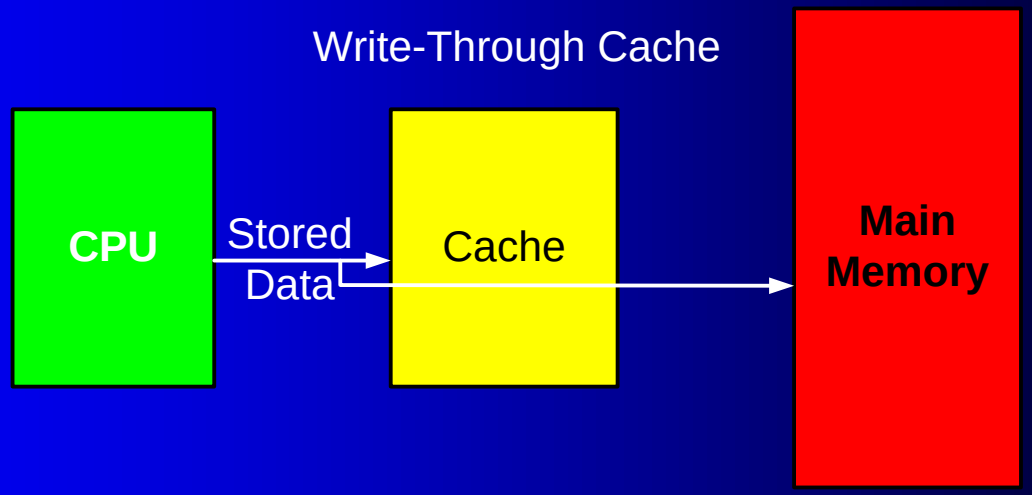
- Управление хранением данных должно отличаться от загрузки в силу следующих причин:
  - сохранение данных не требует простоя ЦП.
  - сохранение меняет содержимое кэш
  - многие уст-ва в/вывода имеют возможность прямого доступа в память.

При этом возникает вопрос – о «memory consistency» ... how do you ensure memory gets the correct value?

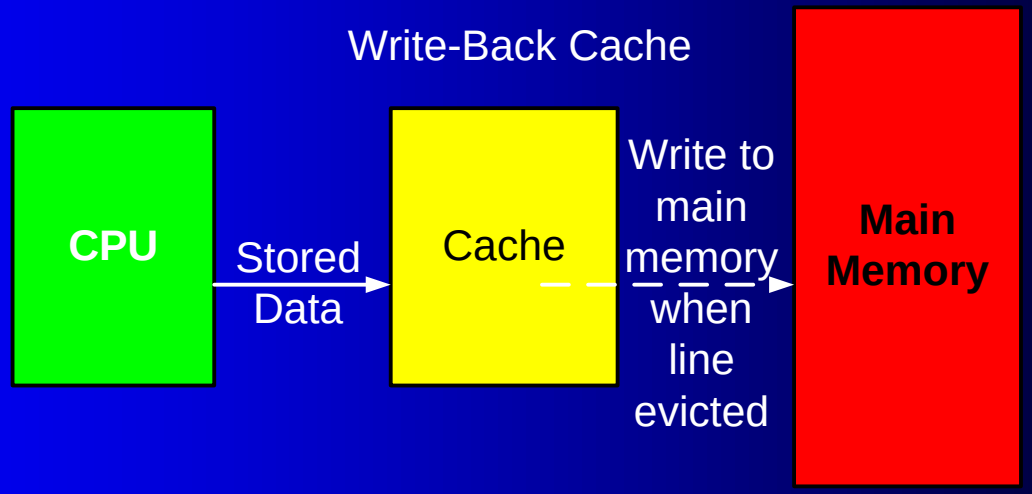


Write-back vs. Write-Through Caches

Метод сквозной записи



Метод обратной записи



Write-Through Cache

Write-through – прежде всего обновляется слово, хранящееся в основной памяти. Если в кэш есть копия слова, то она также обновляется, если копии нет – то из основной памяти в кэш пересылается слово (сквозная с отображением) либо нет (сквозная без отображения).

Главный плюс метода – когда в строка кэш замещается другим блоком – не нужно обновлять память (нет задержки) – она уже обновлена. Прост в реализации.

Но при записи будет потеря времени. Применялся в i80486.

Разновидность – буферизированная сквозная запись (write behind) – в кэш и FIFO, ЦП не ждёт пока FIFO пишет в основную память, т.е. ЦП с ОЗУ не работает вообще.

Write-back Cache

Write-back – слово заносится только в кэш. Если строки в кэш нет, то сначала загрузка строки в кэш, потом её изменение. При замещении строки – её запись в основную память.

Для метода характерно, что при каждом чтении (после определённого этапа) из ОЗУ – две пересылки между ОЗУ и кэш памятью. Какие?

Разновидность – флаговая обратная запись – если в строке изменение, то флаг = 1. При замещении запись в ОЗУ только если флаг=1.

Применялся в МП класса 486 и Pentium от Cyrix.

Сравнения

- Write-back cache в среднем на 10% эффективнее
  - Может записывать сразу всю строку (объединяя несколько операций записи в одну)
  - Использует свойство локальности ссылок
- Write-through cache проще в реализации
  - Не требует флага – была строка записана или же нет
  - Не требует ожидания (пока появится место) на запись строки в ОЗУ при её замещении в кэш
  - Не требует специальных интерфейсов с I/O
- Системы виртуальной памяти предпочитают работать с write-back схемой из-за очень больших накладок записи на диск
  - Имеется тенденция перехода кэш на запись write-back как на более производительную.

Согласование ОЗУ и кэш

Обратная ситуация – устройства ввода/вывода записали новые данные на прямую в память без использования ЦП. Копия данных в кэш – не действительна. Что делать?

Два приёма:

1. Система отслеживает любые изменения ОЗУ и автоматически изменяет соответствующую строку кэш.

2. Прямой доступ к памяти допускается только через кэш (write-allocate).

Write-Allocate vs. Write-no-Allocate

- Write-allocate: Загрузка строки в кэш, затем запись (обновление данных) в кэш
  - Для схемы write-through, запись данных в память будет производиться как есть – без изменения схемы
  - Лучшая производительность, если данные будут затребованы до их замещения в кэш
- Write-no-allocate: Запись в основную память без подгрузки строки в кэш
  - Простая аппаратная реализация процедуры записи
  - Может быть лучше для малых кэш, если записанные данные не должны быть вскоре считаны.

Which makes more sense for writeback/write-through?

Что и как хранить в кэш?

Если любой вид информации (без различия – команды либо данные программы) хранятся в кэш:

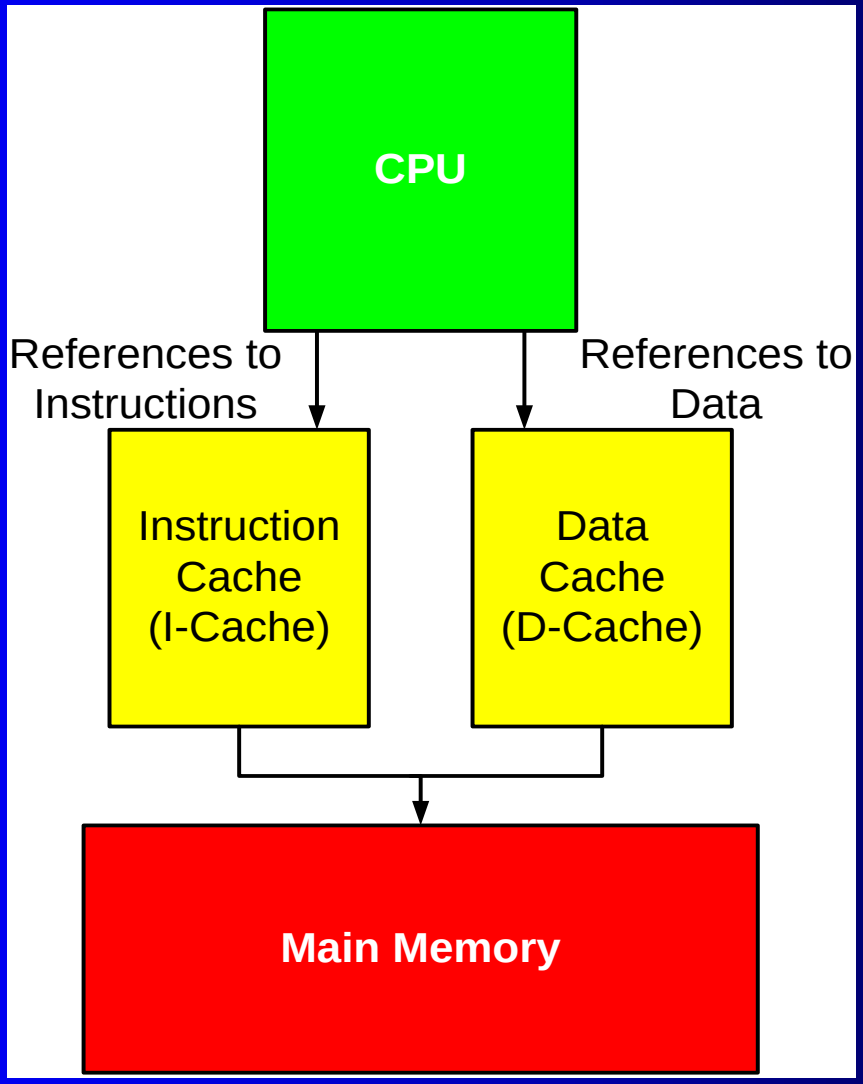
- Это совмещённый (unified) кэш.
- Принстонская архитектура.

В последнее время чаще кэш разделяют – отдельно для команд и отдельно для данных

- Гарвардская архитектура

Совмещённая - проще, при заданной ёмкости – более вероятно попадание -> автоматический баланс между командами и данными, т.е. если в фрагменте программы больше команд, то и кэш будет насыщаться командами…

Instruction and Data Caches



Почему начинает преобладать гарвардская?

- Полоса пропускания: позволяет получать доступ параллельно к командам и данным
- Большинство программ не изменяют собственные команды
  - Самоизменяющиеся программы – слишком сложны, к тому же они значительно повышают нагрузку на оба кэша
- Кэш команд (I-Cache) может быть проще, чем кэш данных D-Cache, при условии, что команды никогда не перезаписываются (изменяются)
- Поток команд обладает высокой локальностью по ссылкам и может обладать более высоким коэф. попаданий на малом кэш
  - Data references never interfere with instruction references

Дисковая кэш

Основная проблема – нерегулярность времени доступа к ячейке информации на ЖД (установить головку считывания/записи, подождать пока уменьшится вибрация, подождать пока нужный сектор окажется под ней).

Дисковая кэш – ОЗУ с произвольным доступом в качестве буфера между дисками и ОП. Обычно от 8 Мб и более. В некоторых системах (UNIX) в качестве дискового ОЗУ – область основной памяти.

Для ДК справедлив принцип локальности- приводит к сокращению времени чтения с 20-30 мс до 2-5 мс, в зависимости от объёма.

В ДК обычно – сквозная запись. Не всю информацию выгодно помещать в кэш, в ряде случаев целесообразно пересылать данные и команды напрямую м/у ОП и диском.

Дисковая кэш

В ДК обычно – сквозная запись. Не всю информацию выгодно помещать в кэш, в ряде случаев целесообразно пересылать данные и команды напрямую м/у ОП и диском.

Имеются механизмы переключения тракта пересылки информации – через кэш или минуя его.

Модель замещения – LRU.

Средства обнаружения и защиты от ошибок.

Архитектура современных НЖМД основана на полностью ассоциативном отображении.

Система ввода-вывода

ВМ помимо ядра содержит многочисленные периферийные устройства (ПУ) и устройство для ввода/вывода данных, для связи между ними. Передача данных от ПУ в память (ядро) ЭВМ – ввод информации, из ядра в ПУ – вывод.

Производительность и эффективность использования ЭВМ определяется составом её ПУ и способом организации их совместной работы. Для обмена данными в ЭВМ используется специальная система ввода/вывода (СВВ).

СВВ обеспечивает обмен данных с помощью сопряжений, которые называются интерфейсами. Интерфейс представляет собой совокупность линий связи, электрических сигналов, …

Система ввода-вывода

Система ввода/вывода должна обеспечивать:

- построение вычислительной системы с переменным составом оборудования
- реализацию параллельной работы процессора над программой и выполнение им процедур ввода/вывода
- простоту и стандартность операций ввода/вывода, обеспечивая независимость программ от особенностей ПУ
- автоматическое распознавание и обслуживание ПУ

Способы обмена:

- программный (не форсированный)
- по прерывания (форсированный)
- при прямом доступе к памяти (аппаратный)

Система ввода-вывода

Программно управляемый обмен данными осуществляется по инициативе процессора и под его управлением. Данные между процессором и памятью, памятью и внешним устройством пересылаются через процессор (регистр AX). При таком обмене процессор на всё время его выполнения отвлекается от других команд – снижается производительность ЭВМ.

Пересылая блок данных ЦП выполняет много вспомогательных действий:

- буферизация данных
- преобразование форматов данных
- подсчёт количества передач
- учёт ограничений по счётчику
- формирование адреса памяти
- обнаружение конечных символов при передаче цепочек

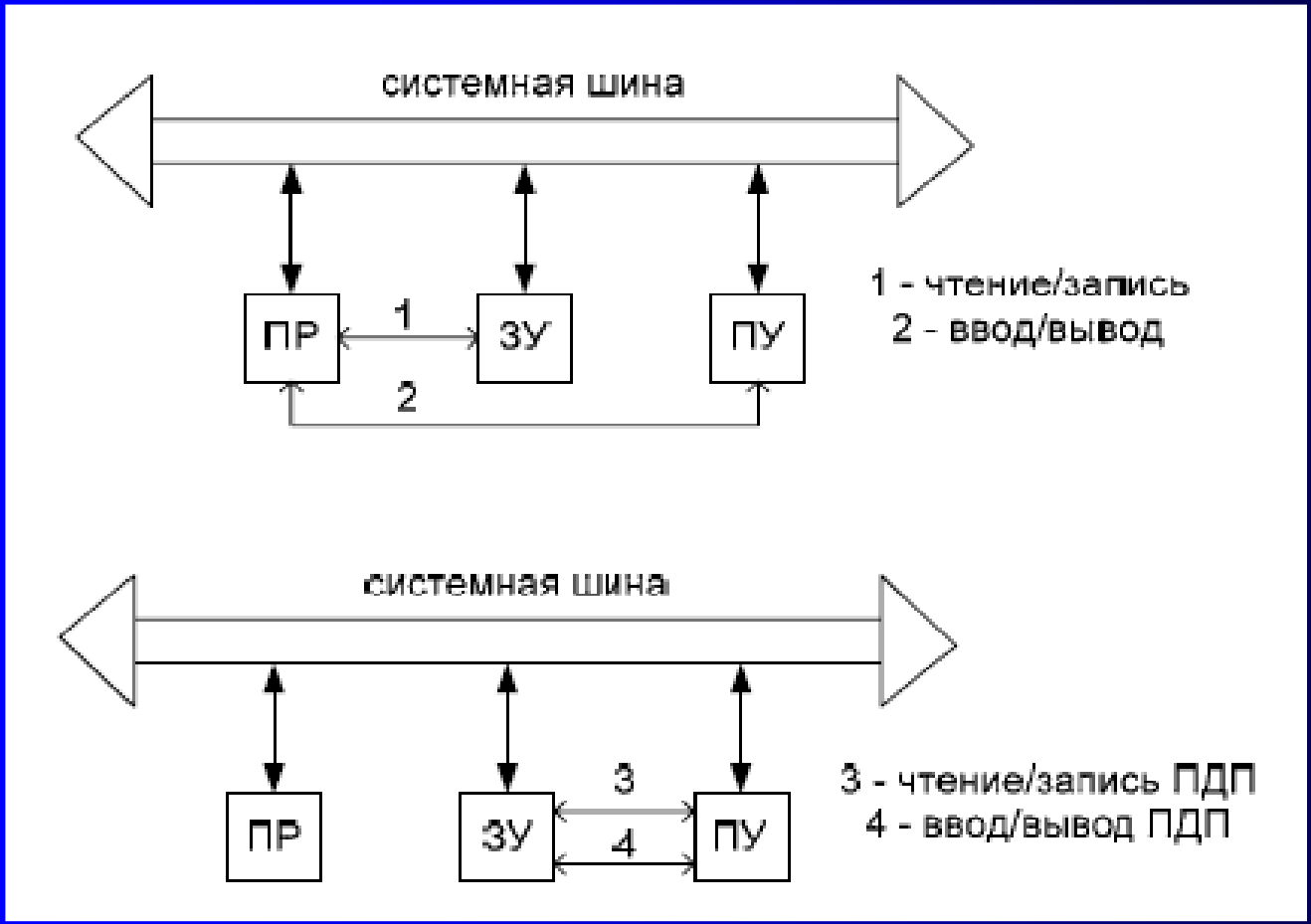
В результате скорость передачи данных снижается в десятки раз.

Система ввода-вывода

Форсировать скорость передачи данных с ПУ можно используя обмен по прерываниям. В этом случае обмен осуществляется по требованию ПУ, когда оно готово к нему, а идентификацию устройства производить по вектору прерывания, переходя соответственно к программе обслуживания ПУ. Это всё повышает скорость передачи, но принцип программно управляемого обмена остаётся неизменным.

Максимально быстрый режим обмена данными осуществляется с помощью ПДП (прямой доступ к памяти). ПДП происходит без участия процессора, данные перекачиваются в ПУ непосредственно из памяти (минуя процессор) и наоборот (аппаратная передача). Этот режим реализуется с помощью дополнительного контроллера ПДП (КПДП).

Система ввода-вывода



Прямой доступ к памяти (мимо процессора)

Изолированный и совмещённый ввод/вывод

Так как адресуемым объектом является ячейка, хранящая информацию, возникает возможность создания единого совмещённого адресного пространства – получается однородно адресуемая система. Такое пространство называется адресным пространством ввода/вывода, отображаемым на память.

Использование совмещённого адресного пространство имеет следующие преимущества:

- для ввода/вывода не нужны специальные команды in/out, все адреса обслуживаются командой mov – меньше команд
- для ввода/вывода не нужны спец сигналы – меньше линий интерфейса
- с регистрами ввода/вывода можно оперировать так же, как и с ячейками памяти;
- можно отвести любой объём памяти под устройства ввода/вывода.