

© Vitas, WWise и др. Большой частью, на основе лекций Василенко в исполнении Dmi (2009 г.) и лекций предыдущих лет.

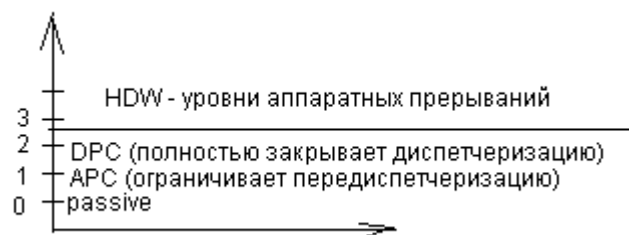
*Ответы на дополнительные вопросы по ПУ. Вопросы (и частично ответы) собраны сдававшими в 2009 году (1-15, 25) и сдававшими раньше*

## **1 DPC/APC? Где они используются? В какой момент драйвер может вызвать DPC?**

**DPC** (Deferred Procedure Call) – механизм отложенного прерывания (процедурного вызова).

Назначение: передать управление на вход драйвера по завершению ввода/вывода, когда контекст ОС доступен и однозначен.

Принцип работы: Контекст ядра однозначно определен в 1 точке: точка входа в ядро. Менеджер DPC находится в этой точке. Когда осуществляется выход из ядра, контекст ядра корректен и в этом месте менеджер анализирует очередь запросов DPC и если она не пуста, он берет первый элемент и передает управление на адрес точки входа в драйвер. Вместе с процедурой DPC связывается глобальный флаг DPC-level, который говорит о том, что система находится в DPC и ни диспетчер, ни планировщик не выполняют никаких действий. Нельзя находиться в DPC долго. Никаких механизмов wait\_for делать нельзя (обращение к виртуальной памяти запрещены, т.к. отказ страницы требует wait\_for). Если прерывание было не в контексте ядра, то менеджер DPC запустится только, когда кто-нибудь запросит kernel.



Используется: При поступлении прерывания о завершении запроса ввода/вывода, действия внутри драйвера не требуются, необходима процедура завершения ввода/вывода (необходимо модифицировать контекст ОС). Драйвер этого сделать не может, так как находится в состоянии прерывания. Действия необходимо перенести на более позднее время, когда будет доступен контекст ОС.

В многопроцессорных системах очередь DPC всегда обслуживается на 1 процессоре.

**APC**, Asynchronous Procedure Call (асинхронный процедурный вызов) – асинхронный способ вызова функции, при котором вызывающая программа продолжает работу, не дожидаясь завершения работы функции.

Используется: При буферизованном вводе/выводе драйверу для обмена предоставляется буфер в пуле ОС и обмен идет с ним. Необходимо данные из пула перенести в буфер прикладной задачи (невозможно в режиме ядра). Однако, находясь в контексте прикладной задачи мы теряем доступ к ОС. Т.е. нужно остаться в контексте ОС, но перейти к прикладной задаче. Нужно поймать попытку перехода в контекст задачи. Для этого используется APC.

Точка входа: точка переключения задачи. Останавливается дальнейший процесс восстановления контекста потока и выполняется процесс завершения процесса обмена. Ловушка APC должна быть привязана к потоку. Заводится указатель на список ловушек APC. В момент установки синхронизации устанавливается ловушка, привязанная к данному потоку в заголовке потока. В момент восстановления потоков диспетчер проверяет очередь. Если очередь не пуста, диспетчер передаёт управление в ловушку APC. Ловушка находится в нулевом кольце защиты в исполнительной системе.

Менеджер ввода-вывода копирует из буфера системы в буфер прикладной задачи. Местонахождение буфера прикладной задачи не определено. После выполнения процесса копирования из системного буфера в буфер прикладной задачи происходит возврат. И так далее, пока очередь не исчерпана. Значит, происходят обращения к диспетчеру памяти. Всё попадает на прикладной уровень. Есть гарантия, что в буфере находятся нужные данные.

## 2 Что такое канал DMA?

Канал – это совокупность ресурсов, зарезервированных для выполнения каких-либо действий.

Канал прямого доступа – это структура включающая в себя характеристики для обращения к памяти: адрес памяти, диапазон регистров и направление (*обмен происходит между ОЗУ и регистрами ВУ*).

- RD – регистр данных.
- R#port – номер порта.
- РАП – регистр адреса в памяти.
- Счётчик байт.
- Регистр состояния.
- Регистр команд – запуск обмена по прямому доступу к памяти после записи всех предыдущих регистров.

Генерация циклов шины: ВУ подключает свой бит Ready на линию DMARq – сигнализирует о готовности порции данных. DMA генерирует циклы шины:

- по переносу данных в Reg данных
- записи данных в приёмник

Reg-счётчик байт уменьшается, адрес увеличивается и KDMA переходит в ожидание. Когда Reg-счётчик байт=0, KDMA генерирует прерывание, поступающее на ЦП, затем в драйвер. По CRC драйвер определяет результат передачи данных (анализ ошибок происходит и в KDMA, и в драйвере ВУ).

## 3 Что такое канал ввода-вывода? Где он находится? Что за информация о контексте потока?

Канал ввода-вывода – это совокупность информации (имя устройства/файла, режим работы) необходимая для работы с данным потоком данных (зарезервированная для выполнения операций ввода/вывода).

Чтобы задать канал ввода-вывода, нужно задать имя устройства. CreateFile берёт строку, options и передаёт подсистеме ввода-вывода. Подсистема ввода-вывода осуществляет поиск в базе данных внешних устройств прав доступа, типа запроса и так далее и перемещает результат запроса в канал ввода-вывода, а прикладной задаче возвращает ID зарезервированного канала.

Контекст, описывающий поток данных, состоит из двух частей:

1. Оперирует понятием прикладной задачи и характеризует данный канал с точки зрения контекста данной задачи. Информацией владеет пользователь подсистемы ввода-вывода прикладной задачи (размещена в контексте задачи).
2. Описывает системные свойства этого потока данных. Она предназначена для исполнительной подсистемы. Информацией владеет операционная система (размещена в контексте ОС).

Система не сможет правильно воспользоваться контекстом при наличии лишь (1). При наличии лишь (2) – дополнительное использование системной памяти и непроизводительные временные затраты.

## 4 Буферизованный и небуферизованный ввод/вывод? В каких случаях используется?

Любой запрос на ввод-вывод имеет буфер(!), но иногда возникает необходимость во втором буфере

Буфер отображается на адреса OS.

Чтобы исключить страничные отказы:

- Выполнять фиксацию буфера в памяти на время выполнения запроса IO, запрет выгрузки этих страниц (для быстрых устройств с большим буфером, т.к. запрет выгрузки большого количества страниц ухудшает эффективность работы системы) – **небуферизованный IO**.
- Для медленных запросов с маленьким буфером заводится область в резидентном пуле – **буферизованный IO**.

## 5 Блокирующий/неблокирующий ввод/вывод.

Способы реализации неблокирующего (?) ввода/вывода:

- создаем новый поток и блокируем его на ввод/вывод (создание потока – накладно, к тому же, нужен стек для него);
- берем его из пула потоков;
- асинхронный ввод/вывод (заводится собственный объект синхронизации, а не в IOSP).

У любого объекта Windows есть заголовок синхронизации, их можно ставить в очередь событий.

Процесс блокируется в ожидании некоторого события, которое ставится в очередь событий.

Выполнив запрос, драйвер говорит, что событие произошло и процесс продолжает работу. Иногда, выдав запрос, нужно параллельно с его обработкой выполнять другие действия. Тогда используется неблокирующий ввод/вывод.

В системах, поддерживающих неблокирующие запросы ввода/вывода, есть модификатор команд: `IO$READ + NoWait`

Чтобы узнать, успешно ли завершён неблокирующий ввод/вывод, в контексте задачи создается буфер IOSB (Input Output Status Block), в который драйвер помещает результаты выполнения операции. Это же происходит и при блокирующей запросах, но все это делается неявно.

Итак, для выполнения операции нужно следующее:

1. Запрос (код операции, параметры)
2. Координаты запроса (FCB, #LBN, длина)
3. Адрес буфера
4. IOSB
5. Объект синхронизации.

Неблокирующий: Один из потоков процесса сделал запрос на ввод-вывод. Блокируется не весь процесс, а только поток, запросивший ввод-вывод. Другой поток продолжает обработку данных до момента, как ему понадобится данные от ввода-вывода. Тогда он ожидает разблокировки потока, запросившего ввод-вывод. В качестве объекта синхронизации выступает поток, запросивший операцию ввода-вывода.

"-": Создание потока – накладно, к тому же, нужен стек для него.

## 6 Отличие обычного цикла шины от цикла прерывания? (очень кривой, даже в вопросе расхождения)

Выставив сигнал Ask после анализа сигнала IRQ, процессор ждёт сообщения номера вектора. В контроллере прерываний есть программируемый регистр базы. Система инициализирует контроллер занятым номером вектора для IRQ0, предполагая, что все остальные векторы идут непрерывно. Таким образом, система может разместить вектора устройств в любом месте непрерывно. Получив, ждёт, пока центральный процессор выставит цикл шины прерываний (без адресной части).

Поэтому в фазе данных по системной шине передаётся номер вектора от КП. Добавляется ещё линия Interrupt. Если Interrupt = 1, то R/W не интересуется. ЦП выставит цикл шины прерываний (без адресной части).

Отличается наличием фазы адреса. Фаза короче. И вот тут надо рассказать к чему приводит отсутствие адреса в цикле шины, т. е. куда же топают запрос ЦП, если нет адреса. По крайней мере, я так понял. Я начал рассказывать про то как выходят из тупика при каскадной обработке прерывания (в лекциях после масс бас), его что-то не воодушевило, правда я рассказывал коряво. Потом начались гипотезы про то, что раз адреса нет то запрос приходит ко всем устройствам (контроллерам), ему тож не понравилось).

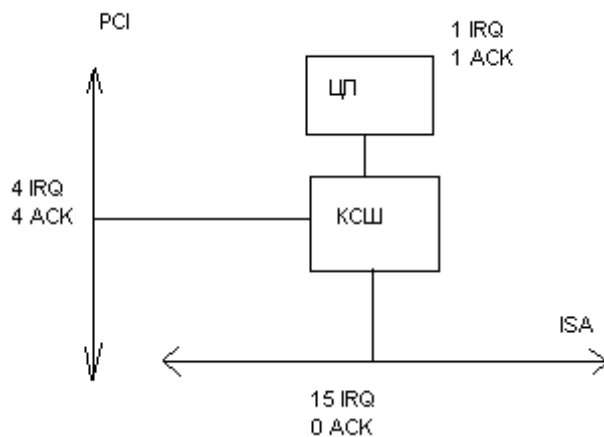
## 7 Гибридная шинная архитектура и недостатки (вроде)?

Архитектура, позволяющая совмещать разные архитектуры прерываний с разными архитектурами системной шины (радиальная – ISA, параллельная – PCI).

ЦП не в состоянии выдавать 4 Ack => нужно отображение параллельной системы на радиальную => нужен специальный узел – *мост*, реагирующий на запросы от параллельной системы, выдающий подтверждение и передающий запрос на ЦП.

К PCI можно подключать устройства Plug'n'Play, система сама динамически настраивает вектора в соответствии с логикой программного контроллера.

- + Можно сделать разные шины, работающие на разной частоте → увеличение скорости
- Большая сложность интерфейса и стоимость ПК



## 8 Недостатки SAN – Storage attached networks (2)?

1. Недостаток такой системы: при реализации PnP за PnP отвечает драйвер SCSI-диска. Поэтому отличить сетевого от локального невозможно. Поэтому для локальных файловых систем этот диск является локальным. Значит, обойти кеширование нельзя, тогда такой диск не может быть применён в режиме реального доступа. Поэтому множественный доступ исключён, так как при одновременном доступе и раздельном КЭШе на этом сетевом диске упадет файловая система. Необходимо гарантировать когерентность КЭШей.
2. Реализация тормозит: Fibre Channel программно и аппаратно не совместим с Ethernet. Поэтому требуется собственная сетевая инфраструктура – iSCSI.

## 9 Когерентность кешей

Если у нас набор процессоров, то у каждого свой кеш, значит нужно заняться проблемой когерентности кешей.

Если когерентность кешей не обеспечена, каждый процесс будет кешировать свой диапазон адресов, или нужно постоянно сбрасывать кеш (*как было сделано до Windows 2000*).

Пока Intel не поддерживал когерентности, невозможно было строить многопроцессорные симметричные архитектуры. Это появилось в Pentium Pro.

### Механизм поддержки когерентности кешей SMEC:

Каждая строка кеша имеет специальные биты состояния, хранящиеся в дескрипторной части строки кеша:

- S – shared
- M – modified
- E – exclusive
- C – cancelled

Симметричная многопроцессорная архитектура объединяется между собой по локальной шине. Она отличается от системной шины реализацией управляющих циклов шины (иначе интерпретируются адреса физической ячейки памяти строки кеша).

Процессоры работают так:

- 1) Кеш сбрасывается.
- 2) Запускается чтение.
- 3) Выставляется физический адрес.
- 4) Процессор, загрузивший ячейку памяти, кеширует её и отмечает как exclusive.
- 5) При любом обращении к памяти любого процессора выставляется физический адрес в памяти.
- 6) При любом обращении к памяти любого процессора выставляется физический адрес памяти.
- 7) Каждый процессор сверяет адрес с кешем.
- 8) Если где-то в другом кеше есть данные, он сообщает, что у него в ячейке последние достоверные значения.

Поэтому он извлекает его из шины поддержки когерентности процессора. Теперь при любом обращении любого процесса к этой строке кеш будет копировать её к себе.

С появлением SМЕС стала возможна симметричная многопроцессорная *архитектура*.

SМЕС используется только для критических секций.

## **10 Плюсы и минусы радиальной системы прерываний?**

+ : реализована наиболее простая системная шина (вся логика обработки прерываний полностью возлагаются на контроллер прерываний, который является программно управляемым, то есть гибкость).

ВУ не надо поддерживать механизм обработки прерываний.

- : количество устройств должно совпадать с количеством линий запросов, изменить это нельзя. Нельзя использовать в мейнфреймах, в сетевых серверах, т.к. они требуют расширяемость.

- : если ЦП генерирует запрос к медленным устройствам ISA, то это тормозит доступ к другим устройствам.

## **11 Отличие синхронных и асинхронных точек входа?**

Синхронные точки входа доступны в произвольный момент. Асинхронные – требуют доступа к аппаратуре, защиты контекста, следовательно, не могут быть реализованы в произвольное время.

У нас есть исполнительная система ОС в виде набора синхронных DLL и асинхронных потоков.

## **12 Типы драйверов, в которых используются асинхронные точки входа?**

Асинхронные точки входа нужны в любых функциональных драйверах, и их может не быть в логических драйверах

## **13 Что нельзя делать в асинхронной точке входа?**

В асинхронной точке входа нельзя делать busy loop и обращаться к страничной памяти.

Обращаться к ВП и производить операции с плавающей запятой (т.к. ожидание).

## **14 Что-то про особенность фильтров нижнего уровня?**

Фильтр нижнего уровня перехватывает запросы функционального драйвера к аппаратуре.

С помощью таких фильтров можно реализовать программный PnP.

Аппаратно PnP работает очень просто: при подключении устройства размыкается контакт и возникает прерывание, которое порождает соответствующие вызовы.

Рассмотрим такой пример: интерфейс USB. Если мы втыкаем обычное устройство напрямую, то все ОК. А вот если воткнуть USB-хаб, а в него уже устройство, как будет работать PnP?

В этом случае реализуется программный PnP. Драйвер-фильтр нижнего уровня опрашивает все разъемы и смотрит, есть ли на них устройство. Если есть, порождает те же вызовы, что и аппаратный PnP.

## 15 Что надо сделать, чтобы в драйвере работал кеш? Как драйвер управляет кешем?

Управлять кешированием можно в устройстве сопряжения с шиной. Один бит запрет/разрешение кеширования чтения, второй бит – отложенное кеширование памяти.

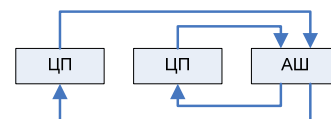
Управлять процессом кеширования надо даже для оперативной памяти при возможности немонопольного доступа к оперативной памяти. Если кешируется буфер ПДП, данные невозможно читать: процессор не осведомлён об изменении содержания буфера.

В некоторых архитектурах необходимо контролировать доступ к регистрам. Для регистров можно это реализовать автоматически, так как для них определены наперёд заданные диапазоны адресов. А для оперативной памяти с виртуальной памятью это невозможно: по физическим адресам нельзя определить природу памяти. При установке бита запрета кеширования запретить его невозможно (возможно только для регистров).

---

## 16 Синхронизация в ОС (с помощью чего и как)

В синхронной и асинхронной шинах для синхронизации ЦП требуется ВУ, которое будет контролировать доступ к разделяемому ресурсу – СШ. Такое устройство называется арбитром шины.



Контроллер прямого доступа имеет наивысший приоритет, т.к. он работает в асинхронном режиме (время получения данных не определено).

ЦП работает в синхронном режиме.

В любом модуле ядра нельзя использовать операции, приводящие к ожиданию на объекте синхронизации. Само ядро управляет объектами синхронизации, следовательно, они сами не могут работать с этими семафорами.

Применение внутриядерных механизмов синхронизации – spin-блокировка. Это процедура XCHG: `xchg AX, [BX]`. Этот механизм spin-блокировки применяется, т.к. определено гарантированное время ожидания.

1. Межпроцессорное ожидание критических ресурсов ядра.
2. (В Windows не реализовано) Если применяется динамическая организация ПДП, для ожидания освобождения канала ПДП.

Спин-блокировки нужны только в многопроцессорных системах. В MSVC не может быть корректно скомпилирован: он (*cl.exe из MSVC, то есть*) заменяет `xchg ax, [bx]` на `nop`. Следовательно, код не будет работать.

Само ядро не является повторно входимым. Следовательно, вы не имеете права использовать в ядре `WaitForObject`. Значит, нельзя применять действия, которые могут явно или неявно к этому приводить. Появление такого ожидания автоматически приводит к краху системы.

Поэтому в ядре нельзя производить операции с виртуальной памятью (проблемы в случае отказа страниц), операции с плавающей запятой.

## 17 Назови плюсы и минусы параллельной системы прерываний.

+: можно к одному каналу IRQ подключать несколько устройств

-: ВУ должно знать вектор прерывания (усложнения устройства КВУ)

PCI – это синхронная шина, поддерживающая параллельную обработку прерываний.

Все устройства, подключенные к системной шине, можно подключить к любой линии запросов.

Таких устройств может быть любое количество.

Центральный процессор анализирует вход. Если есть запрос, он запускает процесс обработки прерываний.

Если несколько, применяется параллельная стратегия обработки. Поэтому система всегда использует только линейную стратегию обработки прерываний.

Фактически эти линии прерываний определяют класс устройств. Низкоскоростные диски – на низком уровне прерываний. Асинхронные – на третьем или четвертом. На более низком уровне RS232, на самом высоком – адаптер.

Метод параллельной (приоритетной) обработки прерываний: Количество устройств не определено. К одному уровню IRQ может подключаться несколько устройств. Контроллера нет, сразу на ЦП. Приоритет скорее всего линейный. Запрос уровнем. Сначала так же. Кому дать сигнал АСК? ЦП выставляет сигнал, на нужный (по приоритету). ВУ выставляет вектор (оно должно знать).

## 18 Почему у драйвера несколько точек входа?

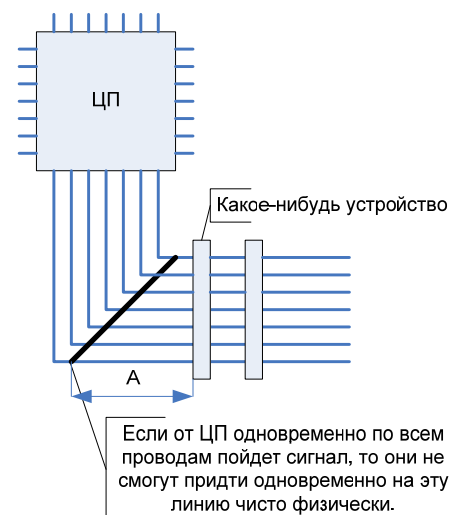
Драйвер не является повторно-входимым (так как устройства ВВ не являются повторно-входимыми). Кроме того, неизвестно время выполнения очередного запроса.

Кол-во точек входа драйвера равно кол-ву параллельных действий, поддерживаемых ОС.

Так как во время его выполнения могут произойти какие-либо системные события, о которых необходимо будет уведомить драйвер (например, сбой питания) – то необходимо несколько точек входа: DriverEntry, IOStrategy, точки входа по отказу по питанию, управлению питанием (смены режима), PnP, обработке DPC, для обслуживания аппаратных прерываний.

## 19 Что такое шина? "какие шины"? И т. д. Тут на много вопросов хватит...

Интерфейс «Системная шина» – открытый интерфейс для подключения устройств. (Интерфейс – это закон перехода границ («по некоторому правилу нечто пересекает эту границу»)).



Все устройства, составляющие ВС, взаимодействует через единый открытый интерфейс (СШ) – это единственный способ взаимодействия узлов ЭВМ. СШ может быть несколько.

СШ – это множество проводов, по которым одновременно передаются сигналы. Это провода надо разместить на материнской плате.

Если сигналы от ЦП отправляются одновременно, то на устройство некоторые сигналы придут с опозданием. Поэтому нужна гарантия, что все сигналы будут в пределах одного временного окна.

Цикл шины – это время, в течение которого осуществляется

взаимодействие двух устройств на данном интерфейсе.

Т.к. шина разделяемый ресурс и возможно взаимодействие нескольких параллельных устройств, то используется специальный сигнал – BUSY.

Сигналы BUSY – определяет момент начала и конец цикла шины.

If (BUSY == 0) СШ свободна; If (BUSY == 1) СШ используется;

Если устройство хочет взаимодействовать, то BUSY устанавливается в единицу. Все устройства постоянно проверяют BUSY. Как только BUSY = 1, то все устройства ждут сообщения о том, с каким устройством хочет взаимодействовать ЦП. ЦП должен это объявить. Как?

Можно пронумеровать все устройства и выставить на шину номер устройства. Но ведь ОП – тоже устройство, но здесь любой байт выступает как отдельное устройство. Поэтому для ЦП нужен не номер устройства, а номер элемента данных устройства.

Вводим Addr. Все устройства, подключенных к шине, имеют диапазон номеров. У любого устройства, подключенного к СШ, имеется специальный узел, который называется дешифратором адреса (обычный компаратор, в котором определен первый элемент и количество элементов). Сравнение. Возможны следующие три варианта:

1. Один компаратор на одном из устройств определил, что адрес попал в его диапазон. Нормальный режим работы.
2. Более 1 устройства определили, что сигнал попадает в их диапазон. ЭВМ работать не будет.
3. Ни 1 устройство не определило, что сигнал попадает в его диапазон. Это приведёт к бесконечному ожиданию. Чтобы такого не происходило, перед началом шины вводится внутренний таймер. Если не появляется Acknowledge, генерируется прерывание по несуществующему адресу.

Все, кто определил, что сигнал не попадает в нужный диапазон адресов – отключаются от СШ и будут анализировать только сигнал BUSY. Когда BUSY станет равным 0, то все опять подключатся к СШ. Только единственное устройство, определившее, что сигнал в его диапазоне, будет работать с СШ для обмена данными. Что хочет инициатор?

- 1) Чтение (передача данных инициатору) – инициатор запрашивает данные по определенному адресу.
- 2) Запись (передача данных от инициатора) – инициатор пишет данные по определенному адресу.

**Асинхронная шина** – это интерфейс СШ, который заранее не определяет время, за которое устройство выполнит обмен информацией. Примеры: PDP/11, UniBus.

**Синхронная шина** – скорости устройств должны попадать в интервалы СШ.

- ФА (фаза адреса) – выставляет на шину для дешифрации адрес.
- ФО (фаза ожидания) – работают и анализируются дешифраторы, ищутся и выставляются на шину данные.
- ФД (фаза данных) – данные выгружаются с шины.

Примеры: ISA, Multiuse, ISE.

Были введены два дополнительных сигнала, т.к. 2 такта ФО для некоторых устройств – это много, а для некоторых – мало.

*Если устройство медленное:* на СШ выставляется сигнал WS (в начале первого такта ФО) и устройство, которое ожидает обмена (например, ЦП) добавляет еще один такт в фазу ожидания, т.к. ФО увеличивается до 3х тактов.

*Если устройство быстрое:* на СШ выставляется сигнал WSO и ЦП убирает один такт из ФО).

Недостаток: цикл шины ограничивается сигналом BUSY. Если в BC 1 ЦП, то это еще нормально, а вот если их будет 4...

На высокочастотных шинах теряется надежность передачи электрических сигналов, т.к. временное окно уменьшается, т.е. надежность шины падает.

**Транзакционная шина.** У Pentium Pro – самая первая такая шина. Поддерживает до 4х ЦП. У каждого из них свой приоритет. Каждый ЦП имеет три фазы:

- ФА (фаза адреса) – выставляет на шину для дешифрации адрес.
- ФО (фаза ожидания) – работают и анализируются дешифраторы, ищутся и выставляются на шину данные.
- ФД (фаза данных) – данные выгружаются с шины.

Когда первый из ФА переходит в ФО, то второй ЦП заходит в фазу ФА. Т.о. полностью используется системная шина.

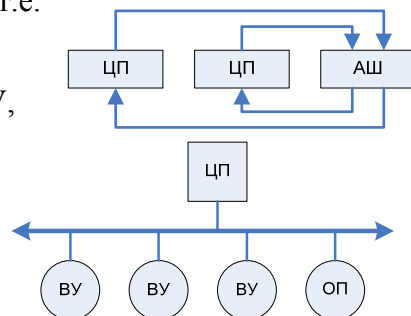
"+": Полностью используется шина

"-": Каждый ЦП должен следить, в какой фазе находятся остальные ЦП, т.е. усложняется архитектура → повышается стоимость.

В синхронной и асинхронной шинах для синхронизации ЦП требуется ВУ, которое будет контролировать доступ к разделяемому ресурсу – СШ.

Такое устройство называется арбитром шины.

Контроллер прямого доступа имеет наивысший приоритет, т.к. он работает в асинхронном режиме (время получения данных не определено). ЦП работает в синхронном режиме.



**Универсальная шина:** Пример: UNIBUS. К СШ подключаются

абсолютно все устройства, включая ОП и ЦП. Шина асинхронная.

Недостаток: если работает с медленным ВУ, то тормозится ОП. → Надо убрать ОП с этой шины.

Появилось 2 шины: СШ (системная шина) и ЛШ (локальная).

1. Медленная шина для ВУ, высокоскоростная шина для ОП

2. Архитектура ЦП не зависит от открытого стандарта СШ. При переходе от одного ЦП к другому меняем ЛШ

Однако, появились устройства, которым перестало хватать скорости СШ. Например, видеоадаптер. Поэтому нужно подключать ВУ отдельно в зависимости от их скоростей. Нужно поделить ВУ на классы.

VESA – взяли на себя труд для подключения высокоскоростной видеопамяти.

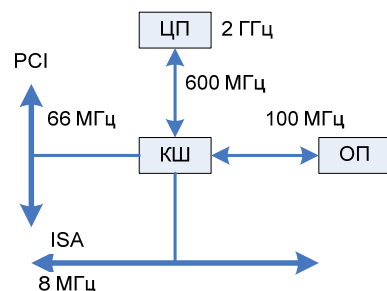
Недостаток: потеряли независимость ПУ (периферийных устройств) от архитектуры ЦП.

**Шина PCI** – это синхронная шина, поддерживающая параллельную обработку прерываний.

Т.к. обмен с шиной идет через КШ PCI, то архитектура ЦП не зависит от PCI. Она поддерживает частоту 33 МГц и механизм Plug and Play.

Недостаток: для организации прерываний и прямого доступа нужен механизм обработки, который можно поместить только в центр схемы → появился узел, через который все ходит. Он начинает тормозить ВС.

В данной схеме КШ выступает в роде моста между различными шинами, преобразуя цикл ЛШ в цикл шины PCI, ОП или ISA. Мост – это преобразователь форматов.



Предположим ЦП обратился к ВУ, который подсоединен к ISA. КШ преобразует цикл шины с 600 МГц на 8 МГц. Пока ВУ не возвратит результат, ЦП блокирует все шины, т.е. получается, что вся система работает с частотой в 8 МГц! Нужно разделить шины, сделав их независимыми.

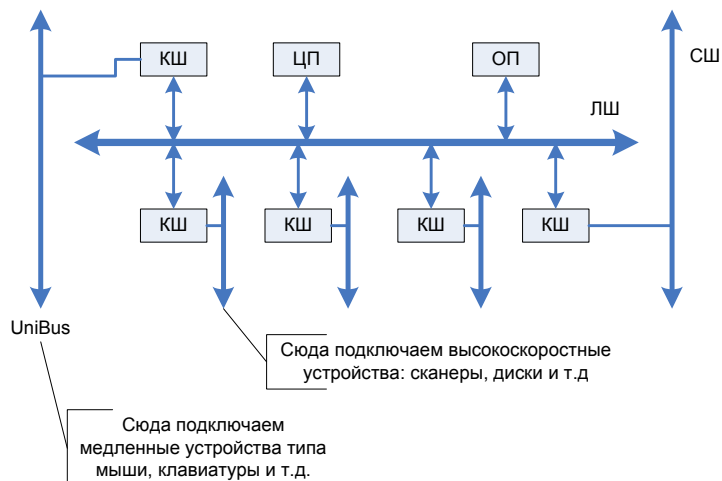
## Пример, Mass Bus.

Все контроллеры шин подключаются к локальной шине (ЛШ).

Преимущество: шины отделены друг от друга, поэтому нет торможения.

Недостаток: нет устройства, которое принимает решение, куда передать запрос.

В данной архитектуре количество шин ограничено и должно быть определено заранее. У каждого КШ есть определенный диапазон адресов. Когда ЦП выставляет адрес на ЛШ, то каждый КШ сверяет: попадает ли адрес на ЛШ в его диапазон, если попадает, то он его обрабатывает. Все, что не попадает в известные отбрасывается на UniBus.



## 20 Назначение SCSI?

SCSI (англ. Small Computer System Interface, произносится скази) — интерфейс, разработанный для объединения на одной шине различных по своему назначению устройств, таких как жёсткие диски, накопители на магнитооптических дисках, приводы CD/DVD, сканеры, принтеры и т. д.

Теоретически возможен выпуск устройства любого типа на шине SCSI.

В компьютерах, совместимых с IBM PC, SCSI не пользуется такой популярностью в связи со своей сложностью и сравнительно высокой стоимостью и применяется преимущественно в серверах.

SCSI широко применяется на серверах, высокопроизводительных рабочих станциях; RAID-массивы на серверах часто строятся на жёстких дисках со SCSI-интерфейсом (однако, в серверах нижнего ценового диапазона всё чаще применяются RAID-массивы на основе SATA). В настоящее время устройства на шине SAS постепенно вытесняют устаревшую шину SCSI.

Система команд SCSI на уровне программного обеспечения употребляется в единых стеках поддержки устройств хранения данных в ряде операционных систем, таких, как Microsoft Windows.

Существует реализация системы команд SCSI поверх оборудования (контроллеров и кабелей) IDE/ATA/SATA, называемая ATAPI - ATA Packet Interface. Все используемые в компьютерной технике подключаемые по IDE/ATA/SATA приводы CD/DVD/Blu-Ray используют эту технологию.

SCSI-шина — это параллельная шина, сервер выполняет функции инициатора обмена. Каждое устройство на шине имеет свой адрес. Внутренние циклы шины с указанием адреса устройства управляют обменом. Контроллер совмещен с ВУ. Сложность интерфейса приводит к возрастанию сложности устройства и к возрастанию стоимости.

+ : возможен параллельный доступ к любому подключенному диску. Интерфейс определяется набором команд, которые имеют свои номера. Первая группа команд: обязательны для каждого устройства, вторая группа: специфична для данного класса устройств, третья группа: специфичны для данного устройства. Каждое устройство на этой шине требует своего драйвера. Можно вклинуться между циклами шины и выполнить некую операцию, драйвер не захватывает доступ к диску в монопольный доступ.

## 21 Про кеш - когда кешируем а когда нет?

В драйвере нельзя кешировать видеоадаптер, сетевой адаптер, ПДП:

1. Видеоадаптер. Его функция: хранение и отображение на экран видеопамати.

Должно с частотой развертки осуществляться сканирование всего экрана. Кэш тупо пробегает по видеопамяти и выводит на экран каждую точку. Для видеоадаптера нужно запретить кэширование, т.е. сквозную запись.

2. Сетевой адаптер. Его функция: передача данных в виде сетевых пакетов в память сетевой платы; плата, получив пакет, отправляет его в сеть. При получении пакета генерируется прерывание, обработчик сетевой карты создает пакет, пакет передается на сетевой адаптер и т.д. Проблема с КЭШ: пришел пакет, его поместили в буфер сетевого кадра; кэш получает следующий пакет, т.е. он не узнает об обновлении данных в буфере, т.к. сетевой адаптер не имеет функции хранения. Для сетевого адаптера нужно запретить кэширование этого диапазона адресов.
3. ПДП. Если кешируется буфер ПДП, данные невозможно читать: процессор не осведомлён об изменении содержания буфера.

Запрос сетевого файлового процессора кешировать нельзя.

ОП. Она не всегда выполняет функцию хранения данных ЦП. Когда осуществляется модификация данных в ОП, неизвестных ЦП – получаем рассогласование данных в КЭШе и ОП.

Для повторения:

Кэш –электрическое устройство для быстрого доступа к памяти. Реализует функции хранения.

КЭШ бывает ассоциативным (эффективный, но не производительный), страничный (производительный, но не эффективный). В настоящее время используется странично-ассоциативный КЭШ.

*В КЭШ попадают все данные*, т.е. кэшированию подлежат все операции без исключения.

Если данные в ОП помечены как некешируемые, то они все равно попадают в КЭШ и после первого обращения к ним помечаются как невалидные, что запрещает их повторное использование.

Кэширование:

- В режиме записи.
  - Сквозная (записываемые данные помещаются в КЭШ и тут же записываются в ОП).
  - Отложенная (запись в ОП кэшируется в КЭШе и не записывается в ОП; запись осуществляется по команде или по истечению определенного промежутка времени).
- В режиме чтения.

Нужно управлять кэшированием (иначе при перепривязке к процессору кэш сбрасывается). Сделать это можно в устройстве сопряжения с шиной. Один бит – запрет/разрешение кэширования чтения, второй бит – отложенное кэширование памяти.

Управлять процессом кэширования надо даже для оперативной памяти при возможности немонопольного доступа к оперативной памяти.

В некоторых архитектурах необходимо контролировать доступ к регистрам. Для регистров можно это реализовать автоматически, так как для них определены наперёд заданные диапазоны адресов. А для оперативной памяти с виртуальной памятью это невозможно: по физическим адресам нельзя определить природу памяти. При установке бита запрета кэширования запретить его невозможно (возможно только для регистров). Это обусловлено обращением к оперативной памяти через шину памяти. К оперативной памяти нельзя обращаться байт за байтом, нужно обращаться за банком.

Обмен данными возможен только порциями, совпадающими с разрядностью шины. Поэтому требуется промежуточная упаковка/распаковка. Это реализует кэш.

Сейчас кэш не для повышения быстродействия, а для промежуточных преобразований.

Без файлового кэша система работает хуже, без страничного кэша не будет работать вовсе.

## 22 В ядре нельзя исп busy loop?

(ядро не реинт не повт входимое -> если кто-то займет его, то другие обломаются)

## 23 Почему RISC быстрее CISC?

CISC, Complex Instruction Set Computer – процессор со сложным набором команд.

RISC, Reduced Instruction Set Computing (technology) – вычисления с сокращённым набором команд. Выкидываются команды, но НЕ которые используются редко, НЕ все, которые можно заменить другими (по типу mul->add). Производятся замены иного рода.

Запрещает использовать методы адресации во всех инструкциях, кроме load и store. Все мнемонические команды остаются (они необходимы всем ЦП).

add R0, R1	команды почти одинаковы с точки
sub R0, R1	зрения внутренней топологии
add R0, R1	эти сильно
add R0, (R1)	отличаются

Запретив команды add R0, (R1) существенно меняем набор команд.

Таким образом, можно максимально сгруппировать команды загрузки из/в память. Еще надо побольше регистров (лучше несколько десятков).

Рост быстродействия RISC колоссален, по сравнению с CISC (если еще использовать кэш для ОП и сгруппировать обращения к ОП вместе).

## 24 Контекст?

**Контекст** – минимальный необходимый объём информации, однозначно описывающий определённый объект.

Следствие: Контекста без объекта не бывает.

Если контекст однозначно определяет объект, то нам объект нужен не всегда. Всегда можно его восстановить.

Контекст ЦП определен только при переходе от одной машинной инструкции к другой.

ОС тоже обладает контекстом, хранится в спец. области – невыгружаемый системный пул (список памяти, список процессов, ...).

ОС – совокупность модулей обработчиков внешних событий. ОС декомпозирована. Связь осуществляется только через системный контекст. Контролировать доступ к системному пулу невозможно.

Контекст ядра хранится в резидентном пуле, контекст сервисов в нерезидентной части.

## 25 Почему вы не сдали лабы? Тогда вы бы ответили с легкостью на все эти вопросы!

Риторический вопрос :-)