

## Лекция 2. Аппаратная и программная поддержка интерфейсов

Основные принципы программирования доступа к периферийным устройствам.  
Методы управления обменом.

Регистровая программная модель периферийного устройства  
Особенности адресации.

Аппаратные средства поддержки работы периферийных устройств:  
контроллеры, адаптеры; мосты.  
Контроль достоверности передачи

# Основные принципы программирования доступа к периферийным устройствам

Разбиение ввода-вывода на несколько уровней, причем нижние – учитывают особенности аппаратуры, верхние - обеспечивают удобный интерфейс для пользователей.

- **независимость от устройств.** Вид программы не должен зависеть от того, читает ли она данные с гибкого диска или с жесткого диска. Единые правила именования ПУ.
- **обработка ошибок как можно ближе к аппаратуре.** Если контроллер обнаруживает ошибку чтения, то он должен попытаться ее скорректировать. Если же это ему не удастся, то исправлением ошибок должен заняться драйвер устройства. Многие ошибки могут исчезать при повторных попытках выполнения операций ввода-вывода, например, ошибки, вызванные наличием пылинок на головках чтения или на диске. И только если нижний уровень не может справиться с ошибкой, он сообщает об ошибке верхнему уровню.
- **использование блокирующих (синхронных) и неблокирующих (асинхронных) передач.** Большинство операций физического ввода-вывода выполняется асинхронно - процессор начинает передачу и переходит на другую работу, пока не наступает прерывание. Пользовательские программы намного легче писать, если операции ввода-вывода блокирующие - после команды READ программа автоматически приостанавливается до тех пор, пока данные не попадут в буфер программы. ОС выполняет операции ввода/вывода асинхронно, но представляет их для пользовательских программ в синхронной форме.

# Основные принципы программирования доступа к периферийным устройствам

- **одни устройства являются разделяемыми, а другие - выделенными.** Диски - это разделяемые устройства, так как одновременный доступ нескольких пользователей к диску не представляет собой проблему. Принтеры - это выделенные устройства, потому что нельзя смешивать строчки, печатаемые различными пользователями. Наличие выделенных устройств создает для операционной системы некоторые проблемы.

Для решения поставленных проблем целесообразно разделить программное обеспечение ввода-вывода на четыре слоя:

Независимый от устройств слой операционной системы.

Обработка прерываний.

Драйверы устройств.

Пользовательский слой программного обеспечения.

# Многоуровневая организация подсистемы ввода-вывода



Пользовательский слой  
программного обеспечения

Независимый от устройств  
слой операционной  
системы.

Драйверы устройств.

Обработка прерываний.

# Обмен данными с ПУ

Программирование доступа к ПУ в общем случае является нетривиальной задачей, даже если не касаться особенностей работы с ПУ, связанных с архитектурой ОС (которая в общем случае реализует виртуализацию ПУ через систему драйверов). Единого интерфейса программирования (API) для работы с ПУ не существует, зачастую даже стандартный интерфейс для определенного типа устройств разрабатывается не сразу.

Ранее разработчики ПО полагались на API, предоставляемый системным BIOS (или BIOS самого устройства), а в сложных случаях прибегали к «ручному» программированию устройства. Однако в многозадачных средах такой подход не работает – требуется обеспечить множественный доступ к одному и тому же устройству. Реализуется это либо программно, через *драйверы*, либо через *интеллектуальный хост-контроллер*, функции которого распределены между «железом» и драйверами.

# Методы управления вводом/выводом

В ВС находят применение три способа организации ввода/вывода:

- программно-управляемый ввод/вывод;
- ввод/вывод по прерываниям;
- прямой доступ к памяти.

# Программно управляемый

Наиболее простым методом обмена является *программно-управляемый доступ (программный доступ, ввод/вывод с опросом)*, или PIO.

*Управляет обменом* (определяет моменты передачи данных, подает адреса и т.д.) *процессор*, чаще всего центральный (но может быть и выделенный процессор ввода-вывода). При этом фактически происходит пересылка данных между регистрами процессора и регистрами/памятью ПУ (или контроллера интерфейса).

# Программа управления вводом - выводом

ЦП выполняет программу, которая обеспечивает прямое управление процессом ввода/вывода, включая проверку состояния устройства, выдачу команд ввода или вывода. Выдав на шину ВВ команду, центральный процессор должен ожидать завершения ее выполнения.

В этой программе:

ЦП с помощью команды ввода/вывода сообщает модулю ввода/вывода, а через него и ПУ о предстоящей операции. Адрес модуля и ПУ, к которому производится обращение, указывается в адресной части команды ввода или вывода.

Модуль исполняет затребованное действие, после чего устанавливает в единицу соответствующий бит в своем регистре состояния. Ничего другого, чтобы уведомить ЦП, модуль не предпринимает.

Для определения момента завершения операции или пересылки очередного элемента блока данных процессор должен периодически опрашивать и анализировать содержимое регистра состояния шины (канала, магистрали) ВВ.



# Команды ввода-вывода

Данные читаются пословно. Для каждого читаемого слова ЦП должен оставаться в *цикле проверки*, пока не определит, что слово находится в регистре данных (РД) модуля ВВ, то есть доступно для считывания.

Существуют четыре типа команд В/ВЫВ, которые может получить модуль ВВ: управление, проверка, чтение и запись.

*Команды управления* используются для активизации ПУ и указания требуемой операции. Например, в устройство памяти на магнитной ленте может быть выдана команда перемотки или продвижения на одну запись. Для каждого типа ПУ характерны специфичные для него команды управления.

*Команда проверки* применяется для проверки различных ситуаций, возникающих в модуле ВВ и ПУ в процессе ввода/вывода: включено ли ПУ, готово ли оно к работе, завершена ли последняя операция ввода/вывода и не возникли ли в ходе ее выполнения какие-либо ошибки. Действие команды сводится к установке или сбросу соответствующих разрядов регистра состояния модуля ВВ.

*Команда чтения* побуждает модуль получить элемент данных из ПУ и занести его в РД. ЦП может получить этот элемент данных, запросив модуль ВВ поместить его на шину данных.

*Команда записи* заставляет модуль принять элемент данных (байт или слово) с шины данных и переслать его в РД с последующей передачей в ПУ.

# Плюсы и минусы PIO

Преимущество PIO – в простоте аппаратной реализации ПУ.

Требуется обеспечить лишь выставление на шину / чтение с шины содержимого регистров или ячеек памяти по сигналу доступа.

Недостаток – в низком быстродействии и необходимости задействовать процессор, который в общем случае будет простаивать ввиду более высокого быстродействия по сравнению с ПУ.

# Метод прямого доступа к памяти

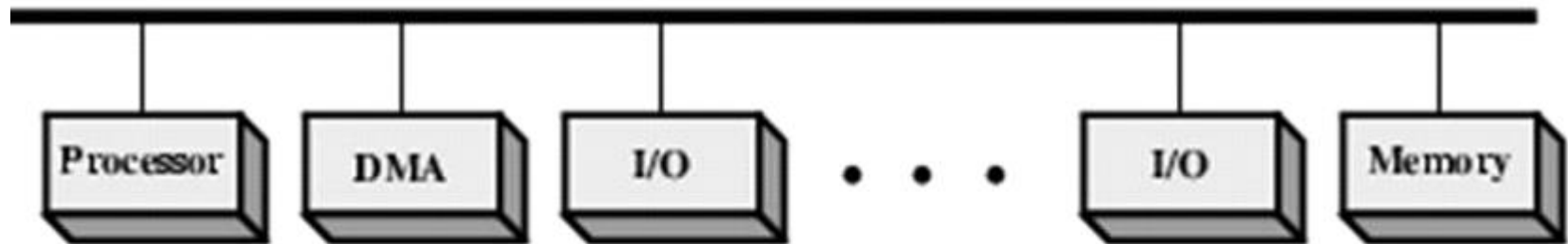
Метод *прямого доступа к памяти* (ПДП, DMA) позволяет выполнять обмен между ОП системы и ресурсами ПУ асинхронно по отношению к вычислительному процессу.

В этом режиме обмен данными между ПУ и основной памятью компьютера происходит без участия процессора.

Обменом в режиме ПДП управляет не программа, выполняемая процессором, а электронные схемы, внешние по отношению к процессору. Обычно схемы, управляющие обменом в режиме ПДП, размещаются в специальном контроллере, который называется *контроллером прямого доступа к памяти*

Контроллер DMA может быть как общесистемным, так и входить в состав ПУ. Его требуется запрограммировать на пересылку данных между двумя адресатами, после чего он самостоятельно вырабатывает сигналы передачи данных.

# Схема 1 подключения ПДП

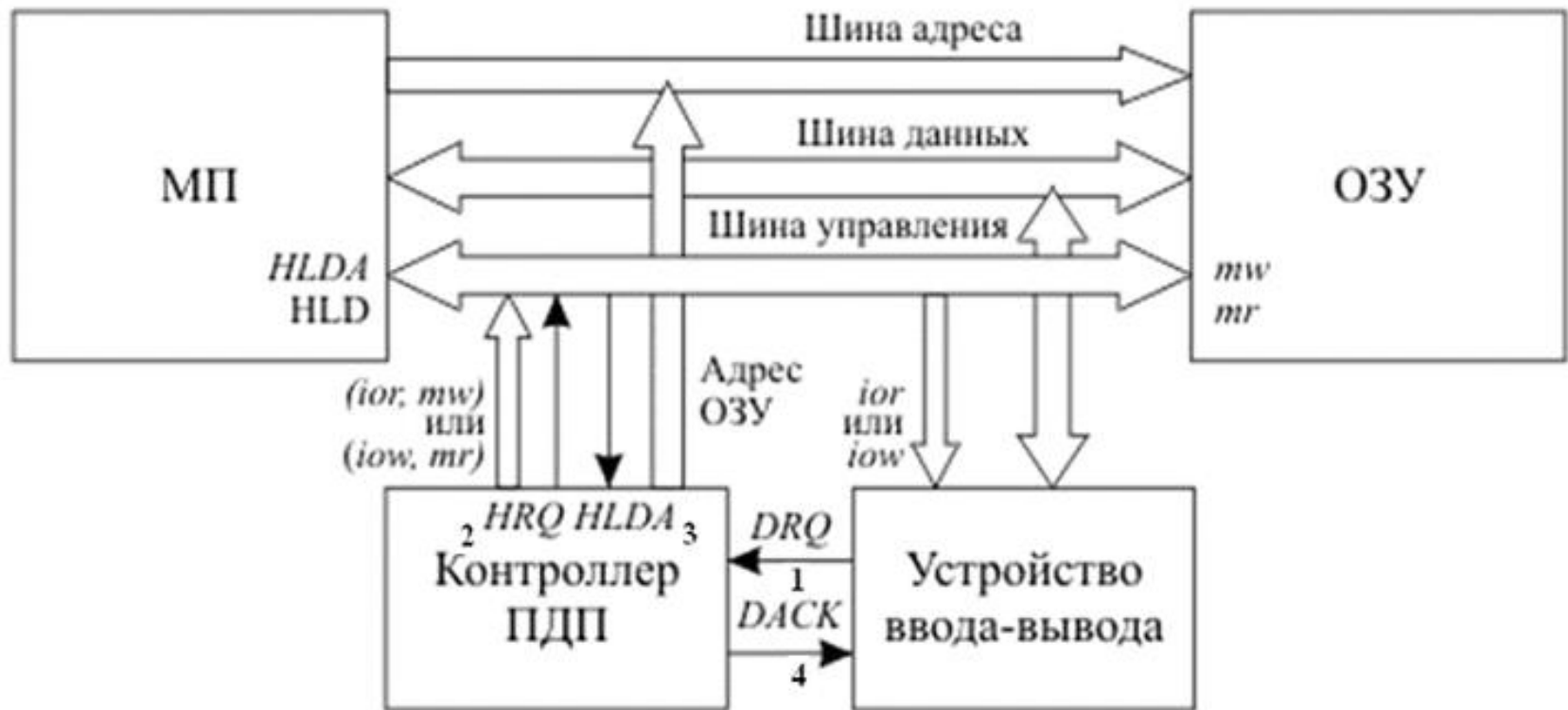


К общей шине подключается контроллер DMA и все вовлеченные в операцию устройства.

В каждой транзакции шина используется дважды  
1) I/O к DMA;            2) DMA к ОЗУ.

При передаче по шине CPU дважды приостанавливает свою работу.

# Структура ЭВМ с КПДП



Обмен данными в режиме ПДП

Перед началом работы **контроллер ПДП** необходимо **инициализировать**: занести начальный адрес области ОП, с которой производится обмен, и длину передаваемого массива данных. В дальнейшем по сигналу запроса прямого доступа контроллер фактически выполняет все те действия, которые обеспечивал микропроцессор при *программно-управляемой передаче*.

Последовательность действий *КПДП* при запросе на *прямой доступ к памяти* со стороны устройства ввода-вывода следующая:

- 1) Принять** запрос на ПДП (сигнал DRQ) от УВВ.
- 2) Сформировать** запрос к МП на захват шин (сигнал HRQ).
- 3) Принять** сигнал от МП (HLDA), подтверждающий факт перевода микропроцессором своих шин в третье состояние.



- 4) **Сформировать** сигнал, сообщаящий устройству ввода-вывода о начале выполнения циклов *прямого доступа к памяти* (DACK).
- 5) **Сформировать** на шине адреса компьютера адрес ячейки памяти, предназначенной для обмена.
- 6) **Выработать** сигналы, обеспечивающие управление обменом (IOR, MW для передачи данных из УВВ в оперативную память и IOW, MR для передачи данных из оперативной памяти в УВВ).
- 7) **Уменьшить** значение в счетчике данных на длину переданных данных.
- 8) **Проверить** условие окончания сеанса прямого доступа (обнуление счетчика данных или снятие сигнала запроса на ПДП). Если условие окончания не выполнено, то изменить адрес в регистре текущего адреса на длину переданных данных и повторить шаги 5-8.

# Регистры контроллера

Контроллер ПДП имеет в своём составе такие элементы:

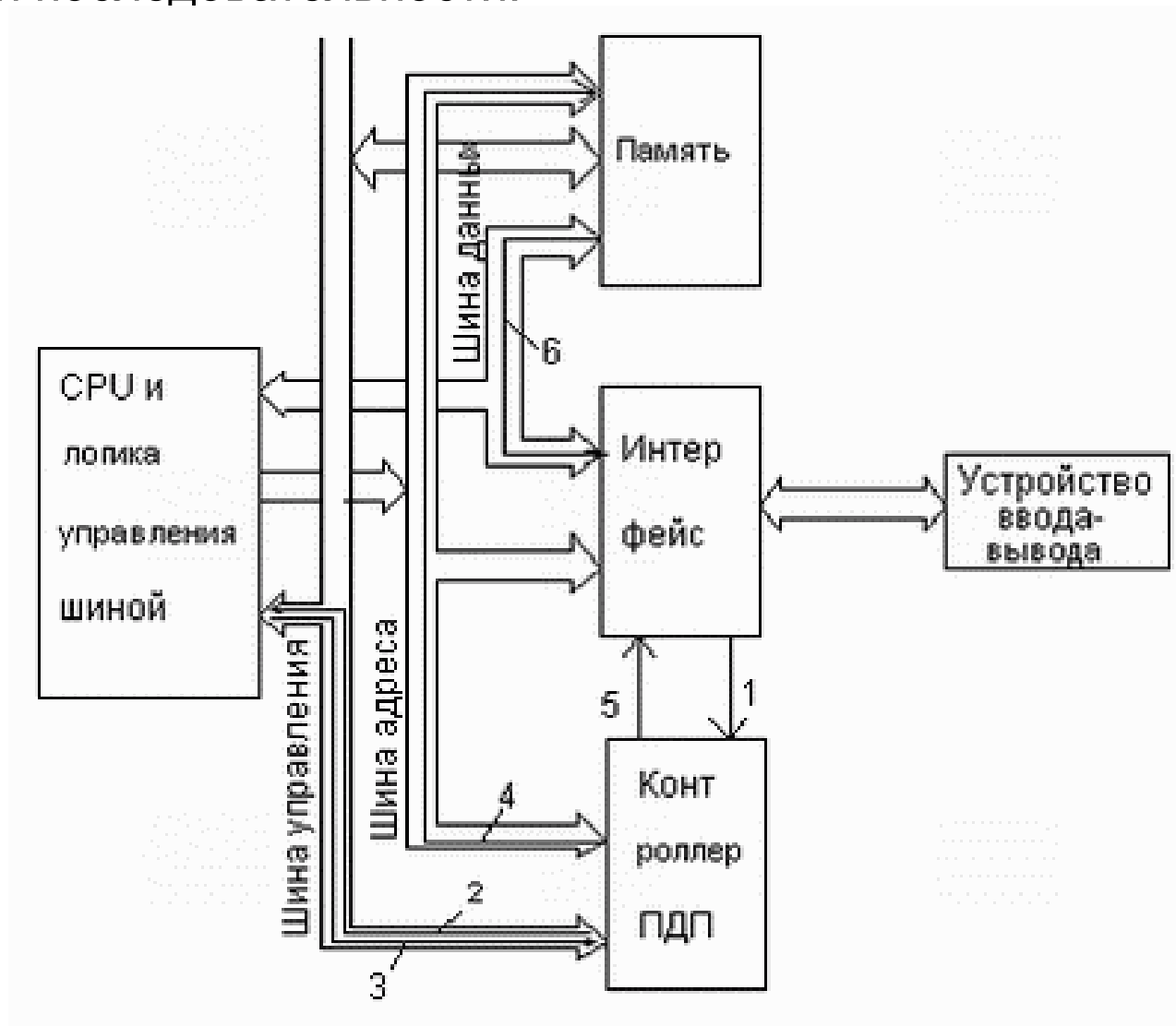
- регистры управления и состояния;
- регистр адреса;
- регистр-счётчик байт;

Общая организация контроллера ПДП приведена на рисунке.





**Работа контроллера прямого доступа памяти происходит в такой последовательности:**




1. Интерфейс устройства ввода–вывода при наличии у него готовых данных для оперативной памяти или готовности принять данные от оперативной памяти, выдаёт сигнал запроса ПДП.
2. Контроллер ПДП формирует сигнал запроса шины и получает управление шиной от центрального процессора.
3. На шину адреса помещается адрес ячейки памяти, который контроллер ПДП выдаёт из своего регистра адреса.
4. Контроллер ПДП выдаёт подтверждение интерфейсу устройства ввода–вывода на ввод или вывод данных.
5. Данные помещаются на шину данных, при этом при выводе данных, данные поступают из оперативной памяти, а при вводе—данные поступают от интерфейса устройства ввода–вывода.
6. Данные, в случае вывода фиксируются интерфейсом устройства ввода–вывода, а в случае ввода, данные поступают в ячейку памяти, адрес которой помещён на шине адреса.
7. Контроллер освобождает шину и управление шиной возвращается центральному процессору.
8. Если происходил ввод данных в оперативную память, то счётчик байт вычитает из своего значения «1», а регистр адреса прибавляет «1», тем самым указывая на следующую свободную ячейку памяти. Если счётчик байт не содержит «0», то алгоритм повторяется, в противном случае передача данных останавливается.

# Канал ввода/вывода

Поскольку в современной компьютерной системе устройств ввода–вывода несколько, то для каждого устройства в составе контроллера ПДП организуется канал, в состав которого входят перечисленные выше регистры. Назначение регистров также видно из их названия.

*Регистр управления* предназначен для хранения и выдачи на шину сигналов, которые управляют и информируют устройства, задействованные в обмене. Эти сигналы указывают устройствам о направлении передачи: ввод или вывод, между какими устройствами, так как контроллер способен управлять и передачами типа память–память, о непрерывности режима ПДП или с перерывами передачи, собственно разрешении режима ПДП, приоритетах каналов.

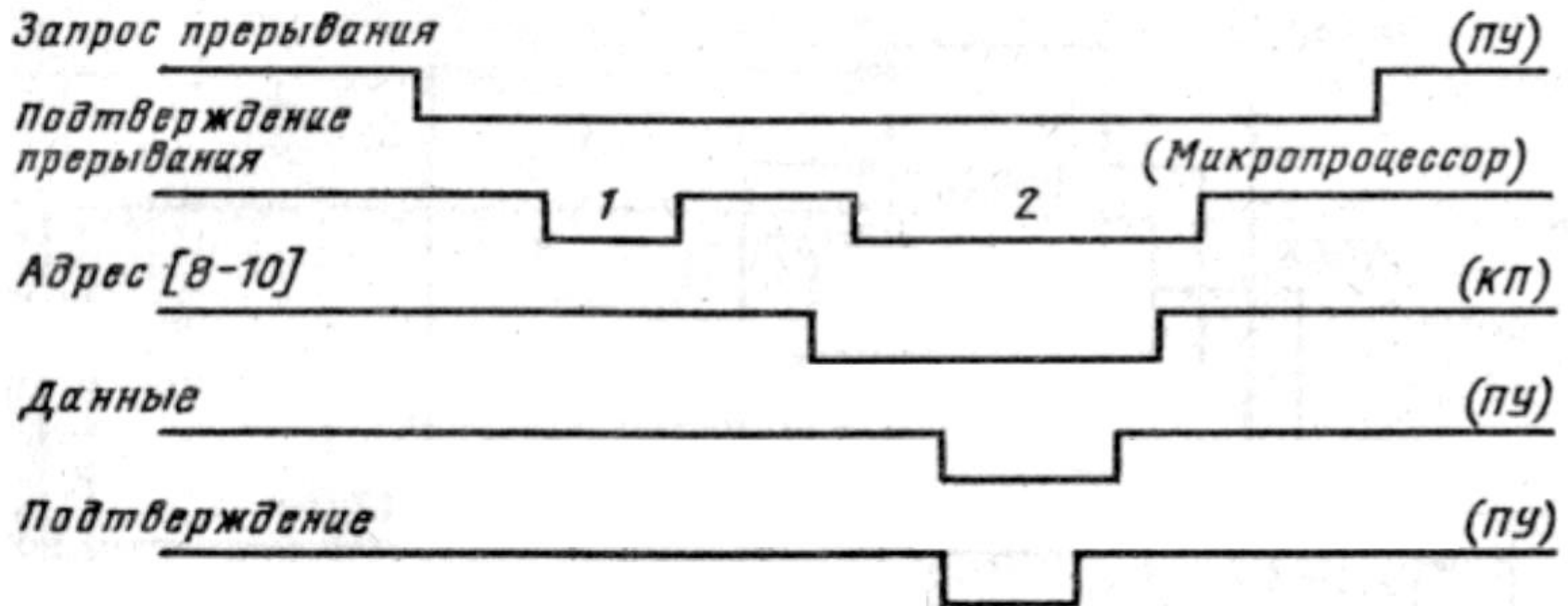


*Регистр состояния* служит для указания состояния контроллера ПДП, а именно: счётчиков каналов и наличия или отсутствия запросов ПДП.

В *регистр адреса* при инициализации контроллера передаётся начальный или конечный адрес массива памяти в который помещаются данные, а в регистр–счётчик байт число передаваемых байт. После каждого переданного байта счётчик уменьшает первоначальное значение числа передаваемых байт. Таким образом контролируется процесс передачи данных между памятью и устройствами ввода–вывода, при достижении в счётчике байт «нуля» режим работы ПДП завершается, контроллер ПДП возвращает управление шиной процессору. А затем, в зависимости от ситуации, устройства либо снова начинают процесс инициализации режима ПДП, либо управление осуществляет центральный процессор.

# По прерыванию

ЦП выдает команду В/ВЫВ, а затем продолжает делать другую полезную работу. Когда ПУ готово к обмену данными, оно через модуль ВВ извещает об этом процессор с помощью запроса на прерывание. ЦП осуществляет передачу очередного элемента данных, после чего возобновляет выполнение прерванной программы.



# PIC, APIC

**Программируемый контроллер прерываний** (Programmable Interrupt Controller, PIC) отвечает за приём запросов прерываний от различных устройств, их хранение в ожидании обработки, выделение наиболее приоритетного из одновременно присутствующих запросов и выдачу его вектора в процессор, когда последний пожелает обработать прерывание. Слово «программируемый» в названии контроллера означает, что режимы его работы устанавливаются программно, а не являются жёстко «зашитыми».

**APIC ( *Advanced Programmable Interrupt Controller* )** — улучшенный ...  
Он был добавлен в процессоре Pentium.

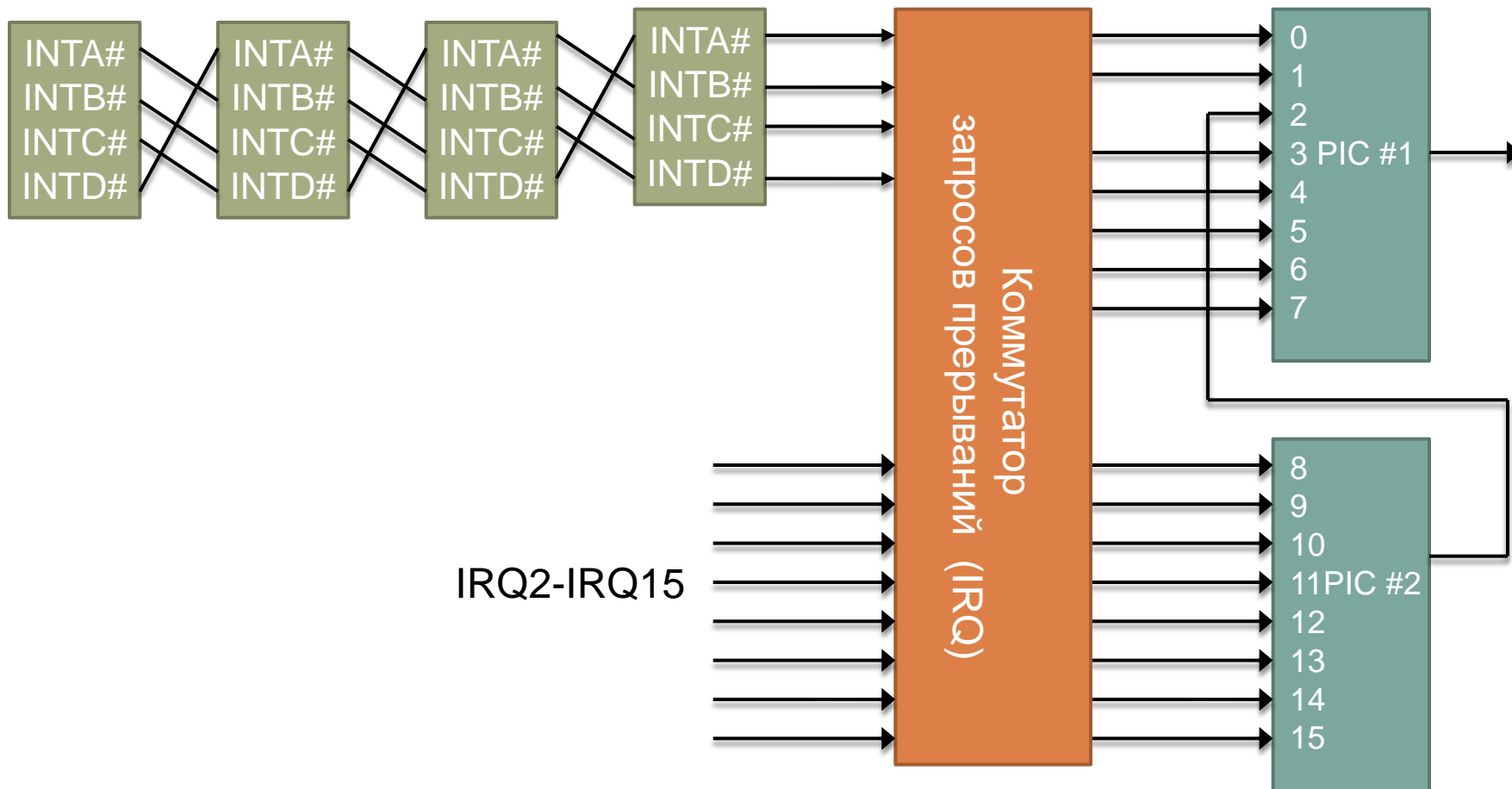
Преимущества расширенного контроллера прерываний:

возможность реализации межпроцессорных прерываний — сигналов от одного процессора другому

поддержка до 256 входов IRQ, в отличие от 15 на классической IBM PC

крайне быстрый доступ к регистрам текущего приоритета прерывания и подтверждения прерывания. Контроллер прерываний, совместимый с IBM PC, исполнялся как устройство шины ISA с очень медленным доступом к его регистрам (порт 0x20). I/O APIC — контроллер, расположенный на системной плате, обычно как часть микросхем обрaмления процессора (например, микросхема Intel 82489DX).

# Схема подачи сигналов INT#



# Сигнализация по линии INTx#

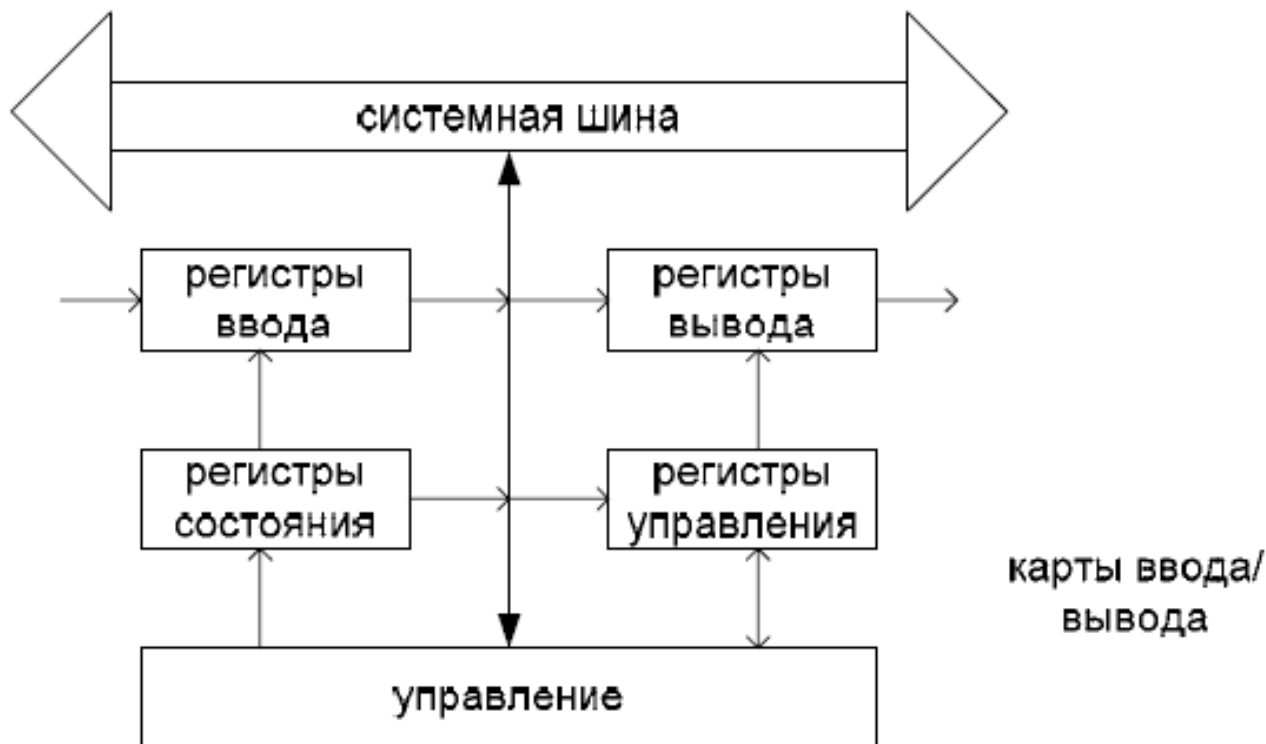
1. Устройство вводит сигнал прерывания, понижая уровень линии INTx#.
2. ЦП получает сигнал прерывания с вектором, соответствующем определенной линии IRQ.
3. Обработчик прерывания (драйвер) обращается к устройству и проверяет, установлен ли в его регистрах сигнал запроса прерывания.
4. Если это было именно его устройство, драйвер сбрасывает сигнал прерывания программным способом и начинает обработку.
5. После отработки прерывания линия запроса все еще может быть в низком уровне из-за прихода прерывания от другого устройства, разделяющего ту же линию – тогда процедура повторяется.



# Программные интерфейсы ПУ

Разработчики новых устройств зачастую создают собственные программные модели и интерфейсы программирования, что приводит к проблемам совместимости с прикладным и системным ПО. Тем не менее, для целого ряда современных устройств разработаны стандартные интерфейсы программирования. В особенности это касается универсальных внешних интерфейсов.

# Состав ПУ



В состав ПУ входят следующие узлы:

- 1) регистр ввода для приема входных данных от МП;
- 2) регистр вывода для выдачи данных из МП в ПУ;
- 3) регистр состояния ПУ;
- 4) регистр управления (регистр режима);
- 5) схема управления и селекции ПУ;

# Регистровая модель ПУ

Изначально разработчики придерживались *регистровой программной модели* ПУ. Устройство представлялось программно доступным (в общем пространстве портов ввода-вывода) набором регистров, среди которых обязательно были три – состояния, управления и данных (т.н. модель CSD). Доступ предполагался методом PIO.

Адресное пространство ввода/вывода может быть совмещено с адресным пространством памяти или быть выделенным.

По мере усложнения архитектуры и повышения требований к устройствам и интерфейсам появилась необходимость реализации более сложной многоуровневой модели программирования с применением объектно-ориентированного подхода.

Современные интерфейсы программирования устройств включают не только аппаратные, но и программные компоненты, входящие в состав ядра операционной системы. Программисту приходится иметь дело не с регистрами, а с системными объектами, а всю низкоуровневую работу с аппаратными ресурсами выполняет драйвер со стандартным интерфейсом программирования.

# Система ввода-вывода, понятие и задачи. Совмещенное адресное пространство СВВ, достоинства и недостатки. Выделенное (изолированное) адресное пространство СВВ, достоинства и недостатки

**Модульность.** ВС проектируются на основе модульного (или агрегатного) принципа.

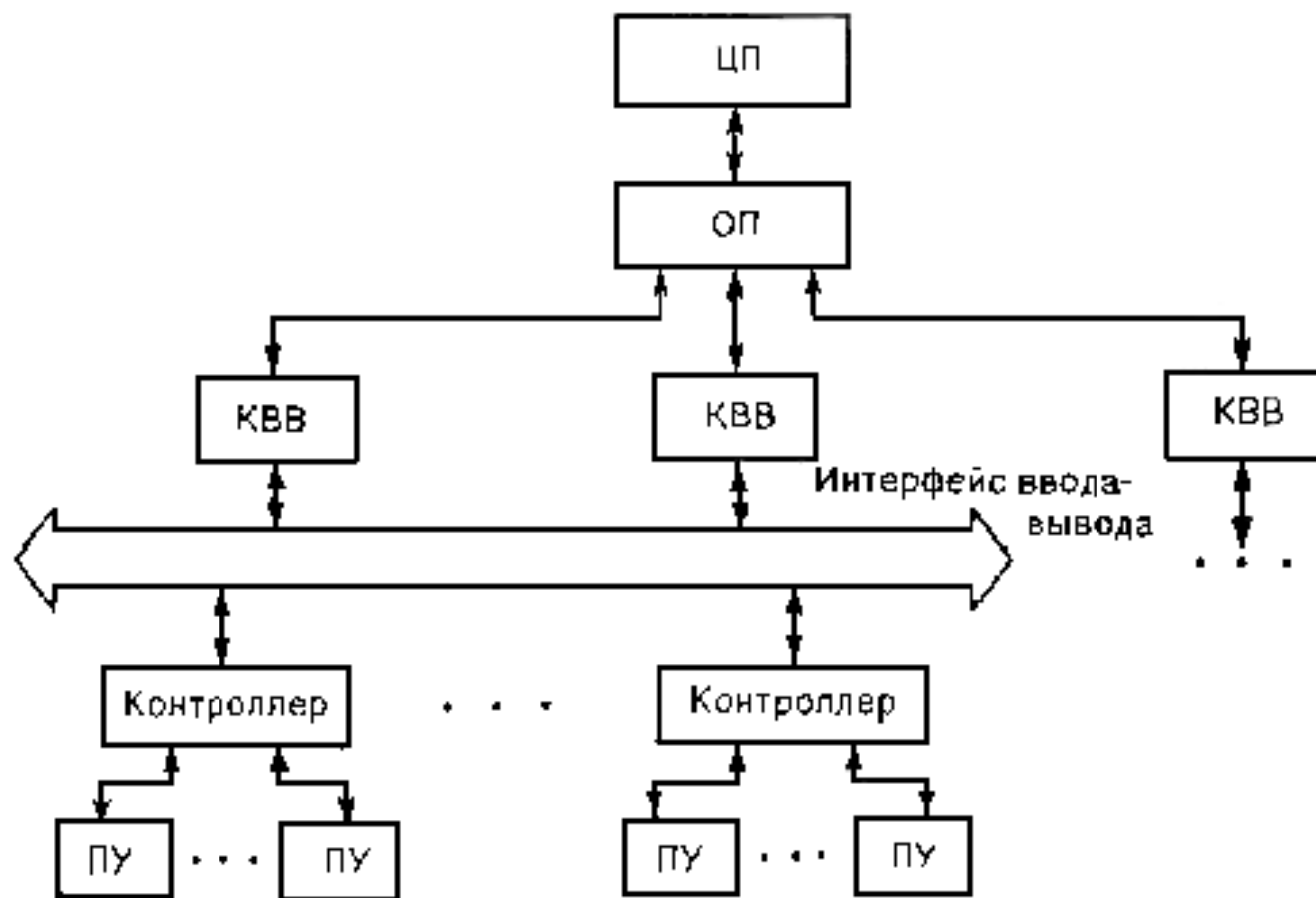
Он заключается в том, что отдельные устройства выполняются в виде конструктивно законченных модулей (агрегатов), которые могут сравнительно просто в нужных количествах и номенклатуре объединяться, образуя ЭВМ.

**Унифицированные** (не зависящие от типа ПУ) **форматы данных**, которыми ПУ обмениваются с ядром ЭВМ, в том числе и унифицированный формат сообщения, которое периферийное устройство посылает в ядро о своем состоянии. Преобразование в индивидуальные форматы данных осуществляют контроллеры и адаптеры.

**Унифицированный интерфейс**, т.е. унифицированный по составу и назначению набор линий и шин, унифицированные схемы подключения, сигналы и алгоритмы (протоколы) управления обменом информацией между ПУ и ядром ЭВМ.

**Унифицированные** (не зависящие от типа ПУ) **формат и выбор команд** процессора для операций ввода-вывода. Операция ввода-вывода с любым ПУ представляет для процессора просто операцию передачи данных независимо от особенностей принципа действия данного ПУ, типа его носителя и т.п..

# Каналы ввода-вывода



# Основные функции канала

- прием команд управления работой канала из центрального процессора;
- адресация внешнего устройства, указанного в принятой команде;
- выполнение действий, заданных в командах;
- установка управляющих сигналов на шинах интерфейса ввода-вывода;
- прием управляющих сигналов, поступающих от ПУ по шинам интерфейса;
- непосредственная передача информации между оперативной памятью и ПУ;
- контроль передаваемой информации на четность;
- подсчет количества передаваемых байт информации;
- прием и анализ информации о состоянии периферийных устройств;
- формирование запросов в центральный процессор на прерывание;
- управление последовательностью прерываний от ПУ и выполнение прерывания.

# Модули ввода/вывода. Функции модуля

Модуль ввода/вывода в составе вычислительной машины отвечает за управление одним или несколькими ПУ и за обмен данными между этими устройствами с одной стороны, и основной памятью или регистрами ЦП — с другой.

Основные функции модуля:

- локализация данных;
- управление и синхронизация;
- обмен информацией;
- буферизация данных;
- обнаружение ошибок.

# Контроллеры и адаптеры

**Контроллеры и адаптеры** представляют собой устройства, выполняющие роль своеобразного переходника соединяющего компьютер с каким либо периферийным устройством.

С помощью **адаптеров и контроллеров можно связать несколько внешних электронных устройств** и успешно управлять их работой. Для работы определенной группы устройств существует своя группа переходников или плат расширения. Посредством этих устройств происходит обмен информационными потоками сообщаемыми компьютеру о текущем состоянии электроники и происходящих в ней сбоях.

Традиционно выделяют две самостоятельных группы адаптеров и контроллеров:

Встраиваемые устройства, располагаются внутри системного блока. Это и есть так называемые платы расширения. Подключаются такие устройства благодаря, имеющимся на материнской плате, слотам.

Выносные устройства, подключаемые посредством имеющихся в компьютере интерфейсов. Например, широко используются USB интерфейсы.



# Назначение и принцип работы контроллера

Контроллер, как правило, представляет собой электронное устройство, иногда выполненное как часть самого процессора или же сложных микросхем его обрaмления, входы которого присоединены электрически к соответствующим выходам различных устройств. Номер входа контроллера прерываний обозначается «IRQ». Следует отличать этот номер от приоритета прерывания, а также от номера входа в таблицу векторов прерываний (INT). Так, например, в IBM PC в реальном режиме работы (в этом режиме работает MS-DOS) процессора прерывание от стандартной клавиатуры использует IRQ 1 и INT 9.

# Контроллер портов ввода-вывода

Одним из контроллеров, которые присутствуют во всех компьютерах, является контроллер портов ввода-вывода.

Типы портов:

параллельные (LPT1-LPT4), к ним обычно присоединяют принтеры и сканеры;

последовательные асинхронные порты (COM1-COM4), к ним подсоединяются мышь, модем и т. д.;

игровой порт - для подключения джойстика;

порт USB (USB 2, 3), к нему подключаются новые модели принтеров, сканеров, модемов, мониторов и т.д. Одним из его достоинств является возможность подключения целой цепочки устройств. Например, через один порт USB подключен принтер, через принтер подключен сканер и т.д.

Некоторые устройства могут подключать и к параллельным, и к последовательным портам, и к порту USB.

# Контроллеры

**Контроллер** - предназначен для соединения различных устройств(видеокамера, DVD плеер, переносные жесткие диски, звуковые карты).

**Контроллер USB** - позволяет оснастить системный блок дополнительными или отсутствующими портами USB для соединения с устройствами поддерживающими этот интерфейс (принтеры, сканеры, мыши, клавиатуры, модемы, игровые устройства)

**Контроллер SATA** - позволяет подключить накопитель с интерфейсом SATA к материнской плате на которой отсутствует данный разъем или как расширение для добавления дополнительных жестких дисков. Контроллер устанавливается как в слот PCI так и PCI-ex.

**Контроллер IDE на SATA** - позволяет подключить накопитель с IDE в SATA разъем.

# Интеллектуальный хост-контроллер

Современные контроллеры интерфейсов снабжены *интеллектуальным хост-контроллером* – устройством, обеспечивающим более гибкое управление процессом обмена данными. В частности, такой хост-контроллер самостоятельно обрабатывает списки задач, формируемые в памяти системы, не требуя от процессора контроля за состоянием ПУ

# Назначение и принцип работы адаптера

Адаптер - устройство сопряжения ЭВМ (ПЭВМ) и ПУ.

Средство сопряжения различных устройств ЭВМ в том числе использующих различные способы представления данных. В коммутационных, соединительных и кабельных устройствах (также – “переходник”) - элемент с разнотипными разъемами, служащий для соединения разнотипных штекеров и/или гнезд, либо для подключения их к телекоммуникационным розеткам различных размеров и типов, изменения разводки или числа проводов в разъемах, соединения разнотипных кабелей.

## **Основное назначение адаптера контроллера.**

1. Подключение внешних запоминающих устройств.
2. Контроль за работой определенного устройства.
3. Передача информации от одного ПК к другому.
4. Преобразование и согласование сигналов между устройствами ПК

# Адаптеры

**Разветвители USB** - для расширения количества USB портов, например из 1 в 4. Разветвители имеют самые различные формы.

**Адаптер DVI->VGA** - позволяет подключить устройство которое не имеет DVI входа, а только VGA. Наиболее часто требуется подключить монитор не оснащенный DVI разъемом к видеокарте у которой в наличии только DVI.

**Адаптер DVI->HDMI** - для подключения видеокарты (компьютера) к ЖК или плазменному телевизору или другому устройству поддерживающими данный тип разъема.

## **Адаптер**

- для подсоединения USB клавиатуры или мыши в разъем PS/2.

**Адаптер PS/2->USB** - для обратного подсоединения PS/2 устройства к USB разъему. Очень часто данный вариант соединения отказывается работать.

# Видеоадаптеры

Видеоадаптер - устройство, преобразующее графический образ, хранящийся как содержимое памяти компьютера (или самого адаптера), в форму, пригодную для дальнейшего вывода на экран монитора. Первые мониторы, построенные на электронно-лучевых трубках, работали по телевизионному принципу сканирования экрана электронным лучом, и для отображения требовался видеосигнал, генерируемый видеокартой.

Назначение: преобразование цифрового сигнала, циркулирующего внутри РС, в аналоговые электрические сигналы, подаваемые на монитор. Другими словами, видеоадаптер выполняет роль интерфейса между компьютером и устройством отображения информации (монитором).

Типы: MDA, CGA, HGC, EGA, VGA, SuperVGA, IBM 8514

Режимы работы: Графический режим, Текстовый режим.

# Назначение и принцип работы моста

Мост - устройство передачи данных, соединяющее две и более компьютерные шины, например мост PCI-PCI. Основная функция моста - передать информацию от основной шины к дополнительной.



# Контроль достоверности передачи

Для повышения достоверности и качества передачи применяются групповые методы защиты от ошибок, избыточное кодирование и системы с обратной связью.

Существует три наиболее распространенных орудия борьбы с ошибками в процессе передачи данных:

- коды обнаружения ошибок;

- коды с коррекцией ошибок, называемые также схемами прямой коррекции ошибок (Forward Error Correction - FEC);

- протоколы с автоматическим запросом повторной передачи (Automatic Repeat Request - ARQ).

Рассмотрим основные механизмы обеспечения целостности передаваемой информации с помощью введения избыточности.

# Механизм контрольных сумм

Контрольная сумма — некоторое значение, рассчитанное по набору данных путём применения определённого алгоритма и используемое для проверки целостности данных при их передаче или хранении. Под контрольную сумму может быть выделена некоторая совокупность  $n$  бит (один бит, полубайт, байт, несколько байт). Контрольной суммой некоторого массива будет являться величина, полученная путем деления с остатком суммы всех элементов массива на максимально возможное числовое значение контрольной суммы, увеличенное на единицу ( $2^n$ ). Значение контрольной суммы добавляется в конец блока данных непосредственно перед началом передачи или записи данных на какой-либо носитель информации. Впоследствии оно проверяется для подтверждения целостности данных. Вероятность ошибки равна  $1/2^n$ .

Функция, реализующая алгоритм и выполняющая преобразование, называется «хеш-функцией» или «функцией свёртки». Важная отличительная особенность хороших алгоритмов хэширования заключается в том, что генерируемые с его помощью значения настолько уникальны и трудноповторимы, что задача нахождения коллизий, т.е. ситуаций, когда разным наборам входных блоков данных соответствует один хэш.

# Механизм четности (parity).

Для передаваемых данных формируется специальный сигнал – признак нечетного количества единиц на линиях. В случае обнаружения нарушения четности в фазе данных сигнал ошибки вырабатывает приемник, для фазы адреса проверку четности выполняет целевое устройство, выставляется соответствующий бит в регистре состояния. Этот метод ограничен определением изменения состояния одиночного бита в байте. В случае изменения состояния двух битов, возможна ситуация, когда вычисление паритетного бита совпадет с записанным. В этом случае система не определит ошибку, и произойдет экстренная остановка системы. Так как приблизительно 90% всех нерегулярных ошибок происходит именно с одиночным разрядом, проверки четности бывает достаточно для большинства ситуаций.

Недостаток этих способов контроля достоверности в том, что алгоритмы суммирования в них слишком просты. Для формирования надежной контрольной суммы можно увеличивать количество бит под контрольную сумму, а также необходим такой алгоритм расчета, когда каждый новый байт может оказать влияние на любые биты контрольной суммы.

Проверка на четность может осуществляться по нескольким битам. В этом случае определяется четность или нечетность целого блока символов.

# Код CRC


Методы обнаружения ошибок предназначены для выявления повреждений сообщений при их передаче по зашумленным каналам (вносящих эти ошибки). Для этого передающее устройство создает некоторое число, называемое контрольной суммой и являющееся функцией сообщения, и добавляет его к этому сообщению. Приемное устройство, используя тот же самый алгоритм, рассчитывает контрольную сумму принятого сообщения и сравнивает ее с переданным значением.

Например, если для расчета контрольной суммы используем простое сложение байтов сообщения по модулю 256, то может возникнуть примерно следующая ситуация.

Сообщение : 6 23 4

Сообщение с контрольной суммой : 6 23 4 33

Сообщение после передачи : 6 27 4 33




Недостаток этого алгоритма в том, что он слишком прост. Если произойдет несколько искажений, то в 1 случае из 256 не сможем их обнаружить. Например:

Сообщение : 6 23 4

Сообщение с контрольной суммой : 6 23 4 33

Сообщение после передачи : 8 20 5 33


Для повышения надежности можно было бы изменить размер регистра с 8 битного на 16 битный (то есть суммировать по модулю 65536 вместо модуля 256), что скорее всего снизит вероятность ошибки с  $1/256$  до  $1/65536$ .



Таким образом, сформулировано 2 требования для формирования надежной контрольной суммы:

**Ширина:** Размер регистра для вычислений должен обеспечивать изначальную низкую вероятность ошибки (например, 32 байтный регистр обеспечивает вероятность ошибки  $1/2^{32}$ ).

**Случайность:** Необходим такой алгоритм расчета, когда каждый новый байт может оказать влияние на любые биты регистра.



Контрольная сумма (Checksum) – Число, которое является функцией некоторого сообщения. Буквальная интерпретация данного слова указывает на то, что выполняется простое суммирование байтов сообщения, что, по видимому, и делалось в ранних реализациях расчетов. Однако, на сегодняшний момент, несмотря на использование более сложных схем, данный термин все имеет широкое применение.

Если сложение, очевидно, не пригодно для формирования эффективной контрольной суммы, то таким действием вполне может оказаться деление при условии, что делитель имеет ширину регистра контрольной суммы.

Основная идея алгоритма CRC состоит в представлении сообщения виде огромного двоичного числа, делении его на другое фиксированное двоичное число и использовании остатка этого деления в качестве контрольной суммы. Получив сообщение, приемник может выполнить аналогичное действие и сравнить полученный остаток с "контрольной суммой" (переданным остатком).

# Циклический избыточный код

(*cyclic redundancy check* - CRC). Основная идея алгоритма CRC состоит в представлении сообщения в виде одного двоичного числа, делении его на другое фиксированное двоичное число и использовании остатка этого деления в качестве контрольной суммы. Получив сообщение, приемник может выполнить аналогичное действие и сравнить полученный остаток с "контрольной суммой" (переданным остатком).

В основе метода CRC лежит понятие *полинома* или *многочлена*. Каждый бит некоторого блока данных соответствует одному из коэффициентов двоичного полинома. Например, полином двоичного числа 11111010 будет выглядеть следующим образом:

$$A(x) = 1 \cdot x^7 + 0 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 0 \cdot x^0 = x^7 + x^5 + x^4 + x^2 + x.$$

Передаваемый блок данных представляется в виде двоичного полинома  $A(x)$ . Для вычисления контрольного кода необходим еще один полином  $G(x)$ , называемый *порождающим* полиномом. Для каждой реализации алгоритма контроля CRC порождающий полином выбирается заранее. Например, для контроллеров гибких магнитных дисков порождающий полином  $G(x) = x^{16} + x^{12} + x^5 + 1$ . Полином  $R(x)$  называется контрольным кодом полинома  $A(x)$  при порождающем полиноме  $G(x)$ , если  $R(x)$  является остатком от деления полинома  $A(x) \cdot x^r$  на  $G(x)$ , где  $r$  – степень полинома  $G(x)$ :  $R(x) = (A(x) \cdot x^r) \bmod G(x)$





Полином (Polynomial) – Полином является делителем CRC алгоритма.

Степень (Width) – Степень (или ширина) CRC алгоритма соответствует степени используемого полинома (или длине полинома минус единица). Например, если используется полином 11010, то степень алгоритма равна 4. Обычно используется степень, кратная 8

# Циклический избыточный код

Так же, как и для контрольных сумм, контрольный код не занимает много места (обычно 16/32 бита), однако вероятность обнаружения ошибки существенно выше. Например, в отличие от контрольных сумм метод CRC сможет обнаружить перестановку двух байт либо добавление единицы к одному и вычитание единицы из другого.

Алгоритм широко используется в аппаратных устройствах (дисковые контроллеры, сетевые адаптеры и др.) для верификации неизменности входной и выходной информации, а также во многих программных продуктах для выявления ошибок при передаче данных по каналам связи (в частности, CRC8, CRC16, CRC32, где 8, 16 и 32 обозначают разрядность кода) применяется для проверки целостности передачи данных.

Такая контрольная сумма проста в реализации и обеспечивает низкую вероятность возникновения коллизий.

# Коррекция ошибок (код исправления ошибок).

Этот метод включает определение ошибки не только в одиночном разряде, но и двух, трех и четырех разрядах. Кроме того ЕСС может также исправлять ошибку в одиночном разряде.

Во время считывания результат ЕСС-слова сравнивается с рассчитанным, подобно тому, как происходит в описанных выше методах проверки контрольных сумм. Основное различие состоит в том, что в проверке четности каждый бит связан с одним байтом, в то время как ЕСС-слово связано с совокупностью байт. Если четность корректна, то для всех групп, то это свидетельствует об отсутствии ошибок. Если одна одно или более значение четности для переданного блока не верно, генерируется уникальный код, называемый синдромом, по которому можно идентифицировать переданный с ошибкой бит.

Примером корректирующих кодов являются код Хемминга и код Рида-Соломона.

# Код Хемминга

В коде Хемминга вводится понятие *кодového расстояния*  $d$  (расстояния между двумя кодами), равного числу разрядов с неодинаковыми значениями. Возможности исправления ошибок связаны с минимальным кодовым расстоянием  $d_{min}$ . Исправляются ошибки кратности  $r = \lfloor (d_{min}-1)/2 \rfloor$  и обнаруживаются ошибки кратности  $d_{min}-1$  (здесь  $\lfloor \rfloor$  означает “целая часть”). Так, при контроле на нечетность  $d_{min} = 2$  и обнаруживаются одиночные ошибки. В коде Хемминга  $d_{min} = 3$ . Дополнительно к информационным разрядам вводится  $L = \log_2 K$  избыточных контролирующих разрядов, где  $K$  - число информационных разрядов,  $L$  округляется до ближайшего большего целого значения.  $L$ -разрядный контролирующий код есть инвертированный результат поразрядного сложения (т.е. сложения по модулю 2) номеров тех информационных разрядов, значения которых равны 1.

Определяется четность или нечетность целого блока символов. Каждый бит четности вычисляется для определенной комбинации бит данных.

# Коррекция ошибок (код исправления ошибок).

Пример 1. Пусть имеем основной код 100110, т.е.  $K = 6$ . Следовательно,  $L = 3$  и дополнительный код равен  $010 \# 011 \# 110 = 111$ , где  $\#$  - символ операции поразрядного сложения, и после инвертирования имеем 000. Теперь вместе с основным кодом будет передан и дополнительный. На приемном конце вновь рассчитывается дополнительный код и сравнивается с переданным. Фиксируется код сравнения (поразрядная операция отрицания равнозначности), и если он отличен от нуля, то его значение есть номер ошибочно принятого разряда основного кода. Так, если принят код 100010, то рассчитанный в приемнике дополнительный код равен инверсии от  $010 \# 110 = 100$ , т.е. 011, что означает ошибку в 3-м разряде.

# Коды Рида-Соломона

*Коды Рида – Соломона* позволяют исправлять ошибки в блоках данных. Элементами кодового вектора являются не биты, а группы битов (блоки, в частности, байты)ю

Кодирование информационного блока может быть выполнено разными способами, при этом исходный блок будет либо дополнен некоторым проверочным блоком, либо перекодирован, из которого затем получаются полученный информационный и проверочный блоки.

Декодировщик последовательно вычисляет синдромы ошибок, строит соответствующий полином ошибок и находит корни данного полинома (они определяют положение искаженных символов в кодовом слове), по которым определяет характер ошибок и исправляет их.

Устранение ошибок с помощью корректирующих кодов (такое управление называют Forward Error Control) реализуют в симплексных каналах связи. В дуплексных каналах достаточно применения кодов, обнаруживающих ошибки (Feedback or Backward Error Control), так как сигнализация об ошибке вызывает повторную передачу от источника. Это основные методы, используемые в информационных сетях.