1. Классификация ввода-вывода (программно-управляемый, прямой доступ к памяти)

Само устройство ввода-вывода состоит из двух частей: контроллера и внешнего устройства. Обычно контроллер находится на плате, которая вставляется в свободный разъём, но есть и исключения: контроллеры некоторых устройств, являющихся неотъемлемой частью ПК (например, клавиатура), располагаются прямо на плате. Таким образом, к шине ВУ подключается контроллер, а ВУ подключается через периферийную шину к контроллеру. Так почему бы не подключить ВУ напрямую? Это сделать можно, но потребление ресурсов ПК будет неэффективным из-за простоя процессора.
Обмен информацией с внешним устройством состоит из инициации обмена, где осуществляются предварительные действия (установка соединения) по подготовке к вводу или выводу данных и обмену данными. Если и инициализацией, и обменом занимается центральный процессор, то такой обмен называется программно-управляемым. Главный минус подобной стратегии ввода-вывода: процессор простаивает в ожидании готовности ВУ. Процессор будет простаивать до тех пор, пока ВУ не подаст сигнал о готовности передачи информации. Преимущества — легкость реализации.
Чтобы исключить периодическую проверку готовности, устройства могут сами инициировать обмен по специальному аппаратному сигналу, который называется запросом прерывания, а соответствующий обмен — управляемым прерываниями ввод-вывод. При таком способе внешнее устройство сигнализирует процессору о необходимости начать обмен, процессор приостанавливает (прерывает) текущую программу, осуществляет ввод-вывод с помощью программы обработки прерывания, а затем продолжает выполнять основную программу. Такой подход с точки зрения производительности более правильный, но он требует больших усилий на обработку прерываний. В БЭВМ можно либо разрешить все прерывания, либо все запретить.
Ввод-вывод с использованием прямого доступа к памяти (ПДП, в английской литературе DMA) организует инициализацию и обмен данными при помощи контроллеров ПДП. Такие контроллеры передают данные непосредственно в память ЭВМ, при этом центральный процессор в обмене данными не участвует. Такой способ не нагружает процессор, следовательно, намного быстрее. Но для реализации данного подхода требуются более сложные контроллеры, что значительно удорожает конструкцию.
В БЭВМ реализованы программно-управляемый ввод-вывод и управляемый прерываниями. Программно-управляемый ввод-вывод достигается с помощью ожидания непосредственно в аккумуляторе бита статуса. Управляемый прерываниями достигается с помощью системы прерываний. Прямой доступ к памяти не реализован в БЭВМ, так как нет соответствующих схем в контроллере ВУ.

Контроллеры ВУ связаны с процессором при помощи системной шины БЭВМ, в сегменты (или, физически, разъемы для установки контроллеров) которой подсоединяются различные шины со стороны контроллера: шина данных, адреса, сигнал чтения/записи, готовности, прерывания и тд. Со стороны процессора: дешифратор приказа, регистр прерывания от КВУ, регистр разрешения прерывания, логика управления шины БЭВМ (для подключения датчиков сигнала КВУ и CPU для обмена данными в один момент времени).

2. Синхронный и асинхронный обмен (сравнение, реализуем ли в БЭВМ)

Программно-управляемый обмен по способу инициации разделяется на синхронный, когда обмен начинается в заранее известный промежуток времени (например, каждую минуту) и асинхронный, когда программе неизвестно время начала обмена данными, и она вынуждена периодически проверять возможность обмена (в БЭВМ готовность определяется по регистру статуса в каждом ВУ).
Обмен данными (прием и передача) может также быть организован синхронно, когда наличие данных на шине подтверждается специальным сигналом синхронизации с постоянной частотой, и асинхронно, с использованием сигналов готовности и/или подтверждения приема-передачи данных. Задачу инициации и обмена данными в ЭВМ осуществляют специальные программы (такие программы еще называют драйверами), которые совместно с аппаратурой ЭВМ организуют и контролируют процесс ввода-вывода. Драйвер знает принципы работы устройства. Синхронный режим передачи данных в БЭВМ вроде как возможен с помощью ВУ-0 - таймера (однако Афанасьев говорит, что нет). Таймер устанавливает готовность на раз в 100·DR миллисекунд.
Завершение обмена и получение драйвером результата также может быть синхронным/асинхронным (синхронное, то есть после окончания обмена сразу приходит сигнал об окончании обмена; асинхронное - значит оповещение об окончании обмена приходит непонятно когда), но это зависит от ОСи.

3. Прямой доступ к памяти (управляемый аппаратурой)

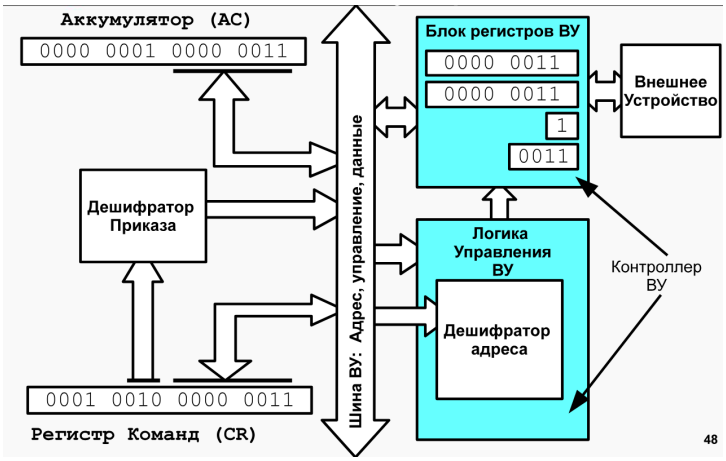
Управляющим устройством является контроллер. Когда УУ занято чем-то другим (расчётами), контроллер берёт на себя функцию устройства, которое инициирует обмен с памятью, и данные, минуя процессор (обычно есть контроллер, который имеет доступ к памяти), пытаются попасть в память. Драйверы в это время устанавливают режим работы

только между контроллером и памятью. Такой способ не нагружает процессор, следовательно, намного быстрее. Но для реализации данного подхода требуются более сложные контроллеры, что значительно удорожает конструкцию.

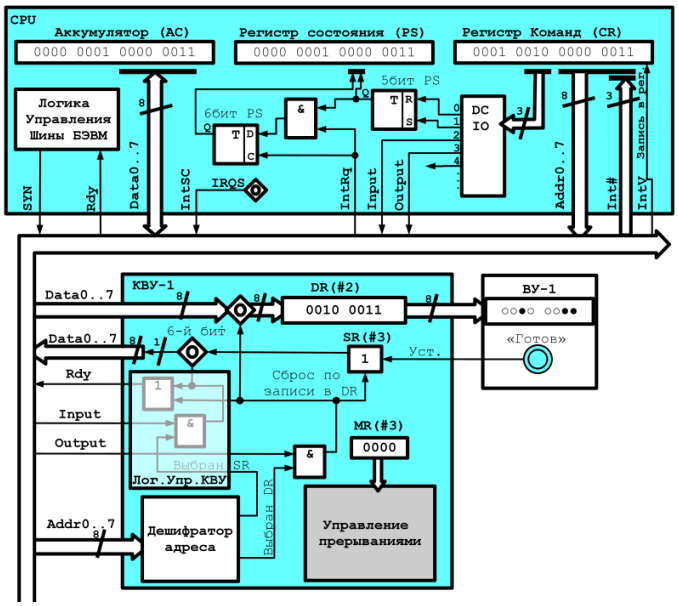
4. Принципы работы дешифратора

Дешифратор — это устройство, на ввод которого подаётся код числа, а на выходе выбирается только одна выходная линия, номер которой соответствует коду на входе дешифратора.
Все контроллеры подключены к одной шине ВУ, дешифратор позволяет выделить обращение к данному ВУ среди всех обращений к устройствам ввода-вывода, подключённых к процессору.

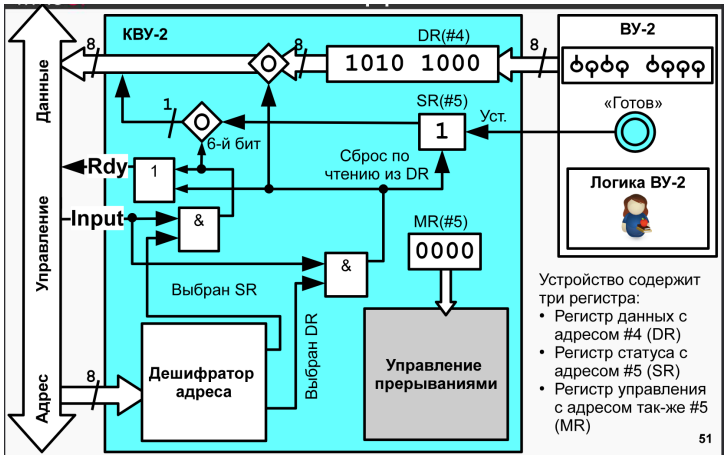
5. Устройство контроллера ввода-вывода



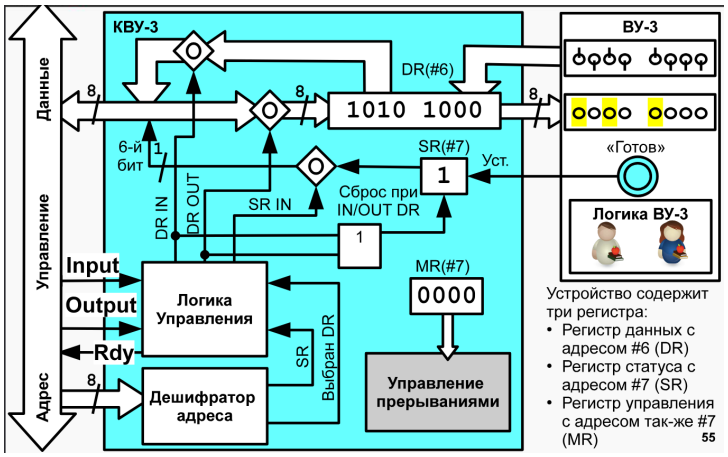
AC подключен к сегменту данных шины внешних устройств (задействуется только младший (правый) байт), к нему же подключён блок регистров внешнего устройства. CR подключён к сегменту адреса через двунаправленную шину (как и AC), оттуда выбирается адрес внешнего устройства (в CR задействуется младший байт для адреса и 8-11 биты для приказа). Адрес поступает на дешифратор адреса (внутри ВУ дешифраторов столько, сколько у ВУ регистров, каждый регистр проверяется отдельно), дальше выбирается 1 регистр из набора и подключается к шине данных. Приказ поступает на дешифратор приказа, превращается в набор управляющих приказов.



Дешифратор адреса дешифрирует регистр из младшего байта CR. Если выбран SR, то проверяется, есть ли вместе с адресом 3 команда IN (DR - аналогично). Готовность на ВУ - просто кнопка, а для БЭВМ это служебный сигнал. Сигнал готовности (Rdy), подтверждающий завершение операции ввода-вывода внутри цикла обмена между AC и DR соответствующего КВУ. В случае операции ввода Rdy подтверждает данные, передаваемые по шине данных, и в обоих случаях операции ввода-вывода сигнализирует о том, что цикл обмена с регистром данных ОК контроллера завершён.



Регистр состояния (SR - State Register), в котором хранится информация о готовности ВУ к обмену данными с процессором. В контроллерах простейших ВУ используются однобитовые регистры готовности, которые часто называют флагом. SR попадает на ту же шину, что и содержимое DR (в 6-й бит). Данные не смешиваются, так как отправка происходит в разные моменты времени (подключаются разные вентили, так как выбираются разные линии в зависимости от выбранного регистра). Если попытаться что-то записать в ВУ-2, то ничего не будет. 1 в SR появляется, когда нажата кнопка готов, а сбрасывается, когда данные считаны из регистра.



Устройство реагирует и на команду Out, и на команду In. 'Логика управления' из себя выдаёт 3 сигнала, один говорит, что у нас операция Out, другой - In, третий - чтение SR.

6. Сколько внешних устройств можно подключить в БЭВМ?

Если инициация обмена синхронная (возможно ли это в БЭВМ - хз, смотреть выше), то ≤ 256, иначе ≤ 128. Пояснение смотреть выше.

7. Описать последовательность действий ВУ и ЦП при асинхронном вводе (выводе аналогично)

Пример. С помощью устройства ввода ВУ-2 (DR4, SR5) записать в ячейку в памяти коды символов слова "ДА" в кодировке KOI8-R.
В начале программа висит в ожидании готовности: считывается статусный (с номером 5) регистр ВУ-2, который передается через шину данных в 6 разряде числа, сравнивается с 0x40 (01000000_2), и пока устройство не готово (SR равен 0). Такие циклы с неопределенным временем завершения называются циклами “spin-loop”, они постоянно проверяют готовность устройства или переменной в программе, загружая процессор. Как только 6 разряд SR принимает значение 1 (устройство готово), в регистр данных ВУ считывается в младшие 8 разрядов аккумулятора, старшие разряды AC при этом остаются неизменными. Для того, чтобы подготовить AC к приему второго символа, происходит обмен байтов аккумулятора при помощи команды SWAB, и сохранение этого символа в старших разрядах в ячейку результата. При данной реализации асинхронного обмена ЭВМ тратит время на ожидание (неопределённо долгое!) момента готовности циклически опрашивая флаг (spin-loop) и не может выполнять никакой другой работы. Разумная организация процедур ввода-вывода позволяет избегать такого зацикливания, например, через периодический опрос флага при выполнении основной программы или через прерывания.

8. О чём свидетельствует флаг готовности равный 0?

О том, что устройство не готово для ввода/вывода.

9. Существуют ли ситуации, в которых ВУ опрашивает свой флаг готовности?

Будто мы аппарат, принимающий монетку на вход и возвращающий какую-нибудь игрушку, подходит (условно пока обрабатывается первая монетка, аппарат не может принять вторую).

10. Можно ли убрать команду AND в блоке проверки готовности ВУ?

CMP + BZS.

11. Про кодировки

Символы: ASCII
ASCII расшифровывается как American Standard Code for Information Interchange. В первых 32 позициях кодировки (всего 128) расположены управляющие символы, которые используются в качестве служебных, например, для перемещения курсора на терминале и организации текстов. Организация текста была придумана исходя из принципа работы пишущей машинки. Поэтому в кодовой таблице есть символы перевода строки (LF), возврата на один символ (BS), возврата каретки (CR) и другие. С помощью символа BS на принтере можно печатать один символ поверх другого (так, например, можно печатать ударение).
Отметим, что разные ОС по-разному кодируют конец строки. Это связано с особенностью работы печатных машинок и телетайпов. UNIX-системы используют один символ CR (при этом виртуальная печатная машинка как бы переводит печатную консоль в начало и автоматически на следующую строку), а Windows - последовательностью из двух: перенос строк и возврат каретки. Из-за этого текстовые файлы между этими ОС не полностью совместимы. При этом старший бит используется не для кодировки символов, а для контроля чётности.

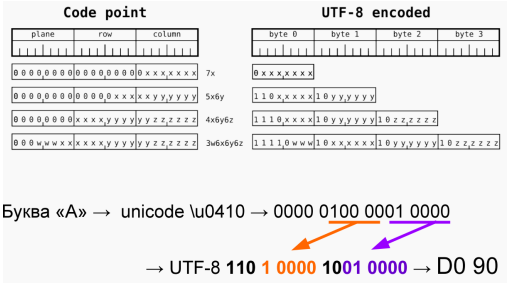
Символы: ASCII (КОИ-7H0)КОИ-7H1 (РУС), КОИ-7H2 (Mix)
Для представления русских символов в СССР был предложен набор кодовых таблиц КОИ-7 (Код Обмена Информацией). КОИ-7 включает в себя 3 «набора» H0, H1, H2. H0 это просто US-ASCII (однако символ доллара $ заменён на символ валюты), а строчные заменены на заглавные русские.
Кодировка является семибитной, как и ASCII, оставляя старший бит для контроля четности. Другой её особенностью было то, что русские символы, идентичные по начертанию с латинскими, находились в тех же местах, что и английские. В случае отсутствия в системе русских шрифтов текст, пусть и с неудобствами в восприятии, всегда можно было прочитать с помощью английских шрифтов.
Минусом такого подхода была усложнённая сортировка данных в алфавитном порядке. Для сортировки была необходима дополнительная таблица, которая содержала коды символов в алфавитном порядке.

Символы: КОИ-8
Когда появилась Extended ASCII, где для символов использовались все 8 разрядов байта, было решено для русских заглавных и строчных символов использовать позиции верхней части таблицы. Данная кодовая таблица получила название КОИ-8. В ней, кроме символов русского языка, были еще и символы псевдографики для рисования таблиц. Следует заметить, что если по каким-то причинам ПО не поддерживало 8-битные кодировки и обрезало старший бит (при контроле четности) то русские символы перемещались в младшую часть таблицы, и их можно было прочитать по сходным с русскими по начертанию английским символам. КОИ-8 до сих пор используется в качестве стандарта для обмена электронной почтой.

Символы: ISO 8859-5 (ГОСТ-основная)
ISO 8859-5 — 8-битная кодовая страница из семейства кодовых страниц стандарта ISO-8859 для представления кириллицы. ISO 8859-5 была создана в 1988 году на базе «основной кодировки». ISO 8859-5 широко применяется в Сербии и иногда в Болгарии на юниксоподобных системах. В России эта кодировка в настоящее время почти не употребляется. Преимущества по сравнению с КОИ-7: русские буквы по алфавиту. Символ занимает 1 байт.
До широкого использования Unicode в российском сегменте вычислительных систем велись «войны кодировок». «Де факто» использовались пять разных таблиц кодировок: с КОИ-8 использовались две кодировки ISO 8859-5 (ГОСТ-основная) и CP 866 (ГОСТ-альтернативная, в DOS), Win-1251, кириллическая кодировка для компьютеров Macintosh.

Символы: WIN1251
Несмотря на многообразие и стандартизацию кириллических символов, компания Microsoft в операционной системе Windows использовала не кодовую страницу CP866 из своей предыдущей операционной системы DOS, а новую, которая была создана на базе кодировок, использовавшихся в ранних «самопальных» русификаторах Windows в 1990—1991 гг. совместно с представителями «Параграфа», «Диалога» и российского отделения Microsoft.

Символы: UNICODE, UTF-8



В настоящее время используется (предложён в 91 году) общемировая кодовая таблица Unicode. Применение этого стандарта позволяет закодировать очень большое число символов из разных систем письменности. Стандарт состоит из двух основных частей: универсального набора символов (Universal character set, UCS) и нескольких форм представления или кодировок (Unicode transformation format, UTF). Универсальный набор символов перечисляет допустимые по стандарту Юникод символы и присваивает каждому символу код в виде неотрицательного целого числа, записываемого обычно в шестнадцатеричной форме с префиксом U+, например, U+040F.
Семейство кодировок определяет способы преобразования кодов символов для передачи в потоке или в файле. В настоящее время описаны более 136 тысяч символов. В своей начальной части Unicode полностью совпадает с ASCII. Если символ превышает 7 разрядов, то он представляется в виде двух байтов. При этом первые 5 битов предваряются преамбулой 110 и упаковываются в байт, следующий байт идет с преамбулой 10, и остальные 6 разрядов копируются во второй байт. Если символ занимает больше 2 байтов, то кодировка происходит как на картинке.
Проблема UTF-8 состоит в том, что один символ может быть закодирован 1-4 байтами. Наиболее часто используемая операция со строками - определение длины. В UTF-8 её сложнее определить, чем для любой однобайтной кодировки, где длина строки равна количеству байтов. Подсчет количества символов для кодировки UTF-8 реализован системными библиотеками. Недостаток несоответствия длины строки количеству байтов можно исправить, используя кодировку UTF-16 или UTF-32, однако для представления строки в данных форматах используется больше памяти.

Big-endian и Little-endian
Представьте, что у вас в регистре лежит слово, а для пересылки слова в память, вы должны расположить байты этого слова в определенном порядке. Как удобней для ЭВМ выполнить это размещение - с младшего или со старшего байта? Этот вопрос породил большое количество споров. Приверженцев разных способов размещения и сам способ размещения назвали Big-endian и Little-endian (аналогия с "Путешествиями Гулливера").
В современных ЭВМ в основном используется Little-endian, младшие разряды слова размещаются в начальных байтах памяти. В течение эволюции ЭВМ было выяснено, что никаких преимуществ один тип кодирования над другим не имеет. До сих пор между различными архитектурами, например, Intel и SPARC, возникают проблемы совместимости.