# Лабораторная работа № 7

# Программирование внешних устройств

Цель работы:

1. Знакомство с внешними устройствами модели учебной ЭВМ.

2. Изучение способов организации взаимодействия процессора и внешних устройств (ВУ) в составе ЭВМ.

Теоретические сведения:

**1. Структура модели ЭВМ**

Моделируемая ЭВМ включает:

- процессор;

- блок регистров общего назначения;

- оперативную память (ОЗУ);

- сверхоперативную память (СОЗУ) (кэш-память);

- устройства ввода (УВв);

- устройства вывода (УВыв).

Процессор, в свою очередь, состоит из:

- центрального устройства управления (УУ);

- арифметического устройства (АУ);

- блока системных регистров (CR, PC, SP и др.).

Структурная схема ЭВМ показана на рисунке 1.

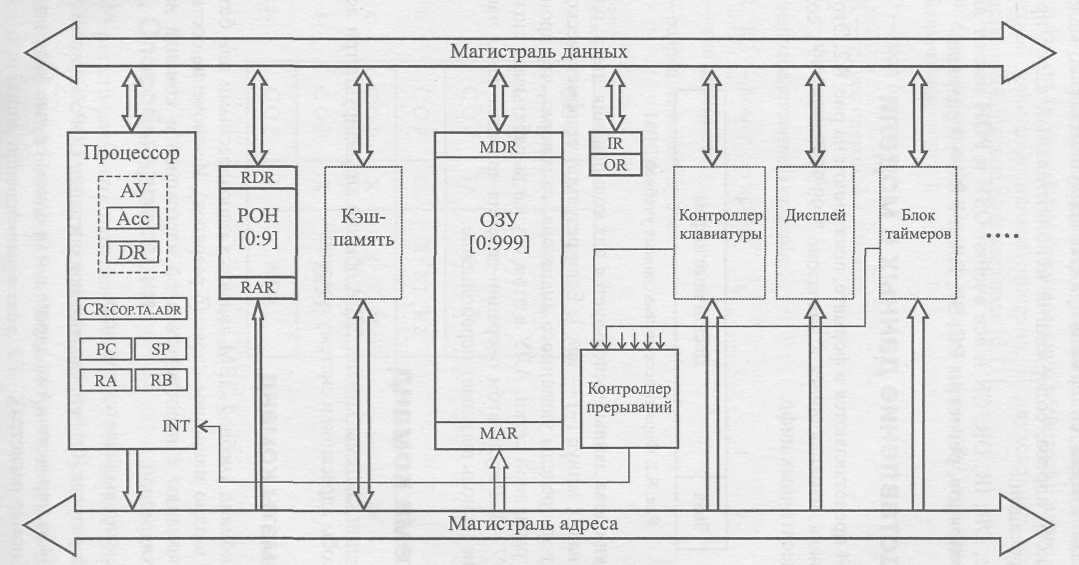


Рис. 1. Общая структура учебной ЭВМ

**Представление данных в модели**

Данные в ЭВМ представляются в формате, показанном на рис. 2. Это целые десятичные числа, изменяющиеся в диапазоне "-99 999... +99 999", содержащие знак и 5 десятичных цифр.

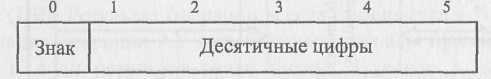


Рис. 2. Формат десятичных данных учебной ЭВМ

***Старший разряд*** слова данных используется для кодирования знака: плюс (+) изображается как 0, минус (-) — как 1.

***Если результат*** арифметической операции ***выходит за пределы*** указанного диапазона, то говорят, что произошло переполнение разрядной сетки. АЛУ в этом случае вырабатывает сигнал переполнения **OV = 1**.

*Результатом операции деления является целая часть частного. Деление на ноль вызывает переполнение*.

**2. Работа с внешними устройствами в программе** **CompModel**

Модель учебной ЭВМ реализована в виде программы CompModel.exe, которая находиться в подкаталоге Программы, расположенном в том же каталоге где и текст данной лабораторной работы. В программной модели учебной ЭВМ использован стандартный интерфейс Windows, реализованный в нескольких окнах.

**2.1. Внешние устройства модели ЭВМ**

***Модели внешних устройств*** (ВУ), используемые в описываемой системе, реализованы по единому принципу.

С точки зрения процессора они ***представляют собой ряд программно-доступных регистров, лежащих в адресном пространстве ввода/вывода***.

Размер регистров ВУ совпадает с размером ячеек памяти и регистров данных процессора — шесть десятичных разрядов.

**Доступ к регистрам ВУ** осуществляется по командам **in** *аа,* **out** *аа,* где *аа* — двухразрядный десятичный адрес регистра ВУ.

Таким образом, общий объем адресного пространства ввода/вывода составляет 100 адресов. Следует помнить, что адресные пространства памяти и ввода/вывода в этой модели разделены.

Разные ВУ содержат различное число программно-доступных регистров, каждому из которых соответствует свой адрес, причем нумерация адресов всех ВУ начинается с 0.

***При создании ВУ*** ему ставится в соответствие ***базовый адрес***в пространстве ввода/вывода, и все адреса его регистров становятся *смещениями* относительно этого базового адреса.

Если в системе создаются ***несколько ВУ***, то их базовые адреса следует выбирать ***с учетом величины адресного пространства***, занимаемого этими устройствами, исключая наложение адресов.

Если ВУ способно формировать ***запрос на прерывание***, то при создании ему ставится в соответствие ***вектор прерывания***— десятичное число. Разным ВУ должны назначаться различные векторы прерываний.

**Программная модель учебной ЭВМ** комплектуется набором внешних устройств, включающим:

* контроллер клавиатуры;
* дисплей;
* блок таймеров;
* тоногенератор,

которым по умолчанию присвоены параметры, перечисленные в табл. 1.

Таблица 1. Параметры внешних устройств

|  |  |  |  |
| --- | --- | --- | --- |
| **Внешнее устройство** | **Базовый адрес** | **Адреса регистров** | **Вектор прерывания** |
| Контроллер клавиатуры | **0** | **0, 1, 2** | **0** |
| Дисплей | **10** | **0, 1, 2, 3** | **Нет** |
| Блок таймеров | **20** | **0, 1, 2, 3, 4, 5, 6** | **2** |
| Тоногенератор | **30** | **0, 1** | **Нет** |

При создании устройств пользователь может изменить назначенные по умолчанию базовый адрес и вектор прерывания.

В описываемой версии системы не предусмотрена возможность подключения в систему нескольких одинаковых устройств.

Большинство внешних устройств содержит регистры *управления* CR и *состояния* SR, причем обычно регистры CR доступны только по записи, а SR — по чтению.

Регистр CR содержит флаги и поля, определяющие режимы работы ВУ, а SR — флаги, отражающие текущее состояние ВУ. Флаги SR устанавливаются аппаратно, но сбрасываются программно (или по внешнему сигналу). Поля и флаги CR устанавливаются и сбрасываются программно при записи кода данных в регистр CR или специальными командами.

Контроллер ВУ интерпретирует код, записываемый по адресу CR как команду, если третий разряд этого кода равен 1, или как записываемые в CR данные, если третий разряд равен 0. В случае получения командного слова запись в регистр CR не производится, а пятый разряд слова рассматривается как код операции.

**2.1.1. Контроллер клавиатуры**

**Контроллер клавиатуры** (рис. 3) представляет собой модель внешнего устройства, принимающего ASCII-коды от клавиатуры ПЭВМ.

Символы помещаются последовательно в *буфер символов,* размер которого установлен равным 50 символам, и отображаются в окне обозревателя (рис. 4).

В состав контроллера клавиатуры входят три программно-доступных регистра:

* DR (адрес 0) — регистр данных;
* CR (адрес 1)— регистр управления, определяет режимы работы контроллера и содержит следующие флаги:
  + Е — флаг разрешения приема кодов в буфер;
  + I — флаг разрешения прерывания;
  + S — флаг режима посимвольного ввода.
* SR (адрес 2) — регистр состояния, содержит два флага:
  + Err — флаг ошибки;
  + Rd — флаг готовности.

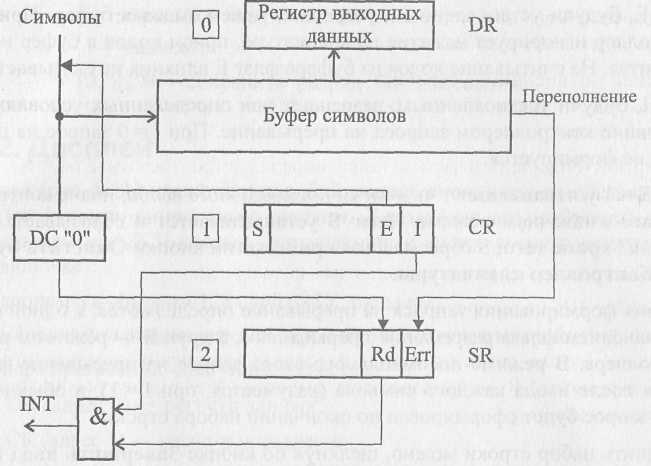


Рис. 3. Контроллер клавиатуры

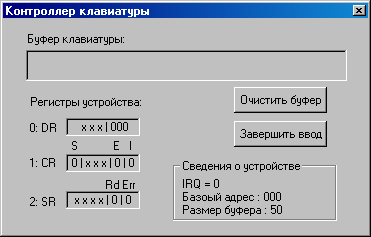


Рис. 4. Окно обозревателя контроллера клавиатуры

***Регистр данных DR***доступен только для чтения, через него считываются ASCII-коды из буфера, причем порядок чтения кодов из буфера соответствует порядку их записи в буфер — каждое чтение по адресу 0 автоматически перемещает указатель чтения буфера. В каждый момент времени DR содержит код символа по адресу указателя чтения буфера.

Флаги ***регистра управления CR***устанавливаются и сбрасываются программно.

Флаг Е, будучи установленным, разрешает прием кодов в буфер.

При Е = О контроллер игнорирует нажатие на клавиатуре, прием кодов в буфер не производится. На считывание кодов из буфера флаг Е влияния не оказывает.

Флаг I, будучи установленным, разрешает при определенных условиях формирование контроллером запроса на прерывание. При 1 = 0 запрос на прерывание не формируется.

Флаг S = 1 устанавливает т. н, *режим посимвольного ввода,* иначе контроллер работает в обычном режиме. Флаг S устанавливается и сбрасывается программно, кроме того, S сбрасывается при нажатии кнопки **Очистить буфер** в окне **Контроллер клавиатуры.**

Условия формирования запроса на прерывание определяются, с одной стороны, значением флага разрешения прерывания I, с другой — режимом работы контроллера. В режиме посимвольного ввода запрос на прерывание формируется после ввода каждого символа (разумеется, при I=1), в обычном режиме запрос будет сформирован по окончании набора строки.

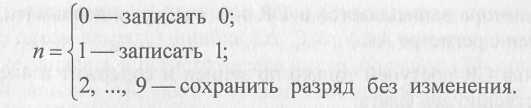
Завершить набор строки можно, щелкнув по кнопке **Завершить ввод** в окне **Контроллер клавиатуры** (см. рис. 8.10). При этом устанавливается флаг готовности Rd (от англ. *ready)* в регистре состояния SR. Флаг ошибки Err (от англ. *error)* в том же регистре устанавливается при попытке ввода в буфер 51-го символа. Ввод 51-го и всех последующих символов блокируется.

Сброс флага Rd осуществляется автоматически при чтении из регистра DR, флаг Err сбрасывается программно. Кроме того, оба эти флага сбрасываются при нажатии кнопки **Очистить буфер** в окне **Контроллер клавиатуры;** одновременно со сбросом флагов производится очистка буфера— весь буфер заполняется кодами 00h, и указатели записи и чтения устанавливаются на начало буфера.

Для программного управления контроллером предусмотрен ряд командных слов. Все команды выполняются при записи по адресу регистра управления CR кодов с 1 в третьем разряде.

**Контроллер клавиатуры интерпретирует следующие командные слова:**

* xxxl0l — очистить буфер (действие команды эквивалентно нажатию кнопки Очистить буфер);
* ххх 102 — сбросить флаг Err в регистре SR;
* ххx10З — установить флаг S в регистре CR;
* ххх 104 — сбросить флаг S в регистре CR.



Если по адресу 1 произвести запись числа ххх0nnс, то произойдет изменение 4-го и 5-го разрядов регистра CR по следующему правилу (1):

**2.1.2. Дисплей**

**Дисплей** (рис. 5) представляет собой модель внешнего устройства, реализующую функции символьного дисплея. Дисплей может отображать символы, задаваемые ASCII-кодами, поступающими на его регистр данных. Дисплей включает:

* видеопамять объемом 128 слов (ОЗУ дисплея);
* символьный экран размером 8 строк по 16 символов в строке;
* четыре программно-доступных регистра:
  + DR (адрес 0) — регистр данных;
  + CR (адрес 1) — регистр управления;
  + SR (адрес 2) — регистр состояния;
  + AR (адрес 3) — регистр адреса.

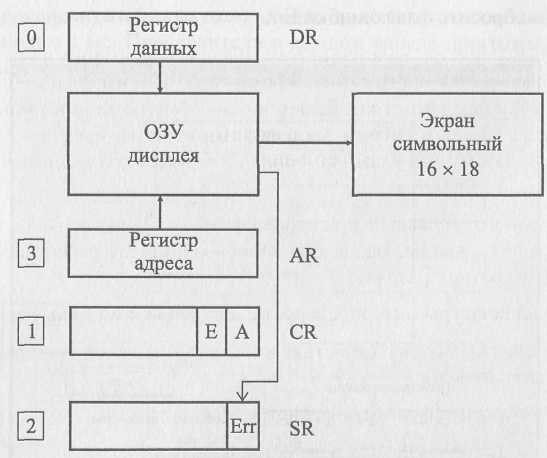


Рис. 5. Контроллер дисплея

Через *регистры адреса* AR и *данных* DR по записи и чтению осуществляется доступ к ячейкам видеопамяти. При обращении к регистру DR по записи содержимое аккумулятора записывается в DR и в ячейку видеопамяти, адрес которой установлен в регистре AR.

*Регистр управления* CR доступен только по записи и содержит в 4-м и 5-м разрядах соответственно два флага:

* Е — флаг разрешения работы дисплея; при Е = 0 запись в регистры AR и DR блокируется;
* А — флаг автоинкремента адреса; при А = 1 содержимое AR автоматически увеличивается на 1 после любого обращения к регистру DR— по записи или чтению.

Изменить значения этих флагов можно, если записать по адресу CR (по умолчанию — 11) код хххОnn, при этом изменение 4-го и 5-го разрядов регистра CR произойдет согласно выражению (1).

Для программного управления дисплеем предусмотрены две команды, коды которых должны записываться по адресу регистра CR, причем в третьем разряде командных слов обязательно должна быть 1:

* xxxlOl — очистить дисплей (действие команды эквивалентно нажатию кнопки **Очистить** в окне **Дисплей),** при этом очищается видеопамять (в каждую ячейку записывается код пробела — 032), устанавливается в 000 регистр адреса AR и сбрасываются флаги ошибки Err и автоинкремента А;
* ххх102 — сбросить флаг ошибки Err.

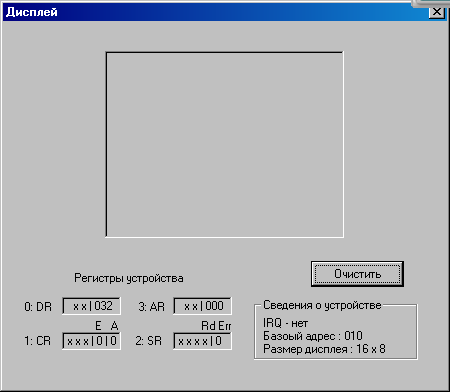


Рис. 6. Окно обозревателя контроллера дисплея

Регистр состояния SR доступен только по чтению и содержит единственный флаг (в пятом разряде) ошибки Err. Этот флаг устанавливается аппаратно при попытке записать в регистр адреса число, большее 127, причем как в режиме прямой записи в AR, так и в режиме автоинкремента после обращения по адресу 127. Сбрасывается флаг Err программно или при нажатии кнопки **Очистить** в окне **Дисплей** (рис. 6).

**2.1.3. Блок таймеров**

**Блок таймеров** (рис. 7) включает в себя три однотипных канала, каждый из которых содержит:

* пятиразрядный десятичный реверсивный счетчик Т, на вход которого поступают метки времени (таймер);
* программируемый предделитель D;
* регистр управления таймером CTR;
* флаг переполнения таймера FT.

Регистры таймеров Т доступны по записи и чтению (адреса 1, 3, 5 соответственно для Tl, T2, ТЗ). Программа в любой момент может считать текущее содержимое таймера или записать в него новое значение.

На входы предделителей поступает общие для всех каналов метки времени CLK с периодом 1 мс, Предделители в каждом канале программируются независимо, поэтому таймеры могут работать с различной частотой.

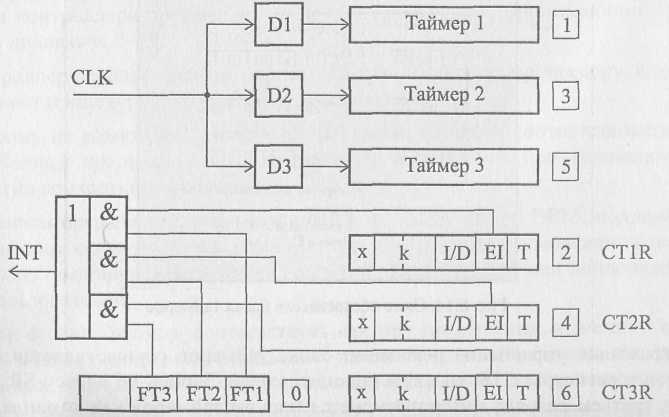


Рис. 7. Блок таймеров

***Регистры управления CTR*** доступны по записи и чтению (адреса 2, 4, 6) и содержат следующие поля:

* Т (разряд 5) — флаг включения таймера;
* EI (разряд 4)— флаг разрешения формирования запроса на прерывание при переполнении таймера;
* I/D (разряд 3) — направление счета (инкремент/декремент), при I/D = О таймер работает на сложение, при I/D =1 — на вычитание;
* *к* (разряды [1:2]) — коэффициент деления предделителя (от 1 до 99).

Флаги переполнения таймеров собраны в один регистр — доступный только по чтению регистр состояния SR, имеющий адрес 0. Разряды регистра (5, 4 и 3 для Tl, Т2, ТЗ соответственно) устанавливаются в 1 при переполнении соответствующего таймера.

Для таймера, работающего на сложение, переполнение наступает при переходе его состояния из 99 999 в 0, для вычитающего таймера — переход из 0 в 99 999.

В окне обозревателя (рис. 8) предусмотрена кнопка Сброс, нажатие которой сбрасывает в 0 все регистры блока таймеров, кроме CTR, которые устанавливаются в состояние 001000. Таким образом, все три таймера обнуляются, переключаются в режим инкремента, прекращается счет, запрещаются прерывания, сбрасываются флаги переполнения и устанавливаются коэффициенты деления предделителей равными 01.

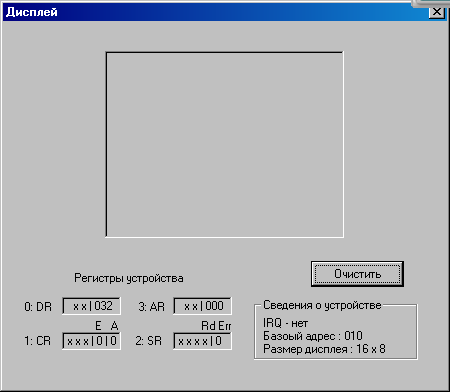


Рис. 8. Окно обозревателя блока таймеров

Программное управление режимами блока таймеров осуществляется путем записи в регистры CTR соответствующих кодов. Запись по адресу SR числа с 1 в третьем разряде интерпретируется блоком таймеров как команда, причем младшие разряды этого числа определяют код команды:

* xхх100 — общий сброс (эквивалентна нажатию кнопки **Сброс** в окне обозревателя);
* xxxl0l — сброс флага переполнения таймера FT1;
* ххх102 — сброс флага переполнения таймера FT2;
* ххх10З — сброс флага переполнения таймера FT3.

**2.1.4. Тоногенератор**

Модель этого простого внешнего устройства не имеет собственного обозревателя, содержит всего два регистра, доступных только для записи:

* FR (адрес 0) — регистр частоты звучания (Гц):
* LR (адрес 1) — регистр длительности звучания (мс).

По умолчанию базовый адрес тоногенератора— 30. Сначала следует записать в FR требуемую частоту тона в герцах, затем в LR — длительность звучания в миллисекундах. Запись числа по адресу регистра LR одновременно является командой на начало звучания.

**2.2. Подсистема прерываний**

В модели учебной ЭВМ предусмотрен механизм векторных внешних прерываний. Внешние устройства формируют запросы на прерывания, которые поступают на входы *контроллера прерываний.*

При подключении ВУ, способного формировать запрос на прерывание, ему ставится в соответствие номер входа контроллера прерываний — вектор прерывания, принимающий значение в диапазоне 0—9.

Контроллер передает вектор, соответствующий запросу, процессору, который начинает процедуру обслуживания прерывания.

Каждому из возможных в системе прерываний должен соответствовать т. н. *обработчик прерывания* — подпрограмма, вызываемая при возникновении события конкретного прерывания.

Механизм прерываний, реализованный в модели учебной ЭВМ, поддерживает *таблицу векторов прерываний,* которая создается в оперативной памяти *моделью операционной системы* (если она используется) или непосредственно пользователем.

Номер строки таблицы соответствует вектору прерывания, а элемент таблицы — ячейка памяти, в трех младших разрядах которой размещается начальный адрес подпрограммы, обслуживающей прерывание с этим вектором.

Таблица прерываний в рассматриваемой модели жестко фиксирована— она занимает ячейки памяти с адресами 100—109. Таким образом, адрес обработчика с вектором 0 должен располагаться в ячейке 100, с вектором 2 — в ячейке 102. При работе с прерываниями не рекомендуется использовать ячейки 100—109 для других целей.

Процессор начинает обработку прерывания (если они разрешены), завершив текущую команду. При этом он:

1. Получает от контроллера вектор прерывания.

2. Формирует и помещает в верхушку стека слово, три младших разряда ([3:5]) которого — текущее значение PC (адрес возврата из прерывания), а разряды [1:2] сохраняют десятичный эквивалент шестнадцатеричной цифры, определяющей значение вектора флагов (I, OV, S, Z). Например, если I=1, OV = О, S = 1, *Z—* 1, то в разряды [1:2] запишется число 1110 =10112.

3. Сбрасывает в 0 флаг разрешения прерывания I.

4. Извлекает из таблицы векторов прерываний адрес обработчика, соответствующий обслуживаемому вектору, и помещает его в PC, осуществляя тем самым переход на подпрограмму обработчика прерывания.

Таким образом, вызов обработчика прерывания, в отличие от вызова подпрограммы, связан с помещением в стек не только адреса возврата, но и текущего значения вектора флагов. Поэтому последней командой подпрограммы обработчика должна быть команда iret, которая не только возвращает в PC три младшие разряда ячейки — верхушки стека (как ret), но и восстанавливает те значения флагов, которые были в момент перехода на обработчик прерывания.

Не всякое событие, которое может вызвать прерывание, приводит к прерыванию текущей программы. В состав процессора входит программно-доступный флаг I разрешения прерывания. При I = 0 процессор не реагирует на запросы прерываний. После сброса процессора флаг I так же сброшен и все прерывания запрещены. Для того чтобы разрешить прерывания, следует в программе выполнить команду ei (от англ. *enable interrupt).*

Выше отмечалось, что при переходе на обработчик прерывания флаг I автоматически сбрасывается, в этом случае прервать обслуживание одного прерывания другим прерыванием нельзя. По команде iret значение флагов восстанавливается, в т. ч. вновь устанавливается I=1, следовательно, в основной программе прерывания опять разрешены.

Если требуется разрешить другие прерывания в обработчике прерывания, достаточно в нем выполнить команду ei. Контроллер прерываний и процессор на аппаратном уровне блокируют попытки запустить прерывание, если его обработчик начал, но не завершил работу.

Таким образом, флаг I разрешает или запрещает все прерывания системы. Если требуется выборочно разрешить некоторое подмножество прерываний, используются программно-доступные флаги разрешения прерываний непосредственно на внешних устройствах.

Как правило, каждое внешнее устройство, которое может вызвать прерывание, содержит в составе своих регистров разряд флага разрешения прерывания (см. формат регистров CR и CTR на рис. 9, 13), по умолчанию установленный в 0. Если оставить этот флаг в нуле, то внешнему устройству запрещается формировать запрос контроллеру прерываний.

Иногда бывает удобно (например, в режиме отладки) иметь возможность вызвать обработчик прерывания непосредственно из программы. Если использовать для этих целей команду call, которая помещает в стек только адрес возврата, то команда iret, размещенная последней в обработчике, может исказить значения флагов (все они будут сброшены в 0, т. к. команда call формирует только три младшие разряда ячейки верхушки стека, оставляя остальные разряды в 000).

Поэтому в системах команд многих ЭВМ, в т. ч. и нашей модели, имеются команды вызова прерываний— int n (в нашей модели n  {0, 1,..., 9}), где n— вектор прерывания. Процессор, выполняя команду int n, производит те же действия, что и при обработке прерывания с вектором n.

Характерно, что с помощью команды int n можно вызвать обработчик прерывания даже в том случае, когда флаг разрешения прерывания I сброшен.

**3. Порядок работы с внешними устройствами модели учебной ЭВМ**

Выше отмечалось, что связь процессора и ВУ может осуществляться в синхронном или асинхронном режиме.

***Синхронный режим***используется для ВУ, всегда готовых к обмену. В нашей модели такими ВУ являются дисплей и тоногенератор — процессор может обращаться к этим ВУ, не анализируя их состояние (правда дисплей блокирует прием данных после ввода 128 символов, формируя флаг ошибки).

***Асинхронный обме****н* предполагает анализ процессором состояния ВУ, которое определяет готовность ВУ выдать или принять данные или факт осуществления некоторого события, контролируемого системой. К таким устройствам в нашей модели можно отнести клавиатуру и блок таймеров.

**Анализ состояния ВУ** может осуществляться процессором двумя способами:

-в программно-управляемом режиме;

- в режиме прерывания.

*В первом случае* предполагается программное обращение процессора к регистру состояния ВУ с последующим анализом значения соответствующего разряда слова состояния. Такое обращение следует предусмотреть в программе с некоторой периодичностью, независимо от фактического наступления контролируемого события (например, нажатие клавиши).

*Во втором случае* при возникновении контролируемого события ВУ формирует процессору запрос на прерывание программы, по которому процессор и осуществляет связь с ВУ.

**4. Вспомогательные таблицы**

В данном разделе представлены вспомогательные таблицы (табл. 2—4) для работы с моделью учебной ЭВМ.

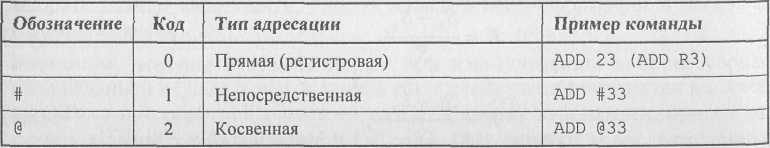


Таблица 2. Типы адресации, их коды и обозначение

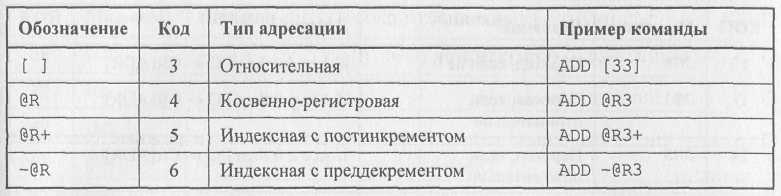


Таблица 3. Таблица кодов ASCII (фрагмент)

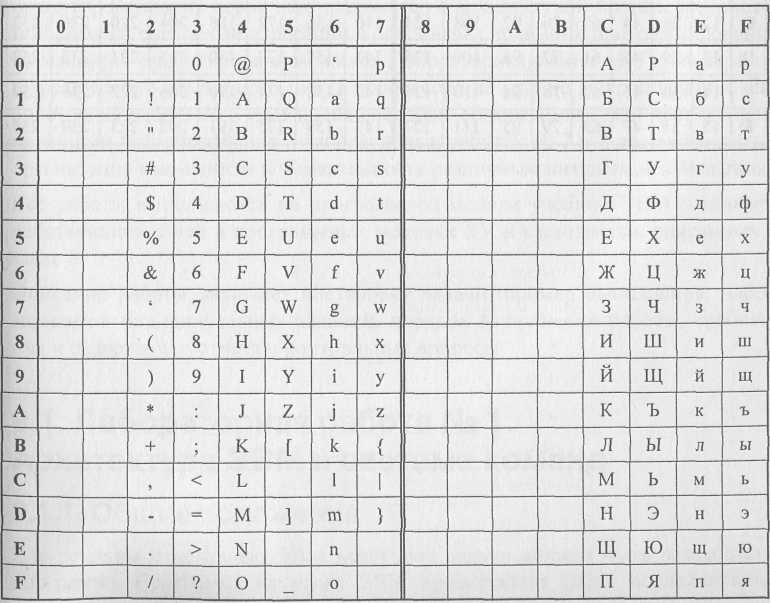
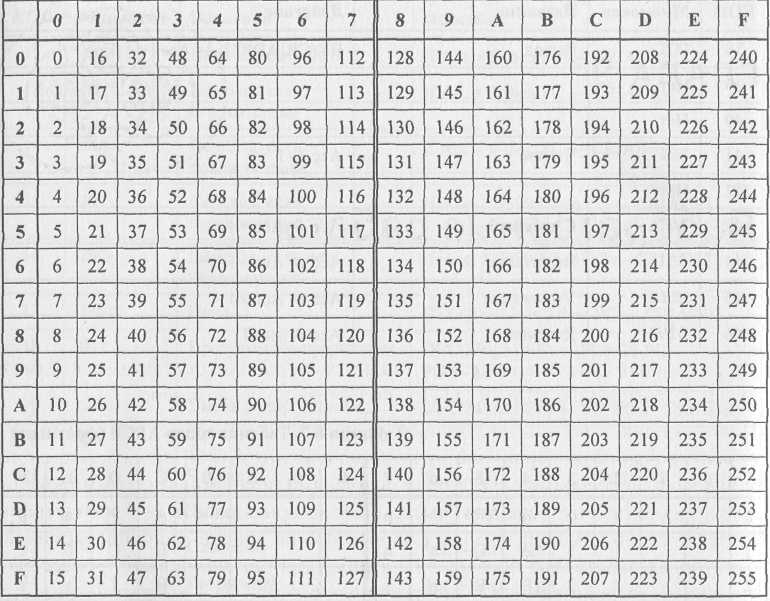


Таблица 4. Перевод шестнадцатиричных кодов в десятичные



Задания для выполнения:

**1.**Изучить теоретические сведения, посвященные работе внешних устройств модели учебной ЭВМ.

**2.** Выполнить задание, указанное в таблице 5. Выбор задания выполняется по номеру Вашего индивидуального задания.

Свой вариант задания (табл. 1) требуется выполнить двумя способами — сначала в режиме программного контроля, далее модифицировать программу таким образом, чтобы события обрабатывались в режиме прерывания программы. Поскольку "фоновая" (основная) задача для этого случая в заданиях отсутствует, роль ее может сыграть "пустой цикл":

М: NOP

NOP

JMP М

Таблица 5. Варианты задания

| № **варианта** | **Задание** | | **Используемые ВУ** | **Пояснения** |
| --- | --- | --- | --- | --- |
| **1** | Ввод пятиразрядных чисел в ячейки ОЗУ | | Клавиатура | Программа должна обеспечивать ввод последовательности ASCII-кодов десятичных цифр (не длиннее пяти), перекодировку в "8421", упаковку в десятичное число (первый введенный символ — старшая цифра) и размещение в ячейке ОЗУ. ASCII-коды нецифр игнорировать |
| **2** | Программа ввода символов с клавиатуры с выводом на дисплей | | Клавиатура, дисплей, таймер | Очистка буфера клавиатуры после ввода 50 символов или каждые 10 с |
| **3** | Вывод на дисплей трех текстов, хранящихся в памяти, с задержкой | | Дисплей, таймер | Первый текст выводится сразу при запуске программы, второй — через 15 с, третий — через 20 с после второго |
| **4** | Вывод на дисплей одного из трех текстовых сообщений, в зависимости от нажатой клавиши | | Клавиатура, дисплей | <1>— вывод на дисплей первого текстового сообщения, <2> — второго, <3> — третьего, остальные символы — нет реакции |
| **5** | Выбирать из потока ASCII-кодов только цифры и выводить их на дисплей | | Клавиатура, дисплей, тоногенератор | Вывод каждой цифры сопровождается коротким звуковым сигналом |
| **6** | Выводить на дисплей каждый введенный с клавиатуры символ, причем цифру выводить "в трех экземплярах" | | Клавиатура, дисплей, тоногенератор | Вывод каждой цифры сопровождается троекратным звуковым сигналом |
| 7 | Вывод содержимого заданного участка памяти на дисплей посимвольно с заданным промежутком времени между выводами символов | | Дисплей, таймер | Остаток от деления на 256 трех младших разрядов ячейки памяти рассматривается как ASCII-код символа. Начальный адрес памяти, длина массива вывода и промежуток времени — параметры подпрограммы |
| **8** | Программа ввода символов с клавиатуры с выводом на дисплей | | Клавиатура, дисплей | Очистка буфера клавиатуры после ввода 35 символов |
| 9 | | Выводить на дисплей каждый введенный с клавиатуры символ, причем заглавную русскую букву выводить "в двух экземплярах" | Клавиатура, дисплей, таймер | Очистка буфера клавиатуры после ввода 48 символов, очистка экрана каждые 15с |
| **10** | | Вывод на дисплей содержимого группы ячеек памяти в числовой форме (адрес и длина группы — параметры подпрограммы) | Дисплей, таймер | Содержимое ячейки распаковывается (с учетом знака), каждая цифра преобразуется в соответствующий ASCII-код и выдается на дисплей. При переходе к выводу содержимого очередной ячейки формируется задержка 10 с |
| 11 | | Определить промежуток времени между двумя последовательными нажатиями клавиш | Клавиатура, таймер | Результат выдается на OR. (Учитывая инерционность модели, нажатия не следует производить слишком быстро.) |

2.1. Запустить программную модель учебной ЭВМ и подключить к ней определенные в задании внешние устройства (меню **Внешние устройства** > **Менеджер ВУ).**

2.2. Написать и отладить программу, предусмотренную заданием, с использованием программного анализа флагов готовности ВУ.

В отчет поместить тексты программ и скриншоты их работы.

**Рассмотрим пример выполнения ЛР**

**Тема:** Программирование внешних устройств.

**Цель**: изучение способов организации взаимодействия процессора и внешних устройств (ВУ) в составе ЭВМ

**Задание**. Написать программу, позволяющую осуществить селективный ввод символов с клавиатуры.

**Используемые ВУ.** Клавиатура, дисплей

**Пояснения**. Все русские буквы, встречающиеся в строке ввода – в верхнюю часть экрана дисплея (строки 1—4), все цифры — в нижнюю часть экрана (строки 5—8), остальные символы не выводить

**Ход работы:**

**Код программы с обработчиком прерываний**

| Адрес | Команда | Примечание |
| --- | --- | --- |
| 0 | RD #27 | Адрес обработчика прерывания клавиатуры |
| 1 | WR 100 | запишем по адресу обработчика IRQ0 |
| 2 | RD #0 | Начальный адрес куда будем писать русские символы |
| 3 | WR R0 | Сохраним |
| 6 | RD #64 | Начальный адрес куда будем писать цифры |
| 7 | WR R1 | Сохраним |
| 8 | RDI #101 | Команда очистки буфера |
| 9 | OUT 1 | для клавиатуры |
| 10 | OUT 11 | и дисплея |
| 11 | RDI #102 | Команда сброса ошибки |
| 12 | OUT 1 | для клавиатуры |
| 13 | OUT 11 | и для дисплея |
| 14 | RDI #11 | Команда разрешения МВУ и работы по прерыванию |
| 15 | OUT 1 | для клавиатуры |
| 16 | RDI #10 | Команда разрешения МВУ |
| 17 | OUT 11 | для монитора без авто увеличения адреса |
| 18 | RDI #103 | Команда установки посимвольного ввода |
| 19 | OUT 1 | для клавиатуры |
| 20 | EI | Разрешаем прерывания от устройств (клавиатуры) |
| 21 | LOOP: | Пустой цикл |
| 22 | NOP | Вся работа в прерывании |
| 23 | NOP |  |
| 24 | JMP LOOP |  |
| 25 | INT0:IN 0 | Считываем код с клавиатуры |
| 26 | WR R2 | Сохраним |
| 27 | SBI #48 | <0 |
| 28 | JS RESET | Игнорируем, на сброс клавиатуры |
| 29 | RD R2 | код |
| 30 | SBI #58 | >9 |
| 31 | JNS CMPRUS | Проверим на русские буквы |
| 32 | RD R1 | Цифру выводим в нижней части экрана, в R1 адрес, куда писать |
| 33 | OUT 13 | В регистр адреса |
| 34 | ADI #1 | инкремент адреса |
| 35 | WR R1 | Сохраним |
| 36 | RD R2 | код |
| 37 | OUT 10 | На дисплей! |
| 38 | JMP RESET | На сброс клавиатуры |
| 39 | CMPRUS:RD R2 | Код, проверим на > 128 - там русские буквы |
| 40 | SBI #128 | Сравним со 128 |
| 41 | JS RESET | < 128 игнорируем |
| 42 | RD R0 | Русские буквы выводим в верхней части экрана, в R0 адрес, куда писать |
| 43 | OUT 13 | в регистр адреса |
| 44 | ADI #1 | инкремент адреса |
| 45 | WR R0 | Сохраним |
| 46 | RD R2 | код |
| 47 | OUT 10 | На дисплей! |
| 48 | RESET:RDI #101 | Команда сброса |
| 49 | OUT 1 | В клавиатуру |
| 50 | RDI #103 | Команда установки посимвольного ввода (сбрасывается при сбросе) |
| 51 | OUT 1 | В клавиатуру |

**Код программы с программным анализом флагов и готовности ВУ**

| Адрес | Команда | Примечание |
| --- | --- | --- |
| Мнемокод |
| 0 | RD #0 | Начальный адрес, куда будем писать русские символы |
| 1 | WR R0 | Сохраним |
| 2 | RD #64 | Начальный адрес куда будем писать цифры |
| 3 | WR R1 | Сохраним |
| 6 | RDI #101 | Команда очистки буфера |
| 7 | OUT 1 | для клавиатуры |
| 8 | OUT 11 | и дисплея |
| 9 | RDI #102 | Команда сброса ошибки |
| 10 | OUT 1 | для клавиатуры |
| 11 | OUT 11 | и для дисплея |
| 12 | RDI #10 | Команда разрешения МВУ |
| 13 | OUT 1 | для клавиатуры без прерываний (по опросу) |
| 14 | OUT 11 | для монитора без автоувеличения адреса |
| 15 | RDI #103 | Команда установки посимвольного ввода |
| 16 | OUT 1 | для клавиатуры |
| 17 | LOOP:IN 2 | Читаем регистр состояния клавиатуры |
| 18 | DIV #10 | делим на 10, чтобы второй разряд готовности стал в первом разряде |
| 19 | WR R3 | Сохраним |
| 20 | DIV #2 | Разделим на 2 |
| 21 | MUL #2 | Умножим на 2 |
| 22 | SUB R3 | Отнимем сохранное |
| 23 | JZ LOOP | Если четное, то нет кода в буфере, идем на цикл |
| 24 | IN 0 | Считываем код с клавиатуры |
| 25 | WR R2 | Сохраним |
| 26 | SBI #48 | <0 |
| 27 | JS RESET | Игнорируем, на сброс клавиатуры |
| 28 | RD R2 | код |
| 29 | SBI #58 | >9 |
| 30 | JNS CMPRUS | Проверим на русские буквы |
| 31 | RD R1 | Цифру выводим в нижней части экрана, в R1 адрес, куда писать |
| 32 | OUT 13 | в регистр адреса |
| 33 | ADI #1 | инкремент адреса |
| 34 | WR R1 | Сохраним |
| 35 | RD R2 | код |
| 36 | OUT 10 | На дисплей! |
| 37 | JMP RESET | На сброс клавиатуры |
| 38 | CMPRUS:RD R2 | код, проверим на > 128 - там русские буквы |
| 39 | SBI #128 | Сравним со 128 |
| 40 | JS RESET | < 128 игнорируем |
| 41 | RD R0 | Русские буковки выводим в верхней части экрана, в R0 адрес, куда писать |
| 42 | OUT 13 | в регистр адреса |
| 43 | ADI #1 | инкремент адреса |
| 44 | WR R0 | Сохраним |
| 45 | RD R2 | код |
| 46 | OUT 10 | На дисплей! |
| 47 | RESET:RDI #101 | Команда сброса |
| 48 | OUT 1 | В клавиатуру |
| 49 | RDI #103 | Команда установки посимвольного ввода (сбрасывается при сбросе) |
| 50 | OUT 1 | В клавиатуру |
| 51 | JMP LOOP | На чтение состояния клавиатуры |