Оглавление

[Задания на лабораторные работы по курсу АВМиС(АПЭВМ). 2](#__RefHeading___Toc107824543)

[Лабораторная работа №1 " Последовательный порт " 2](#__RefHeading___Toc107824544)

[ЗАДАНИЕ 2](#__RefHeading___Toc107824545)

[Теоретические сведения 2](#__RefHeading___Toc107824546)

[Вопросы к защите 3](#__RefHeading___Toc107824547)

[Список рекомендуемой литературы 3](#__RefHeading___Toc107824548)

[Лабораторная работа №2. Программирование контроллера прерываний. 3](#__RefHeading___Toc107824549)

[Задание 3](#__RefHeading___Toc107824550)

[Теоретические сведения 3](#__RefHeading___Toc107824551)

[ВАРИАНТЫ заданий 4](#__RefHeading___Toc107824552)

[Вопросы к защите 4](#__RefHeading___Toc107824553)

[Список рекомендуемой литературы 5](#__RefHeading___Toc107824554)

[Лабораторная работа №3. Программирование системного таймера. 5](#__RefHeading___Toc107824555)

[Задание 5](#__RefHeading___Toc107824556)

[Теоретические сведения 5](#__RefHeading___Toc107824557)

[ВАРИАНТЫ заданий 6](#__RefHeading___Toc107824558)

[Вопросы к защите 7](#__RefHeading___Toc107824559)

[Список рекомендуемой литературы 7](#__RefHeading___Toc107824560)

[Лабораторная работа № 4. Программирование часов реального времени. 7](#__RefHeading___Toc107824561)

[Задание 7](#__RefHeading___Toc107824562)

[Теоретические сведения 8](#__RefHeading___Toc107824563)

[Вопросы к защите 9](#__RefHeading___Toc107824564)

[Список рекомендуемой литературы 9](#__RefHeading___Toc107824565)

[Лабораторная работа № 5. Программирование клавиатуры. 10](#__RefHeading___Toc107824566)

[Задание 10](#__RefHeading___Toc107824567)

[Теоретические сведения 10](#__RefHeading___Toc107824568)

[Вопросы к защите 11](#__RefHeading___Toc107824569)

[Список рекомендуемой литературы 11](#__RefHeading___Toc107824570)

[Лабораторная работа №6 11](#__RefHeading___Toc107824571)

[Защищенный и реальный режим процессора. Переход из одного режима в другой и обработка прерываний. 11](#__RefHeading___Toc107824572)

[Задание 11](#__RefHeading___Toc107824573)

[Лабораторная работа №7. Защищенный режим процессора. Мультизадачность. 12](#__RefHeading___Toc107824574)

[Задание 12](#__RefHeading___Toc107824575)

# Задания на лабораторные работы по курсу АВМиС(АПЭВМ).

Условия выполнения лабораторных работ:

* 1. Программирование всех устройств осуществляется через порты ввода/вывода.
  2. Любое задание на лабораторную работу может включать в себя набор некоторых условий, реализация которых *обязательна* для выполнения лабораторной работы.
  3. Операционная система выполнения лабораторной работы – любая (рекомендуется DOS, Lunux, Windows.
  4. Язык программирования – любой (рекомендуется C, С++ , ASM).
  5. Для допуска к экзамену необходимо выполнить 7 ЛР или 7,8.
  6. На выполнение и защиту ЛР 1-6 отводится 4 часа. На ЛР7 – 8 часов.
  7. Если ЛР сдана в срок, то оценка идет с коэффициентом 1, если просрочка равна одной ЛР (например, на выполнение ЛР1 затрачено более 4 часов и сдается она вместе с ЛР2), то предыдущая имеет коэффициент ½, если просрочка равна двум ЛР, то предыдущая имеет коэффициент ¼ и т.д

# Лабораторная работа №1 " Последовательный порт "

# ЗАДАНИЕ

Разработать  программный  модуль  реализации  процедуры  передачи (приема) байта информации через последовательный интерфейс между двумя портами.

Программа должна демонстрировать программное взаимодействие с последовательным интерфейсом с использованием следующих механизмов:

1. прямое взаимодействие с портами ввода-вывода (write, read)
2. использование BIOS прерывания 14h,
3. работа с COM-портомами через регистры как с устройствами ввода-вывода.

# Теоретические сведения

Последовательный порт или COM-порт - двунаправленный последовательный интерфейс, предназначенный для обмена байтовой информацией. Последовательный потому, что информация через него передаётся по одному биту, бит за битом (в отличие от параллельного порта). Наиболее часто для последовательного порта персональных компьютеров используется стандарт RS-232C.

Простейший алгоритм передачи данных через последовательный интерфейс состоит из следующих фаз:

–     инициализация порта;

–     передача данных;

–     прием данных;

–     анализ состояния порта.

Команды ввода/вывода микропроцессора:

* **IN AL,port8**- ввод байта в регистр AL из указанного порта;
* **IN AL,DX**- ввод байта в регистр AL из порта по адресу указанному в DX;
* **OUT port8,AL**- вывод байта из регистр AL указанного порта;
* **OUT DX,AL**- вывод байта из регистр AL порта по адресу указанному в DX;

Пример:*'записываем в LCR режим работы сом порта:*

*'8 бит всимволе,1 стоп бит, проверка паритета на четность,выдавать 0 в случае обрыва, DLAB=1*

*mov al,DBh 'записываем в AL значения для регистра LCR=DBh*

*out 3fBh,al'записываем данные в регистр UART LCR*

*'задаем скорость обмена 115 000 бит/сек DIM=00h, DLL=01h*

*mov al,01h*

*out 3f8h,al 'запись регистра DLL=01h*

*mov al,00h*

*out 03f9h,al 'запись регистра DIM=00h*

*'снимаем бит DLAB=1*

*mov al,5Bh 'DLAB=0*

*out 3fBh,al*

*'послать байт 03h в линию связи*

*mov al,03h*

*out 3f8h,al 'посылает байт 03h на скорости 115 000 бит/сек*

    Перед записью байта данных в регистр передатчика необходимо убедиться в том, что регистр хранения передатчика свободен, то есть убедиться в том, что передача предыдущего символа завершена. Признаком того, что регистр передатчика свободен, является установленный бит 5(THRE=1) регистра состояния линии LSR.  
    Аналогично тому как это делается при передаче данных, перед вводом символа из порта приемника необходимо убедиться в том, что бит 0 регистра LSR установлен (т.е. DR=1). Это означает, что символ принят из линии и находится в буферном регистре приемника.

# Вопросы к защите

1. Назначение сигналов СОМ порта по стандарту RS-232C.
2. Назовите регистры последовательных портов и их назначение.
3. Алгоритмы режимов синхронизации приема- передачи данных
4. Для чего используется режим диагностики и как включается?
5. Назовите преимущества последовательных портов.

# Список рекомендуемой литературы

1. Зубков С.В. Ассемблер для DOS, Windows и Unix.
2. Пирогов В.Ю. Assembler. Учебный курс.
3. https://www.softelectro.ru/rs232prog.html

# Лабораторная работа №2. Программирование контроллера прерываний.

# Задание

Написать резидентную программу выполняющую перенос всех векторов аппаратных прерываний ведущего и ведомого контроллера на пользовательские прерывания. При этом необходимо написать обработчики аппаратных прерываний, которые будут установлены на используемые пользовательские прерывания и будут выполнять следующие функции:

1. Выводить на экран в двоичной форме следующие регистры контроллеров прерывания (как ведущего, так и ведомого):

* регистр запросов на прерывания;
* регистр обслуживаемых прерываний;
* регистр масок.

При этом значения регистров должны выводиться всегда в одно и то же место экрана.

1. Осуществлять переход на стандартные обработчики аппаратных прерываний, для обеспечения нормальной работы компьютера.

# Теоретические сведения

Каждый момент времени центральный процессор может работать только с одним устройством. Циклический опрос каждого устройства с последующей обработкой запроса оказался неэффективным. Решение задачи оказался контроллер прерываний, который принимает запросы от устройств и в соответствии с приоритетом направляет их процессору, если прерывание от данного устройства не замаскировано (разрешено) в регистре масок. Если прерывание разрешено и устройство его запросило, то устанавливается соответствующий устройству бит в регистре запросов.

Контроллер прерываний состоит из двух микросхем, подключенных каскадно (ведущий и ведомый контроллеры), каждая из которых имеет по 8 линий прерываний (IRQ0-IRQ7, IRQ8-IRQ15). За каждой линией закреплено определенное устройство.

Когда процессор получает запрос на прерывание, он сохраняет свое текущее состояние и переключается на выполнение запрошенной операции. При этом устанавливается бит в регистре обслуживания (бит запроса сбрасывается). После обслуживания прерываний сбрасывается бит обслуживания, посылается сигнал EOI (endofinterrupt), процессор переключается на выполняемую ранее задачу.

Для доступа к контроллеру прерываний используются порты 20h и 21h (для ведущего), A0 иA1h (для ведомого).

Регистр масок доступен через порт 21h /A1h. Чтобы изменить определенный бит, нужно считать значение из этого регистра, изменить нужный бит, записать значение обратно.

Чтобы считать регистр запросов, его нужно выбрать (записать в 20h/A0h значение 0Ah), а затем считать содержимое из порта 20h/A0h.

Чтобы считать регистр обслуживания, его нужно выбрать (записать в 20h/A0h значение 0Bh), а затем считать содержимое из порта 20h/A0h.

Для резидентной программы понадобится следующий фрагмент кода:

unsigned far \*fp; //объявляем указатель

FP\_SEG (fp) = \_psp; // получаем сегмент

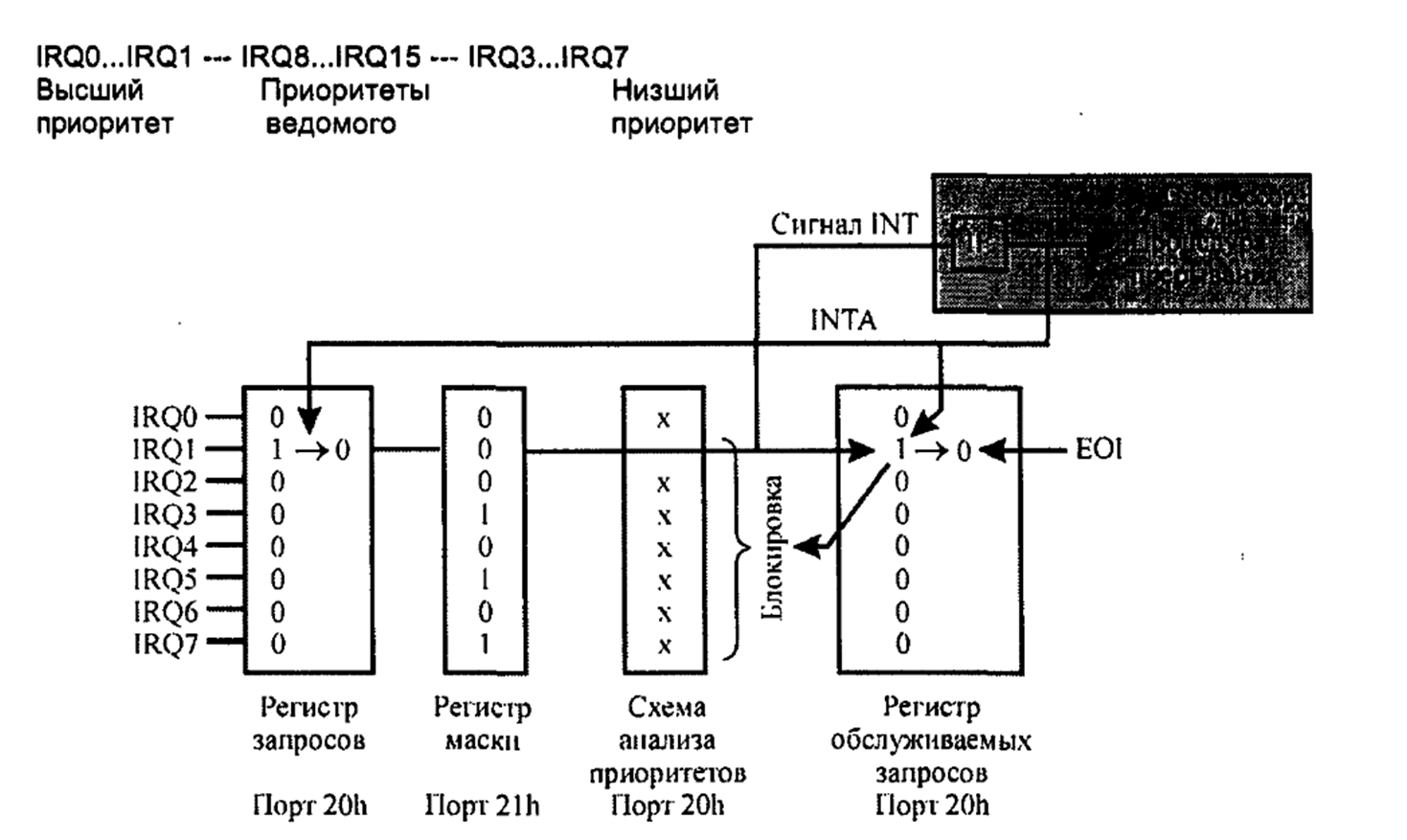
FP\_OFF (fp) = 0x2c; // и смещение сегмента данных с переменными среды

\_dos\_freemem(\*fp); //чтобы его освободить

\_dos\_keep(0,(\_DS -\_CS)+(\_SP/16)+1);//оставляем резидентной, указывая //первым параметром код завершения, а //вторым - объем памяти, который должен //быть зарезервирован для программы //после ее завершения

# ВАРИАНТЫ заданий

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер варианта** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| Базовый вектор | 60Н /  08H | 68Н /  08H | 70Н /  08H | 78Н /  08H | 80Н /  08H | 88Н /  08H | 90Н /  08H | 98Н /  08H | A0Н /  08H | A8Н /  08H |
| **Номер варианта** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| Базовый вектор | B0Н /  08H | B8Н /  08H | C0Н /  08H | C8Н /  08H | D0Н /  08H | D8Н /  08H | 08H / 60H | 08H / 68Н | 08H / 70Н | 08H / 78Н |
| **Номер варианта** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** |
| Базовый вектор | 08H / 80Н | 08H / 88Н | 08H / 90Н | 08H / 98Н | 08H / A0Н | 08H / A8Н | 08H / B0Н | 08H / B8Н | 08H / C0Н | 08H / C8Н |



Пройдя через схему анализа приоритетов, сигнал запроса прерывания поступает на вход регистра обслуживаемых запросов и дает разрешение на установку в 1 его бита (однако не устанавливает его). Одновременно сигнал поступает на вход INT микропроцессора. Микропроцессор регистрирует поступление сигнала INTлишь в том случае, если установлен флаг разрешения прерываний IF в регистре флагов. Таким образом, сброс флага IF командой cli запрещает все аппаратные прерывания (не затрагивая программные).

Микропроцессор, получив сигнал INT, отвечает на него выходным сигналом INTA (INTerrupt Acknowledge, подтверждение прерывания), который поступает в контроллер прерываний и выполняет там два действия: устанавливает бит регистра обслуживаемых запросов, разрешенный сигналом запроса прерывания, и сбрасывает аналогичный бит регистра запросов.

# Вопросы к защите

1. Для чего в программе обработчика прерываний необходимо указывать команду EOI?
2. Для чего используются команды CLI и STI?
3. Объяснить понятие «вектор прерывания».
4. Что делает команда IRET?
5. Что такое «вложенное прерывание»?
6. Назначение таблицы векторов прерываний.

# Список рекомендуемой литературы

1. В. Несвижский "Программирование аппаратных средств в Windows", с. 495 (486).

# Лабораторная работа №3. Программирование системного таймера.

# Задание

Задание состоит из двух частей. Первая часть общая для всех. Вторая часть по вариантам

***Первая часть (общее задание)***. Запрограммировать второй канал таймера таким образом, чтобы динамик компьютера издавал звуки.

***Вторая часть***:

1. Для всех каналов таймера считать слово состояния и вывести его на экран в двоичной форме.
2. Для всех каналов таймера рассчитать коэффициент деления (значение счетчика CE) и вывести его на экран в шестнадцатеричной форме.

# Теоретические сведения

Системный таймер - устройство, подключенное к линии запроса IRQ0 и вырабатывающее прерывание int8h. Таймер реализуется на микросхеме Intel 8253 или 8254 (Отечественные аналоги: К1810ВИ53 и К1810ВИ54).

Таймер состоит из трёх независимых каналов: 0 - отсчитывает текущее время после включения компьютера, 1- для работы с контроллером прямого доступа к памяти, 2 - связан с системным динамиком.

Каждый канал содержит регистры:

* состояния канала RS (8 разрядов);
* управляющего слова RSW (8 разрядов);
* буферный регистр OL (16 разрядов);
* регистр счетчика CE (16 разрядов);
* регистр констант пересчета CR (16 разрядов)

Каналы таймера подключаются к внешним устройствам при помощи линий:

* GATE - управляющий вход;
* CLOCK - вход тактовой частоты;
* OUT - выход таймера.

Регистр счетчика CE работает в режиме вычитания. Его содержимое уменьшается по спаду сигнала CLOCK при условии, что на вход GATE установлен уровень логической 1.

В зависимости от режима работы таймера при достижении счетчиком CE нуля тем или иным образом изменяется выходной сигнал OUT.

Буферный регистр OL предназначен для запоминания текущего содержимого регистра счетчика CE без остановки процесса счета. После запоминания буферный регистр доступен программе для чтения.

Регистр констант пересчета CR может загружаться в регистр счетчика, если это требуется в текущем режиме работы таймера.

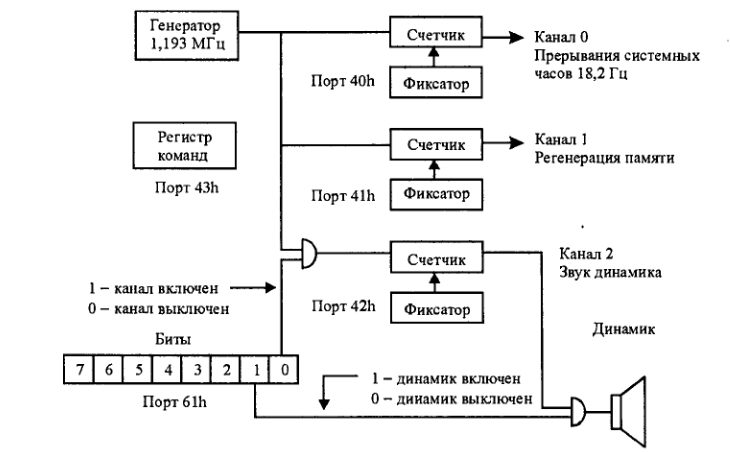
Возможны шесть режимов работы таймера, которые делятся на три типа:

Режимы 0, 4 - однократное выполнение функций.

Режимы 1, 5 - работа с перезапуском.

Режимы 2, 3 - работа с автозагрузкой.

Для подключения динамика используется порт 61h.



# ВАРИАНТЫ заданий

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вариант** | Частота, Гц (длительность, мс) | | | | | | | | |
|  | 659 (200) | 622 (200) | 659 (200) | 622 (200) | 659 (200) | 493 (200) | 622 (200) | 523 (200) | 440 (200) |
|  | 293 (200) | 392 (400) | 293 (200) | 392 (400) | 523 (200) | 659 (200) | 880 (400) | 784 (200) | 698 (200) |
|  | 784 (200) | 392 (200) | 440 (200) | 440 (200) | 784 (200) | 440 (200) | 493 (200) | 493 (200) |  |
|  | 329 (200) | 329 (100) | 329 (200) | 415 (400) | 523 (200) | 659 (200) | 587 (400) | 523 (200) | 659 (400) |
|  | 196 (400) | 261 (400) | 329 (400) | 196 (400) | 261 (400) | 329 (400) | 196 (400) | 261 (400) | 329 (400) |
|  | 392 (400) | 392 (400) | 392 (400) | 329 (1000) | 349 (400) | 349 (400) | 349 (400) | 293 (800) | 293 (1000) |
|  | 392  (400) | 329  (400) | 329  (400) | 392  (400) | 329  (400) | 329  (400) | 392  (400) | 349  (400) | 329  (400) |
|  | 329  (400) | 246  (400) | 246  (400) | 261  (400) | 293  (400) | 329  (400) | 392  (400) | 349  (400) |  |
|  | 392  (800) | 392  (400) | 293  (400) | 196  (400) | 196  (400) | 392  (400) | 392  (400) | 293  (400) | 196  (400) |
|  | 349  (600) | 392  (300) | 440  (600) | 349  (300) | 440  (300) | 440  (300) | 392  (300) | 349  (300) | 392  (300) |

# Вопросы к защите

1. Перечислите режимы работы системного таймера и их типы. Объясните, в чём различия.
2. Объясните назначение и использование каналов таймера в IBM PC.
3. Форматы управляющего слова для задания режима работы таймера.
4. Последовательность программирования системного таймера.
5. Как системный таймер используется для генерации звуков?
6. Какую информацию можно получить из слова состояния?
7. В каких пределах может быть частота звука, генерируемого с помощью таймера?
8. \*Нарисуйте временн**ы**е диаграммы работы таймера для 2 и 3 режимов.

# Список рекомендуемой литературы

В. Несвижский "Программирование аппаратных средств в Windows", с. 317(311), 335(328).

# Лабораторная работа № 4. Программирование часов реального времени.

# Задание

Задание по лабораторной работе состоит из двух частей. Первая часть общая для всех, вторая – по вариантам, выдаваемым преподавателем.

***Первая часть (общее задание)***. Написать программу, которая будет считывать и устанавливать время в часах реального времени. Считанное время должно выводиться на экран в удобочитаемой форме.

Условия выполнения лабораторной работы для первой части:

1. Перед началом установки новых значений времени необходимо считывать и анализировать старший байт регистра состояния 1 на предмет доступности значений для чтения и записи. Начинать операцию записи новых значений, можно только в случае, когда этот бит установлен в '0' – то есть, регистры часов доступны.
2. После проверки доступности регистров (пункт 1), необходимо отключить внутренний цикл обновления часов реального времени, воспользовавшись старшим битом регистра состояния 2. После окончания операции установки значений этот бит должен быть сброшен, для возобновления внутреннего цикла обновления часов реального времени.
3. Новые значения для регистров часов реального времени должны вводится с клавиатуры в удобном для пользователя виде.

***Вторая часть***

1. Используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний реализовать функцию задержки с точностью в миллисекунды (с возможностью изменения частоты генерации).

Условия выполнения данного варианта:

* 1. Задержка должна вводится с клавиатуры в миллисекундах в удобной для пользователя форме.
  2. После окончания периода задержки программа должна сообщить об этом в любой форме. При этом зависание компьютера не допускается
  3. Перебусмотреть возможность изменения интервала генерации задержки по умолганию (1/1024 Гц) на другое значение.

1. Используя аппаратное прерывания часов реального времени и режим будильника ЧРВ (4A использовать не желательно) реализовать функции программируемого будильника.
   1. Время будильника вводится с клавиатуры в удобной для пользователя форме.
   2. При срабатывания будильника программа должна сообщить об этом в любой форме. При этом зависание компьютера не допускается.

***Общие условия для всей лабораторной работы.***

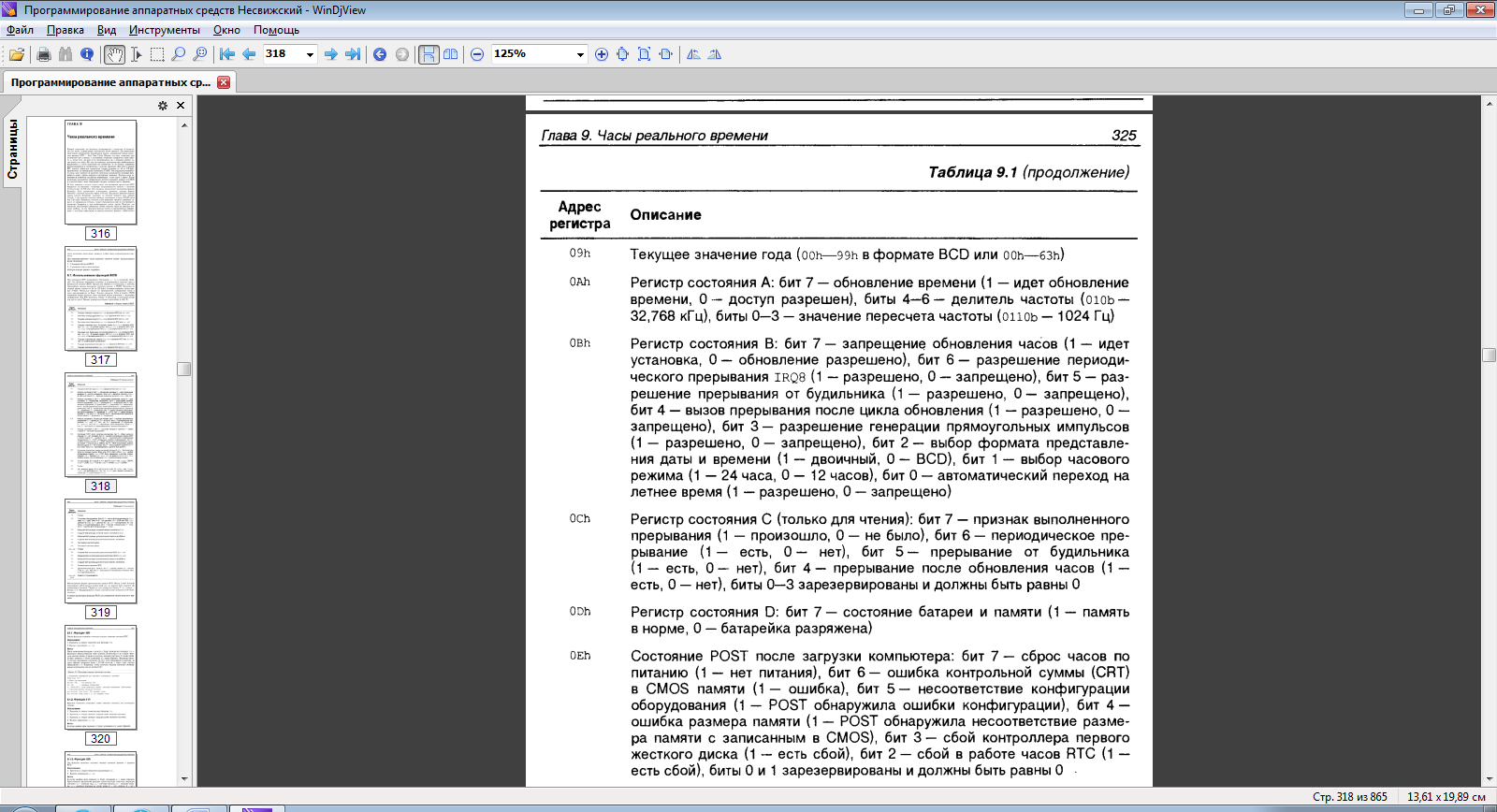
1. В программе должно быть реализовано меню, позволяющее выбрать тестируемый функционал (установка времени, считывание времени, задержка и т.д.)
2. Весь ввод/вывод данных с/на консоль должен выполняться в удобной для пользователя форме.
3. Зависание компьютера не допускается ни в ходе работы программы, ни после ее завершения.

# Теоретические сведения

Часы реального времени (Real Time Clock) - специальный модуль, используемый для непрерывного отсчёта времени. За счёт использования батарейки этот модуль работает даже когда компьютер выключен.

Для хранения данных в часах реального времени предусмотрено несколько регистров (данные записаны в формате BCD):

|  |  |
| --- | --- |
| **Адрес регистра** | **Описание значения** |
| 00h | Текущая секунда |
| 01h | Секунды будильника |
| 02h | Текущая минута |
| 03h | Минуты будильника |
| 04h | Текущий час |
| 05h | Часы будильника |
| 06h | Текущий день недели *(01-07, 01 - воскресенье)* |
| 07h | Текущий день месяца |
| 08h | Текущий месяц |
| 09h | Текущий год (последние 2 цифры) |



Для доступа к часам реального времени используются всего два порта: 70h и 71h. Порт 70h используется только для записи и позволяет выбрать адрес регистра в CMOS памяти. Порт 71h применяется как для записи, так и для чтения данных из указанного (через порт 70h) регистра CMOS. Оба порта являются 8-разрядными. Кроме того, бит 7 в порту 70h не относится к работе с RTC(ЧРВ), а управляет режимом немаскируемых прерываний (1 — прерывания запрещены). Если у вас будут проблемы с доступом к регистрам CMOS, следует запрещать прерывания (команда cli) перед началом работы и разрешать прерывания (команда sti) после. Но, как правило, этого не требуетcя.

Основное назначение часов - это непрерывный отсчет времени cуток даже в тот период, когда компьютер выключен, а также извещение процессора о наступлении определенного времени (режим будильника). Часы реального времени реализованы на базе микросхемы типа Motorola МС 146818 (RT/CMOS RAM Chip) и обеспечивают непрерывный отсчет времени при выключении питания на компьютере за счет аккумуляторной батареи. Эта микросхема включает также память, содержимое которой сохраняется при исключении питания. Аппаратура часов реального времени использует только первые 14 байт этой памяти, а остальные предназначены для сохранения конфигурации компьютера. Часы позволяют генерировать три типа прерываний по линии 0 второго контроллера прерываний (IRQ 8), что соответствует в IBM PC прерыванию 70h:

1. периодические прерывания с интервалом 976.562 мкс;
2. прерывание от будильника;
3. прерывание по окончанию обновления значения часов.

### Регистры часов

Как уже говорилось ранее, часы используют первые 14 байт памяти микросхемы типа Motorola MC146818, которые будем в дальнейшем называть регистрами часов. Чтобы считать любой памяти микросхемы, в том числе и данные из регистра часов необходимо выполнить следующие шаги:

1. записать адрес считываемого регистра в порт 70h;
2. считать данные регистра из порта 71h.

Например, следующие команды используются для чтения регистра 0 часов реального времени:

mov al,0 ; Записать адрес регистра

out 70h,al

jmp short $+2

in al,71h ; Считать регистр

Аналогично, для записи значения в регистр часов требуется занести адрес модифицируемого регистра в порт 70h, после чего вывести в порт 71h записываемое значение. Например:

mov al,0 ; Записать адрес регистра

out 70h,al

jmp short $+2

mov al,20h ; Занести новое значение

out 71h,al

Следует отметить, что чтение и запись данных порта 71h должны выполняться немедленно после записи номера регистра в порт 70h. Кроме того, в процессе операций ввода-вывода с регистрами часов должны быть запрещены прерывания, что позволит избежать модификации порта 70h процедурой обработки прерывания до чтения или записи данных.  
Адресация регистров часов приведена в таблице:

|  |  |
| --- | --- |
| Адрес регистра | Назначение регистра |
| 00h | Секунды |
| 01h | Секунды для будильника |
| 02h | Минуты |
| 03h | Минуты для будильника |
| 04h | Часы |
| 05h | Часы для будильника |
| 06h | День недели |
| 07h | Дата |
| 08h | Месяц |
| 09h | Год |
| 0Ah | Регистр состояния 1 |
| 0Bh | Регистр состояния 2 |
| 0Ch | Регистр состояния 3 |
| 0Dh | Регистр состояния 4 |
| 32h | Столетие |

Регистры времени 00h-09h и 32h содержат время в двоично-десятичном формате (например, дата 26 хранится в виде 26h). Столетие задастся в виде первых двух цифр полного номера года (например, 19h). На машинах PS/2 байт столетия расположен по адресу 37h.  
**Регистр состояния 1** (А)имеет следующий формат:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UIP | DV | | | RS | | | |

Бит *UIP* определяет момент обновления показаний часов и может быть установлен в 0 - часы доступны для чтения или - 1 - аппаратура выполняет обновление показаний часов. Для правильного чтения или записи значений часов эти операции необходимо выполнять, когда значение бита UIP равно 0.  
Биты *DV* задают значение частоты для обновления показаний часов. Для правильного функционирования часов данные биты должны быть установлены в значение 010, что соответствует частоте 32.768 Кгц. Это единственная частота, обеспечивающая правильное время в часах.  
Биты *RS* позволяют выбрать делитель выходной частоты. Для правильного функционирования эти биты должны быть усыновлены в значение 0110, что соответствует выходной частоте прямоугольных колебаний 1.024 Кгц и интервалу времени между периодическими прерываниями 976.562 мкс.  
Формат **регистра состояния 2 (B)** приведен ниже:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UPD | PIE | AIE | UIE | SQWE | DM | CF | DSE |

Бит *UPD* используется для обновления показания часов. Когда он устанавливается в 1, внутренний цикл обновления часов прекращается, и программа может инициализировать любой регистр часов. Для запуска часов надо установить этот бит в 0.   
Бит *PIE* управляет генерацией периодических прерываний t интервалом 976.562 мкс. Когда он установлен в 0, прерывания запрещены. При установке этого бита в 1, периодические прерывания разрешены.   
Аналогично, бит *AIE* задает, разрешены ли прерывания от будильника, а бит *UIE* разрешает или запрещает прерывания при завершении цикла обновления показаний часов.  
Бит *SQWE* разрешает или запрещает генерацию прямоугольных колебаний (см. биты RS в регистре состояния 1). Функция генерации прямоугольных колебаний не используется в компьютере и бит устанавливается в 0.   
*DM* определяет формат времени и даты. Если этот бит установлен в 0, то используется двоично-десятичный формат. При единичном значении бита данные представляются в двоичном виде. Для IBM PC этот бит установлен в 0.  
Бит *CF* задаст формат представления времени. Значение О указывает, что используется 12-ти часовой формат, а значение 1 - 24-х часовой формат. В компьютерах IBM PC используется 24-х часовой формат.  
Бит *DSE* установлен в 1, если в часы заносится летнее время и программы, пользующиеся показаниями часов, осуществляют пересчет времени. Обычно эта функция не используется, и бит установлен в 0.  
**Регистр состояния 3(C)** доступен только по чтению. Он содержит текущий статус прерывания от часов и имеет следующий формат:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT | PI | AI | UI | Резерв | | | |

Бит *INT* указывает на наличие прерывания (0 - нет прерывания, 1 - выработано прерывание).  
Биты *PI, АI и UI* уточняют тип прерывания, выработанного часами. Единичное значение бита PI указывает, что выработано периодическое прерывание. Аналогично, единичное значение бита AI сигнализирует о генерации прерывания от будильника, а бита UI - прерывания об окончании обновления показаний часов.   
Обработчик прерывания по линии IRQ8 должен прочитать регистр состояния 3. Иначе, следующее прерывание не будет выработано.  
**Регистр состояния 4** доступен только по чтению и имеет следующий формат:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VRB | Резерв | | | | | | |

В этом регистре состояния задействован только бит *VRB* который сигнализирует о состоянии аккумуляторной батареи. Единичное значение говорит о нормальном питании. Нулевое значение указывает, что батарея разряжена и данные в часах недостоверны.

# Вопросы к защите

Перечислите режимы работы системного таймера и их типы. Объясните, в чём различия.

Перечислите регистры системного таймера и их назначение.

# Список рекомендуемой литературы

В. Несвижский "Программирование аппаратных средств в Windows", с. 323(316).

\_102153768.

# 

# Лабораторная работа № 5. Программирование клавиатуры.

# Задание

Программируя клавиатуру помигать ее индикаторами. Алгоритм мигания произвольный. Условия реализации программы, необходимые для выполнения лабораторной работы:

1. Запись байтов команды должна выполняться только после проверки незанятости входного регистра контроллера клавиатуры. Проверка осуществляется считывание и анализом регистра состояния контроллера клавиатуры.
2. Для каждого байта команды необходимо считывать и анализировать код возврата. В случае считывания кода возврата, требующего повторить передачу байта, необходимо повторно, при необходимости – несколько раз, выполнить передачу байта. При этом повторная передача данных не исключает выполнения всех оставшихся условий.
3. Для определения момента получения кода возврата необходимо использовать аппаратное прерывания от клавиатуры.
4. Все коды возврата должны быть выведены на экран в шестнадцатеричной форме.

# Теоретические сведения

Для работы с клавиатурой используется 2 регистра: 60h – регистр данных, 64h – регистр состояния (статуса).

Для управления индикаторами через 60h отправляется код EDh. Затем маска, в соответствии с которой должны загореться индикаторы.

Формат «маски»:

|  |  |  |  |
| --- | --- | --- | --- |
| Биты 7-3 не используются | Caps Lock | Num Lock | Scroll Lock |

1 – включить

0 - выключить

Так клавиатура работает медленно, запись байтов команды должна выполняться только после проверки незанятости входного регистра контроллера клавиатуры. Проверка осуществляется считыванием и анализом регистра состояния контроллера клавиатуры (64h, бит 1).

На обработку каждого байта клавиатура отвечает кодом возврата. Если в регистре 60h находится код

FA – байт обработан успешно,

FE – произошла ошибка.

В случае ошибки передачу байта нужно повторить. Пересылка выполняется до 3 раз, если ошибка не исчезла, нужно вывести сообщение и выйти из программы.

При нажатии клавиши блок клавиатуры передает ее код сканирования центральному процессору. Когда клавиша отпускается, клавиатура снова передает ее код, но увеличенный на 128 (или шестнадцатеричное значение 80). То есть, коды для нажатия и отпускания клавиш различаются.

Когда выполняется какое-либо действие с клавишей (нажатие или отпускание), **процессор клавиатуры обнаруживает его и запоминает в буфере. Затем формируется прерывание с номером** **9.**

**Принципы работы клавиатуры**

Клавиатура выполнена, как правило, в виде отдельного устройства, подключаемого к компьютеру тонким кабелем. Малогабаритные компьютеры Lap-Top используют встроенную клавиатуру.

Внутри клавиатуры находится специализированная микросхема (контроллер клавиатуры), которая отслеживает нажатия на клавиши и посылает номер нажатой клавиши в персональный компьютер.

Если рассмотреть сильно упрощенную принципиальную схему клавиатуры, представленную на рисунке, можно заметить, что все клавиши находятся в узлах матрицы:

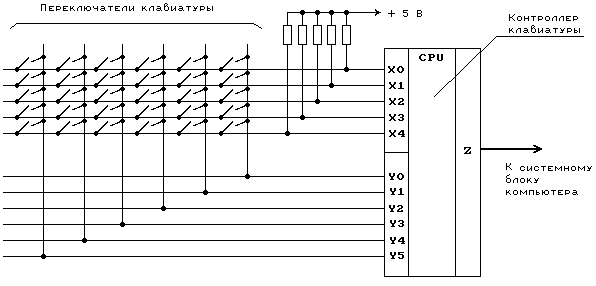


Рис.1. Упрощенная схема клавиатуры

Все горизонтальные линии матрицы подключены через резисторы к источнику питания +5 В. Клавиатурный компьютер имеет два порта - выходной и входной. Входной порт подключен к горизонтальным линиям матрицы (X0-X4), а выходной - к вертикальным (Y0-Y5).

Устанавливая по очереди на каждой из вертикальных линий уровень напряжения, соответствующий логическому 0, клавиатурный контроллер опрашивает состояние горизонтальных линий. Если ни одна клавиша не нажата, уровень напряжения на всех горизонтальных линиях соответствует логической 1 (т.к. все эти линии подключены к источнику питания +5 В через резисторы).

Если оператор нажмет на какую-либо клавишу, то соответствующая вертикальная и горизонтальная линии окажутся замкнутыми. Когда на этой вертикальной линии контроллер установит значение логического 0, то уровень напряжения на горизонтальной линии также будет соответствовать логическому 0.

Как только на одной из горизонтальных линий появится уровень логического 0, клавиатурный контроллер фиксирует нажатие на клавишу. Он посылает в персональный компьютер запрос на прерывание и номер клавиши в матрице. Аналогичные действия выполняются и тогда, когда оператор отпускает нажатую ранее клавишу.

Номер клавиши, посылаемый клавиатурным процессором, однозначно связан с распайкой клавиатурной матрицы и не зависит напрямую от обозначений, нанесенных на поверхность клавиш. Этот номер называется скан-кодом (Scan Code).

Слово scan ("сканирование"), подчеркивает тот факт, что клавиатурный контроллер сканирует клавиатуру для поиска нажатой клавиши.

Но программе нужен не порядковый номер нажатой клавиши, а соответствующий обозначению на этой клавише ASCII-код. Этот код не зависит однозначно от скан-кода, т.к. одной и той же клавише могут соответствовать несколько значений ASCII-кода. Это зависит от состояния других клавиш. Например, клавиша с обозначением '1' используется еще и для ввода символа '!' (если она нажата вместе с клавишей SHIFT).

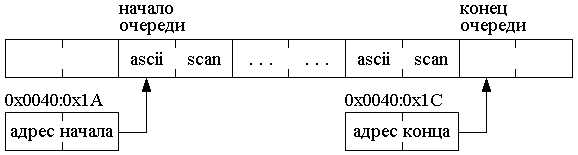
Поэтому все преобразования скан-кода в ASCII-код выполняются программным обеспечением внутри ПК. Как правило, эти преобразования выполняют модули BIOS. Для использования символов кириллицы эти модули расширяются клавиатурными драйверами.

Если нажать на клавишу и не отпускать ее, клавиатура перейдет в режим автоповтора. В этом режиме в центральный компьютер автоматически через некоторый период времени, называемый периодом автоповтора, посылается код нажатой клавиши. Режим автоповтора облегчает ввод с клавиатуры большого количества одинаковых символов.

**Структура буфера клавиатуры**

Клавиатура является основным средством ввода информации при работе на компьютере. В компьютерах IBM PC взаимодействие программ с клавиатурой организовано в асинхронном режиме. При нажатии клавиши на клавиатуре код нажатой клавиши поступает в буфер клавиатуры и хранится там до тех пор, пока программа не затребует его. Если скорость нажатия клавиш и скорость обработки нажатия клавиши программой отличаются, то или программа ожидает поступления кода очередной клавиши в буфер, или коды нажатых клавиш накапливаются в буфере клавиатуры. Хранение информации в буфере организовано в виде очереди (FIFO - First Input First Output), буфер клавиатуры находится в оперативной памяти компьютера и доступен для обычных операций чтения и записи.

Очередь буфера клавиатуры реализована следующим образом:



Каждой нажатой клавише в очереди соответствует ячейка размером в два байта, в первом байте записан ascii код, а во втором - scan код нажатой клавиши. Следует учитывать, что при считывании слова из буфера клавиатуры scan код оказывается в старшем байте, а ascii код в младшем байте слова в соответствии с порядком следования байтов принятом для IBM PC. В двух словах оперативной памяти по адресам 0x0040:0x1A и 0x0040:0x1C хранятся смещения адресов начала (головы) и конца (хвоста) очереди буфера клавиатуры. Следует обратить внимание, что эти смещения указаны относительно адреса 0040:0000h. Сам буфер размером 32 байта (16 слов) расположен в компьютере IBM PC/XT по адресу 0000h:041Eh и организован в виде кольцевой очереди. Следует отметить, что рост буфера (клавиши нажимаются, а символы из буфера не считываются) происходит в сторону хвоста.

В компьютерах моделей IBM PC/AT и IBM PS/2 расположение клавиатурного буфера задается содержимым двух слов памяти с адресами 0000h:0480h (смещение адреса начала буфера относительно 0040h) и 0000h:0482h (смещение конца буфера относительно 0040h). Обычно эти ячейки памяти содержат значения, соответственно, 001Eh и 003Eh, что соответствует расположению клавиатурного буфера в IBM PC/AT и IBM PS/2 его расположению в IBM PC/XT.

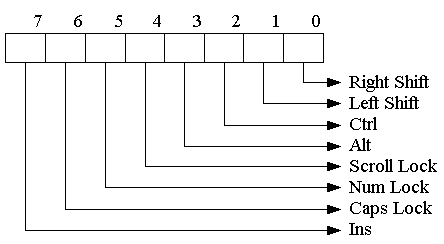
Указателями на начало и конец клавиатурного буфера обычно управляют обработчики прерываний INT 09h и INT 16h. Программа извлекает из буфера коды нажатых клавиш, используя различные функции прерывания INT 16h.

Заметим, что вы можете удалить все символы из буфера клавиатуры, установив оба указателя на начало буфера. Однако есть более предпочтительный способ с использованием, например, прерывания BIOS INT 16h.

Для чтения scan-кода последней нажатой клавиши (даже в том случае если буфер уже заполнен) можно использовать 60-й порт. При этом следует учитывать, что многократное обращение к порту будет возвращать один и тот же результат до тех пор, пока не будет нажата следующая клавиша на клавиатуре.

При переполнении внутреннего буфера клавиатуры или буфера, расположенного в области данных BIOS программа-обработчик прерывания INT 09h генерирует звуковой сигнал.

Кроме буфера клавиатуры и ячеек памяти с указателями головы и хвоста очереди с обработкой нажатия клавиш клавиатуры связано ещё некоторые элементов в области памяти данных BIOS. Одним из этих элементов является флаг состояния клавиатуры, располагающийся в области переменных BIOS по адресу 0000:0417h. Его размер составляет один байт, отдельные биты флага содержат информацию о состоянии клавиш Shift, Ctrl, Alt, Scroll Lock, Num Lock, Caps Lock и Ins. Соответствие перечисленных клавиш и битов флага показано на рисунке:



# Вопросы к защите

Что такое код возврата?

Как определить, является ли код кодом нажатия клавиши или ее отпускания?

Через какие порты происходит доступ к клавиатуре?

\_102153768.pd

# Список рекомендуемой литературы

В. Несвижский "Программирование аппаратных средств в Windows"

\_102153768.

# Лабораторная работа №6

# Защищенный и реальный режим процессора. Переход из одного режима в другой и обработка прерываний.

# Задание

Написать программу, которая выполняет следующие действия:

Переход из реального режима в защищенный.

Перехватывает аппаратное прерывание от клавиатуры и заданное аппаратное прерывание, в обработчике которого выполняет определенные действия. Прерывание от клавиатуры реализуют все, номер аппаратного прерывания задается по вариантам (два варианта).

Прерывание от таймера.

Прерывание от часов реального времени.

По наступлению определенного события выполняет обратный переход из защищенного режима в реальный и завершает свою работу.

Для прерываний от таймера и часов реального времени обработчик прерывания должен отслеживать количество вызовов прерывания и отсчитывать секунды, выводя их на экран. Количество секунд после которых выполняется обратный переход в реальный режим и выход из программы (то самое определенное событие) считывается с клавиатуры перед переходом в защищенный режим.

Для прерывания от клавиатуры необходимо считывать скан-коды клавиш и выводить их на экран. По нажатию определенной клавиши (любой на выбор студента) осуществляется обратный переход в реальный режим и выход из программы.

При выполнении данной лабораторной работы должны быть соблюдены следующие условия:

После завершения работы программы компьютер должен продолжать корректно функционировать. Зависания, перезагрузки и другие аналогичные «события» недопустимы.

Переход в защищенный режим процессора должен быть выполнен по алгоритму, используемому в процессорах начиная с 386. Переход в защищенный режим с использованием алгоритма для 286 процессора недопустим.

# Лабораторная работа №7. Защищенный режим процессора. Мультизадачность.

# Задание

Написать программу, которая выполняет следующие действия:

Выполняет переход из реального режима в защищенный.

Организует минимум три задачи, переключающиеся между собой.

Выполняет обратный переход из защищенного режима в реальный.

Каждая задача должна выполнять действие, по которому можно будет сделать вывод о том, что она работает (например, одна задача выводит символы справа налево и сверху вниз, другая организует счетчик, третья выводит символы слева на право и снизу вверх).

Возврат в реальный режим осуществляется по нажатию на “Esc”.

Переключение задач должно выполняться с использованием аппаратных прерываний по вариантам:

Прерывание от таймера.

Прерывание от часов реального времени.

Для прерываний от таймера и часов реального времени переключение задач осуществляется через определенные промежутки времени, вводимые с клавиатуры до перехода в защищенный режим.

При выполнении данной лабораторной работы должны быть выполнены следующие условия:

После завершения работы программы компьютер должен продолжать корректно функционировать. Зависания, перезагрузки и другие аналогичные «события» недопустимы.

Переход в защищенный режим процессора должен быть выполнен по алгоритму, используемому в процессорах начиная с 386. Переход в защищенный режим с использованием алгоритма для 286 процессора недопустим.