Оглавление

Задания на лабораторные работы по курсу АВМиС(АПЭВМ). 2
Лабораторная работа №1 " Последовательный порт " 2
ЗАДАНИЕ 2
Теоретические сведения 2
Вопросы к защите 3
Список рекомендуемой литературы 3
Лабораторная работа №2 "Команды MMX/XMM" 3
ЗАДАНИЕ 3
Теоретические сведения 3
ВАРИАНТЫ заданий 5
Вопросы к защите 5
Лабораторная работа №3. Программирование контроллера прерываний. 6
Задание 6
Теоретические сведения 6
ВАРИАНТЫ заданий 7
Вопросы к защите 7
Список рекомендуемой литературы 7
Лабораторная работа №4. Программирование системного таймера. 7
Задание 7
Теоретические сведения 8
ВАРИАНТЫ заданий 9
Вопросы к защите 10
Список рекомендуемой литературы 10
Лабораторная работа № 5. Программирование часов реального времени. 10
Задание 10
Теоретические сведения 11
Вопросы к защите 12
Список рекомендуемой литературы 13
Лабораторная работа № 6. Программирование клавиатуры. 13
Задание 13
Теоретические сведения 13
Вопросы к защите 14
Список рекомендуемой литературы 14
Лабораторная работа №7 14
Защищенный и реальный режим процессора. Переход из одного режима в другой и обработка прерываний. 14
Задание 14
Лабораторная работа №8. Защищенный режим процессора. Мультизадачность. 15
Задание 15

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, C++ , 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.

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-портом через регистры как с устройством 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.

Вопросы к защите

1. Назначение сигналов COM 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 "Команды MMX/XMM"

ЗАДАНИЕ

Создать приложение, которое выполняет заданные вычисления (в соответствии с вариантом) тремя способами:
1) с использованием команд MMX
2) на ассемблере, без использования команд MMX
3) на языке Си
После вычислений должны быть выведены время выполнения и результат для каждого случая.
Значения элементов матриц генерируются приложением (не вводятся с клавиатуры). Вычисления производятся многократно (например 1 млн раз). Размер матриц (векторов) кратен количеству элементов в регистре MMX.

Теоретические сведения

MMX (MultimediaExtensions- мультимедийное расширение) — коммерческое название дополнительного набора инструкций, выполняющих характерные для процессов кодирования/декодирования потоковых аудио/видео данных действия за одну машинную инструкцию. Впервые появился в процессорах Pentium MMX. Разработан в лаборатории Intel, в первой половине 1990-х.
SIMD (англ.SingleInstruction, MultipleData) - принцип компьютерных вычислений, позволяющий обеспечить параллелизм на уровне данных.
Для обработки данных и хранения промежуточных результатов в Pentium MMX используются восемь 64-разрядных регистров MM0..MM7, которые физически совмещены со стеком регистров математического сопроцессора. При выполнении любой из MMX-команд

происходит установка «режима MMX» с отметкой этого в слове состояния сопроцессора (FPU TagWord). С этого момента стек регистров сопроцессора рассматривается как набор MMX-регистров; завершает работу в режиме MMX команда EMMS (EndMultiMediaState). С одной стороны, такая реализация позволила избежать проблем совместимости с механизмами переключения контекста в существующих операционных системах, поскольку число регистров процессора, и, следовательно, код, выполняющий их сохранение и восстановление, не изменились. С другой стороны, переход между режимами занимает значительное время, и совмещение, например, в одном цикле команд сопроцессора с командами MMX может не только не ускорить, а даже существенно замедлить выполнение программы. Поэтому при необходимости для достижения наилучших результатов рекомендуется группировать эти команды отдельно друг от друга.
Команды технологии MMX работают с 64-разрядными целочисленными данными, а также с данными, упакованными в группы (векторы) общей длиной 64 бита. Такие данные могут находиться в памяти или в восьми MMX-регистрах.
Команды технологии MMX работают со следующими типами данных:

- упакованные байты (восемь байтов в одном 64-разрядном регистре) (англ. *packedbyte*);
- упакованные слова (четыре 16-разрядных слова в 64-разрядном регистре) (*packedword*);
- упакованные двойные слова (два 32-разрядных слова в 64-разрядном регистре) (*packeddoubleword*);
- 64-разрядные слова (*quadword*).

MMX-команды имеют следующий синтаксис: `instruction dest, src`. Здесь `instruction` - имя команды, `dest` - выходной операнд, `src` - входной операнд.
Большинство команд имеют суффикс, который определяет тип данных:

- B, W, D, Q указывают тип данных. Если в суффиксе есть две из этих букв, первая соответствует входному операнду, а вторая — выходному.

SSE (англ.StreamingSIMDExtensions) - это SIMD набор инструкций, разработанный Intel. Технология SSE позволяла преодолеть проблему одновременного использования сопроцессора (его регистры использовались для MMX и работы с вещественными числами).
SSE включает в архитектуру процессора восемь 128-битных регистров (xmm0 до xmm7), каждый из которых трактуется как 4 последовательных значения с плавающей точкой одинарной точности. SSE включает в себя набор инструкций, который производит операции со скалярными и упакованными типами данных.
Преимущество в производительности достигается в том случае, когда необходимо произвести одну и ту же последовательность действий над разными данными.

ВАРИАНТЫ заданий

1. Вычитание матриц.
2. Вычисление матричного выражения A = B + kC, где A, B, C – матрицы, k – скалярный множитель.
3. Нахождение суммы квадратов всех элементов матрицы.
4. Сформировать из двух массивов "максимальный" (результат[i] = max(массив1[i], массив2[i])). *pmaxub, pmaxsb, pmaxuw, pmaxsw.
5. Копирование элементов одной матрицы в другую.
6. Нахождение суммы всех элементов матрицы.
7. Умножение матриц.
8. Выполнить операцию побитового И для каждой пары соответствующих элементов.
9. Нахождение сумм по столбцам матрицы.
10. Нахождение сумм по строкам матрицы.
11. Сложение матриц.
12. Сформировать массив средних значений (результат[i] = avg(массив1[i], массив2[i])) *pavgb, pavgw
13. Сформировать из двух массивов "минимальный" (результат[i] = min(массив1[i], массив2[i])). *pminub,pminsb,pminuw,pminsw
14. Выполнить операцию побитового ИЛИ для каждой пары соответствующих элементов.
15. Выполнить операцию побитового исключающего ИЛИ для каждой пары соответствующих элементов.

Вопросы к защите

1. На каком главном принципе основана технология MMX?
2. С какими типами данных работают MMX команды?
3. Какая команда обеспечивает переход процессора из режима исполнения MMX-команд в режим исполнения обычных команд с плавающей запятой?
4. Что делает команда movq mm0, mm1?
5. Назначение команды EMMS.
6. Что такое SIMD команды?
7. Назначение команд MMX (где они могут применяться).
8. *Какое максимальное ускорение можно получить, используя MMX-команды?

Список рекомендуемой литературы

http://www.codenet.ru/progr/optimize/mmx.php
http://www.club155.ru/x86cmdsimd-41
http://miu.by/rus/kaf_ep/kaf_download

Лабораторная работа №3. 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.
2. 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	60H / 08H	68H / 08H	70H / 08H	78H / 08H	80H / 08H	88H / 08H	90H / 08H	98H / 08H	A0H / 08H	A8H / 08H
Номер варианта	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Базовый вектор	B0H / 08H	B8H / 08H	C0H / 08H	C8H / 08H	D0H / 08H	D8H / 08H	08H / 60H	08H / 68H	08H / 70H	08H / 78H
Номер варианта	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Базовый вектор	08H / 80H	08H / 88H	08H / 90H	08H / 98H	08H / A0H	08H / A8H	08H / B0H	08H / B8H	08H / C0H	08H / C8H

Вопросы к защите

9. Для чего в программе обработчика прерываний необходимо указывать команду EOI?
10. Для чего используются команды CLI и STI?
11. Объяснить понятие «вектор прерывания».
12. Что делает команда IRET?
13. Что такое «вложенное прерывание»?
14. Назначение таблицы векторов прерываний.

Список рекомендуемой литературы

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

Лабораторная работа №4. Программирование системного таймера.

Задание

Задание состоит из двух частей. Первая часть общая для всех. Вторая часть по вариантам. Варианты выдаются преподавателем после того как студенты разделятся на бригады.

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

Вторая часть (два варианта) :

1. Для всех каналов таймера считать слово состояния и вывести его на экран в двоичной форме.
2. Для всех каналов таймера рассчитать коэффициент деления 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.

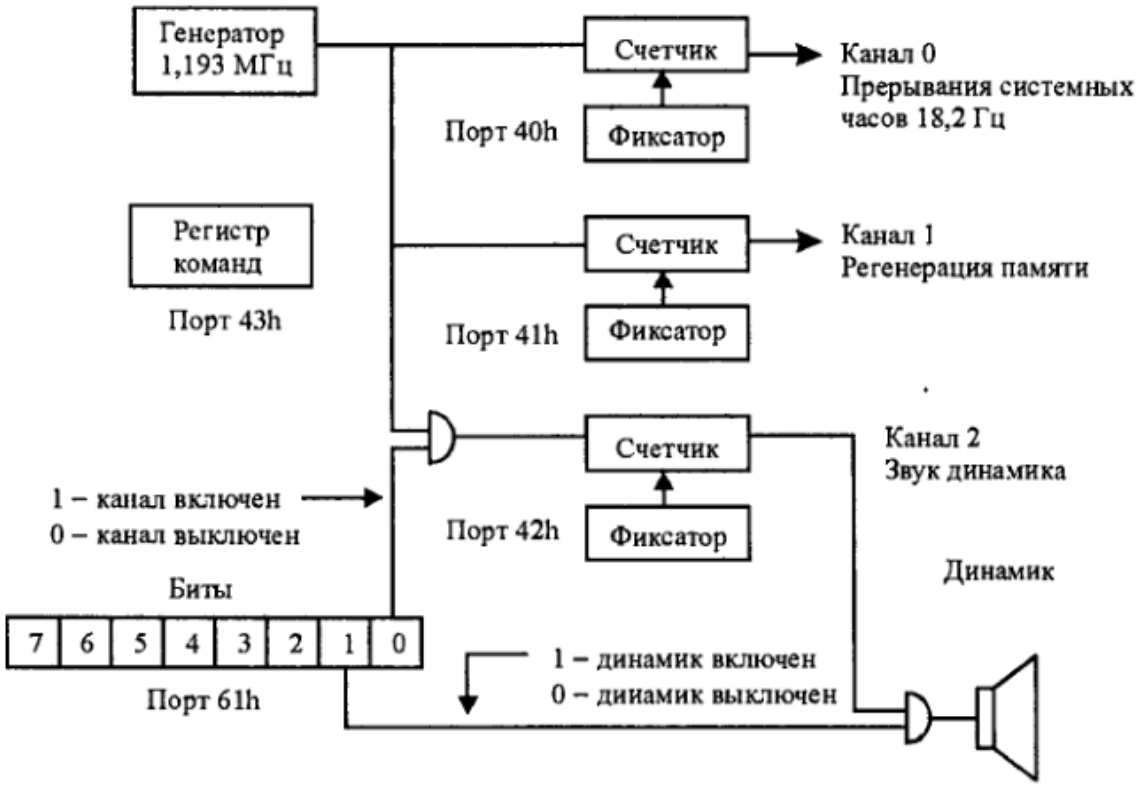
В зависимости от режима работы таймера при достижении счетчиком CE нуля тем или иным образом изменяется выходной сигнал OUT.

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)								
1.	659 (200)	622 (200)	659 (200)	622 (200)	659 (200)	493 (200)	622 (200)	523 (200)	440 (200)
2.	293 (200)	392 (400)	293 (200)	392 (400)	523 (200)	659 (200)	880 (400)	784 (200)	698 (200)
3.	784 (200)	392 (200)	440 (200)	440 (200)	784 (200)	440 (200)	493 (200)	493 (200)	
4.	329 (200)	329 (100)	329 (200)	415 (400)	523 (200)	659 (200)	587 (400)	523 (200)	659 (400)
5.	196 (400)	261 (400)	329 (400)	196 (400)	261 (400)	329 (400)	196 (400)	261 (400)	329 (400)
6.	392 (400)	392 (400)	392 (400)	329 (1000)	349 (400)	349 (400)	349 (400)	293 (800)	293 (1000)
7.	392 (400)	329 (400)	329 (400)	392 (400)	329 (400)	329 (400)	392 (400)	349 (400)	329 (400)
8.	329 (400)	246 (400)	246 (400)	261 (400)	293 (400)	329 (400)	392 (400)	349 (400)	
9.	392 (800)	392 (400)	293 (400)	196 (400)	196 (400)	392 (400)	392 (400)	293 (400)	196 (400)
10.	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 временны**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 № 5. 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 (два варианта).

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

Условия выполнения данного варианта:

1.1. Задержка должна вводится с клавиатуры в миллисекундах в удобной для пользователя форме.

1.2. После окончания периода задержки программа должна сообщить об этом в любой форме. При этом зависание компьютера не допускается.

2. Используя аппаратное прерывания часов реального времени и режим будильника реализовать функции программируемого будильника.

2.1. Время будильника вводится с клавиатуры в удобной для пользователя форме.

2.2. 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)

`0Ah` Регистр состояния A: бит 7 — обновление времени (1 — идет обновление времени, 0 — доступ разрешен), биты 4—6 — делитель частоты (`010b` — 32,768 кГц), биты 0—3 — значение пересчета частоты (`0110b` — 1024 Гц)

`0Bh` Регистр состояния B: бит 7 — запрещение обновления часов (1 — идет установка, 0 — обновление разрешено), бит 6 — разрешение периодического прерывания `IRQ8` (1 — разрешено, 0 — запрещено), бит 5 — разрешение прерывания от будильника (1 — разрешено, 0 — запрещено), бит 4 — вызов прерывания после цикла обновления (1 — разрешено, 0 — запрещено), бит 3 — разрешение генерации прямоугольных импульсов (1 — разрешено, 0 — запрещено), бит 2 — выбор формата представления даты и времени (1 — двоичный, 0 — BCD), бит 1 — выбор часового режима (1 — 24 часа, 0 — 12 часов), бит 0 — автоматический переход на летнее время (1 — разрешено, 0 — запрещено)

`0Ch` Регистр состояния C (только для чтения): бит 7 — признак выполненного прерывания (1 — произошло, 0 — не было), бит 6 — периодическое прерывание (1 — есть, 0 — нет), бит 5 — прерывание от будильника (1 — есть, 0 — нет), бит 4 — прерывание после обновления часов (1 — есть, 0 — нет), биты 0—3 зарезервированы и должны быть равны 0

`0Dh` Регистр состояния D: бит 7 — состояние батареи и памяти (1 — память в норме, 0 — батарейка разряжена)

`0Eh` Состояние POST после загрузки компьютера: бит 7 — сброс часов по питанию (1 — нет питания), бит 6 — ошибка контрольной суммы (CRT) в CMOS памяти (1 — ошибка), бит 5 — несоответствие конфигурации оборудования (1 — POST обнаружила ошибки конфигурации), бит 4 — ошибка размера памяти (1 — POST обнаружила несоответствие размера памяти с записанным в CMOS), бит 3 — сбой контроллера первого жесткого диска (1 — есть сбой), бит 2 — сбой в работе часов RTC (1 — есть сбой), биты 0 и 1 зарезервированы и должны быть равны 0

9.2. Использование портов

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, 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 требуется. Пример получения числа установленных флоппи-дисководов показан в листинге 9.5.

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).

Лабораторная работа № 6. 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.**

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?

Список рекомендуемой литературы

В. Несвижский "Программирование аппаратных средств в Windows"

Лабораторная работа №7

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 №8. 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.