МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ВОЛОГОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ЭВМ И ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Факультет электроэнергетический

Направление подготовки: 230100.62 – Информатика и вычислительная техника

Профиль подготовки: Программное обеспечение средств вычислительной техники и автоматизированных систем

ВОЛОГДА
2015

УДК 621.374.1

ЭВМ и ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА: методические указания для выполнения лабораторных работ. – Вологда: ВоГУ, 2015. – 64 с.

В методических указаниях приведены все необходимые сведения для проведения лабораторных работы по курсу «ЭВМ и периферийные устройства».

Утверждено редакционно-издательским советом ВоГУ

Составитель: В.Б.Анкудинов, канд. техн. наук, доцент

Рецензент: А.Н.Андреев, канд. техн. наук, доцент

ЛАБОРАТОРНАЯ РАБОТА №1

АРИФМЕТИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Часть 1. ЛОГИЧЕСКИЕ ПЕРЕМЕННЫЕ И ЛОГИЧЕСКИЕ ФУНКЦИИ

Цель лабораторной работы состоит в получении навыков по вычислению логических функций и их составлению по заданной таблице истинности с последующей схемной реализацией на простейших (элементарных) логических компонентах.

ПРИМЕЧАНИЕ: ***Для правильного выполнения приложений лабораторных работ необходимо в среде табличного процессора FAR MENEDGER вначале загрузить программу CLARUS, а затем запускать приложение соответствующей части лабораторной работы.***

1.ПРОСТЕЙШИЕ ЛОГИЧЕСКИЕ ОПЕРАЦИИ

ДИЗЪЮНКЦИЯ (логическое сложение, операция ИЛИ).

Дизъюнкция n переменных равна 1, если хотя бы одна переменная (аргумент) равна единице.

$$Y = X1 + X2 + ... + Xn.$$

КОНЪЮНКЦИЯ (логическое перемножение, операция И).

Конъюнкция n переменных равна 1, если все переменные (аргументы) равны 1.

$$Y = X1 * X2 * ... * Xn.$$

ИНВЕРСИЯ (логическое отрицание, операция НЕ).

Инверсия переменной (аргумента), равной 0, равняется 1 и наоборот.

$$Y = \overline{X}.$$

ИСКЛЮЧАЮЩЕЕ ИЛИ (сумма по модулю 2, операция неравнозначность).

$$Y = X1*\overline{X2} + \overline{X1}*X2.$$

На экране приводятся графические обозначения элементов, реализующих простейшие логические функции, с показом результатов работы при заданных значениях логических аргументов. Предлагаются схемы, результат работы которых следует вычислить и ввести в программу в предлагаемой форме.

Схемы и результаты их работы зарисовать.

2.ОСНОВЫ СИНТЕЗА КОМБИНАЦИОННЫХ СХЕМ

На экране показываются действия правил написания логической функции по таблице истинности для трехвходовой комбинационной схемы (пороговой ячейки).

По полученной логической функции в виде совершенной дизъюнктивной нормальной формы (СДНФ) составить схему, реализующую эту функцию.

Задания по вариантам:
где К-номер комбинации аргументов; Х- аргументы логические; У- логическая функция.

Вариант 1

К	Х3	Х2	Х1	У
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Вариант 2

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Вариант 3

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Вариант 4

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Вариант 5

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Вариант 6

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Вариант 7

К	Х3	Х2	Х1	У
0	0	0	0	
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Вариант 8

К	Х3	Х2	Х1	У
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Вариант 9

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Вариант 10

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Вариант 11

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Вариант 12

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Вариант 13

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Вариант 14

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Вариант 15

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Вариант 16

К	Х3	Х2	Х1	У
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Вариант 17

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Вариант 18

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Вариант 19

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Вариант 20

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Вариант 21

К	Х3	Х2	Х1	У
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

Вариант 22

К	Х3	Х2	Х1	У
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

3. РАБОТА В ЛАБОРАТОРИИ

3.1. Запустить исполняемый модуль LOGF.EXE.

3.2. Изучите предлагаемый материал. Ответьте на поставленные программой вопросы.

3.3. По номеру списка в журнале выбрать вариант задания на синтез комбинационной схемы и по правилам синтеза реализовать схему. Минимизацию делать необязательно.

4. СОДЕРЖАНИЕ ОТЧЕТА

В отчет следует включить:

4.1. Теоретические основы алгебры логики, с которой познакомились в первой части лабораторной работы.

4.2. Графическое обозначение логических элементов, которые рассмотрели при выполнении лабораторной работы.

4.3. Показать синтез комбинационной схемы по заданию.. Изобразить схему на элементах, изученных в части 1.

Часть 2. ОСНОВЫ КОДИРОВАНИЯ ЦИФРОВОЙ ИНФОРМАЦИИ

Цель данной части лабораторной работы состоит в получении навыков кодирования цифровой информации, переводе чисел из одной системы счисления в другую.

1. ЗАПИСЬ ЧИСЕЛ В ПОЗИЦИОННЫХ СИСТЕМАХ СЧИСЛЕНИЯ

Для выполнения части 2 необходимо запустить исполняемый модуль KODL.EXE.

На экране дается краткая информация о позиционных системах счисления, на примерах наиболее часто используемых в практике применения ЭВМ - восьмиричной, двоичной, шестнадцатиричной и двоично-десятичной системах счисления.

При знакомстве с материалом этого раздела, необходимо записать результаты Ваших решений в предлагаемых примерах.

Также в отчете требуется сделать запись с экрана правил преобразования любых чисел, веса разрядов в различных системах счисления.

2. ЗАДАНИЯ КОДОВ ДЛЯ ПРЕОБРАЗОВАНИЯ

В этом разделе лабораторной работы даны три варианта (A, B, C) для закрепления изученного материала по кодированию информации в ЭВМоска: По образцу, показанному на экране, ввести свои результата расчетов по вариантам в предложенных примерах. Все варианты должны быть выполнены, иначе лабораторная работа считается не выполненной.

3. РАБОТА В ЛАБОРАТОРИИ

3.1. Запустить исполняемый модуль KODL.EXE и изучить теоретические вопросы кодирования, предлагаемые программой.

3.2. Осуществить решение задач, предложенных программой.

3.3. Предложить свои примеры решения по образцу из программы.

4. СОДЕРЖАНИЕ ОТЧЕТА

В отчет следует включить:

4.1. Теоретические основы кодирования цифровой информации, с которой познакомились в первой части лабораторной работы.

4.2. Результаты проверенных решений примеров.

4.3. Выводы.

Часть 3. ОСНОВНЫЕ ОПЕРАЦИИ С ЦИФРОВЫМИ ДАННЫМИ

Цель третьей части лабораторной работы в получении навыков осуществления различных операций с цифровыми данными (кодами). Приобретение навыков выполнения примеров, содержащих как логические, так и арифметические операции.

1. ПРОСТЕЙШИЕ ЛОГИЧЕСКИЕ (ПОБИТНЫЕ) ОПЕРАЦИИ

Для выполнения этой части лабораторной работы необходимо запустить исполняемый модуль OWDC.EXE.

Кратко представлены (в форме напоминания) элементы алгебры логики из первой части лабораторной работы, чтобы был более понятен смысл побитной (поразрядной) обработки цифровой информации. Приводятся примеры с выполнением логических операций над пятиразрядными двоичными данными. Правила выполнения этих операций следует записать для отчета.

2. ПРАВИЛА ВЫПОЛНЕНИЯ ОСНОВНЫХ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ В ЭВМ

На экране показаны правила и действия устройств ЭВМ по выполнению арифметических операций. Следует записать этот материал для отчета [2].

3. ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

В заключении приводятся задания по вариантам, которые требуется выполнить сначала на бумаге, а затем ввести результаты в программу. При получении положительного ответа работа считается выполненной.

4. СОДЕРЖАНИЕ ОТЧЕТА

В отчет следует включить:

4.1. Все рассмотренные в лабораторной работе примеры побитной и арифметической обработки цифровых данных.

4.2. Результаты проверенных решений примеров.

4.3. Выводы.

ЛАБОРАТОРНАЯ РАБОТА №2

ЭЛЕКТРОННЫЕ УСТРОЙСТВА ЭВМ

Цель работы: изучение электронных устройств, применяемых для создания процессоров и системных элементов среды ЭВМ: простейшие логические элементы, устройства памяти, схемы управления направлением передачи информации по информационным шинам, схемы реализации портов ввода/вывода.

1. ОБЩИЕ ПОЛОЖЕНИЯ

Лабораторная работа представляет из себя программный продукт, предназначенный для изучения большого количества отечественных интегральных схем, применяемых в ЭВМ, как для построения процессоров, так и для реализации систем на базе ЭВМосква: Приложение называется CHE.EXE (этот учебник разработан на кафедре ЭВМ НИИТа при содействии Рос НПС «Купол». Авторы: Грамолин В., Павлов А., Павлова А., Ляндрес В.). После запуска приложения из главного меню необходимо познакомиться с руководством. (Если программа не работает нормально, можно провести запуск через Far Menetger, запустив предварительно CLARUS, как в предыдущей работе).

Руководство предназначено для обучения правилам работы с электронным учебником. Все сообщения подсистемы будут выводиться в окнах с надписью "Руководство". В нижней части рамки окна всегда находится подсказка, т.е. клавиша, которую необходимо нажать или строка меню, которую необходимо выбрать для перехода к следующей теме Руководства. Весь процесс обучения организован в виде игры, во время которой необходимо искать неисправности в работе цифровых микросхем и собранных из них модулей. Каждая обнаруженная неисправность принесет несколько очков. За ошибочные ответы начисляются штрафные очки. На каждой микросхеме можно набрать не более 3 штрафных очков, в противном случае происходит возврат в режим исправной микросхемы для дальнейшего изучения. Для победы необходимо изучить все относящиеся к текущей игре микросхемы и обнаружить в каждой из них по 4 неисправности. Счёт, набранный по окончании игры, поможет оценить свои знания.

ПРИМЕЧАНИЕ: указанные правила не являются баллами для оценки Ваших знаний у преподавателя, а только для самообразования. Если компонент Вам хорошо понятен, то можно переходить к изучению следующего.

После знакомства с основными правилами работы с учебником выберите в меню раздел «Открыть учебник». Откроется главное меню, вид которого показан на рис.1.

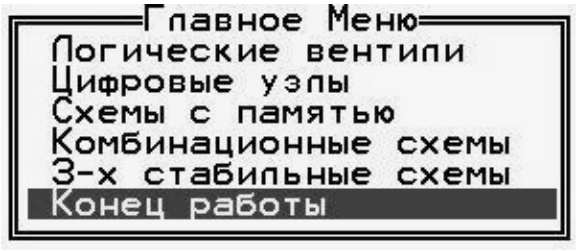

Рис.1. Главное меню учебника

Далее необходимо выбрать (по очереди) каждый раздел меню и познакомиться с каждой интегральной микросхемой, представленной в разделе. Пример меню раздела «Логические вентили» на рис.2.

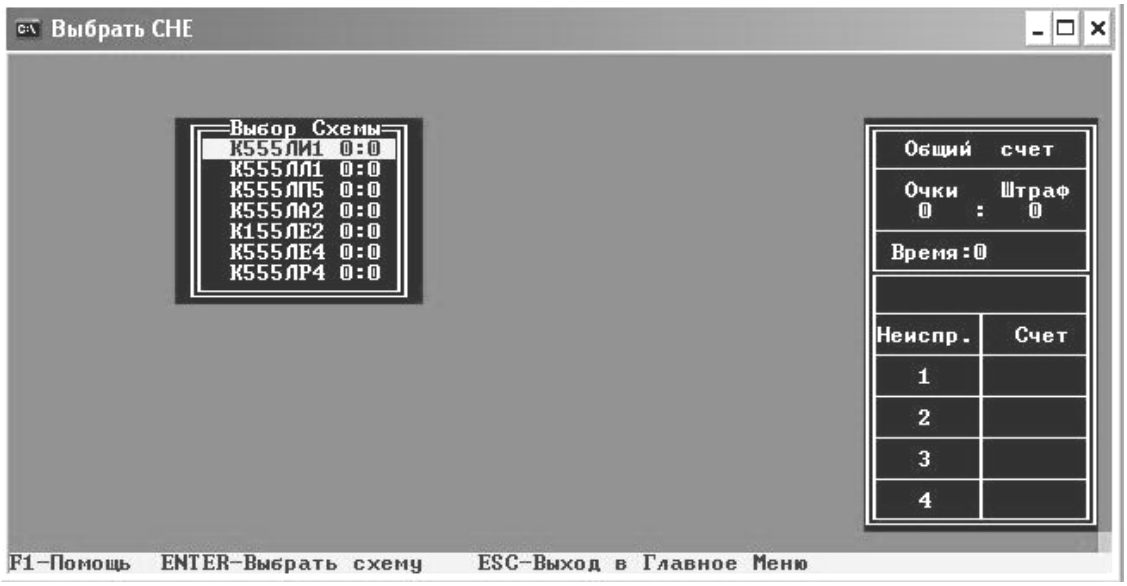

Рис. 2. Начало раздела «Логические вентили»

Выбрав из списка микросхему К555ЛИ1, появится информация о составе микросхемы (рис.3.). На входах имеются пронумерованные прямоугольники, имитирующие цветом состояние входа: коричневый цвет – состояние 1, белый – 0. Для изменения состояния входа необходимо нажать соответствующую клавишу. При первом обращении к изучаемой микросхеме она представлена в исправном состоянии. Для получения информации о микросхеме необходимо нажать клавишу F1. После освоения логики работы микросхемы следует приступить к закреплению освоенного материала, нажав клавишу ENTER, вводится первая ошибка в микросхему. Задача студента, манипулируя состоянием входов микросхемы, определить неисправность в микросхеме. В программе CHE.EXE для закрепления знаний по изучаемой микросхеме предлагается поиск ошибки повторить 4 раза.

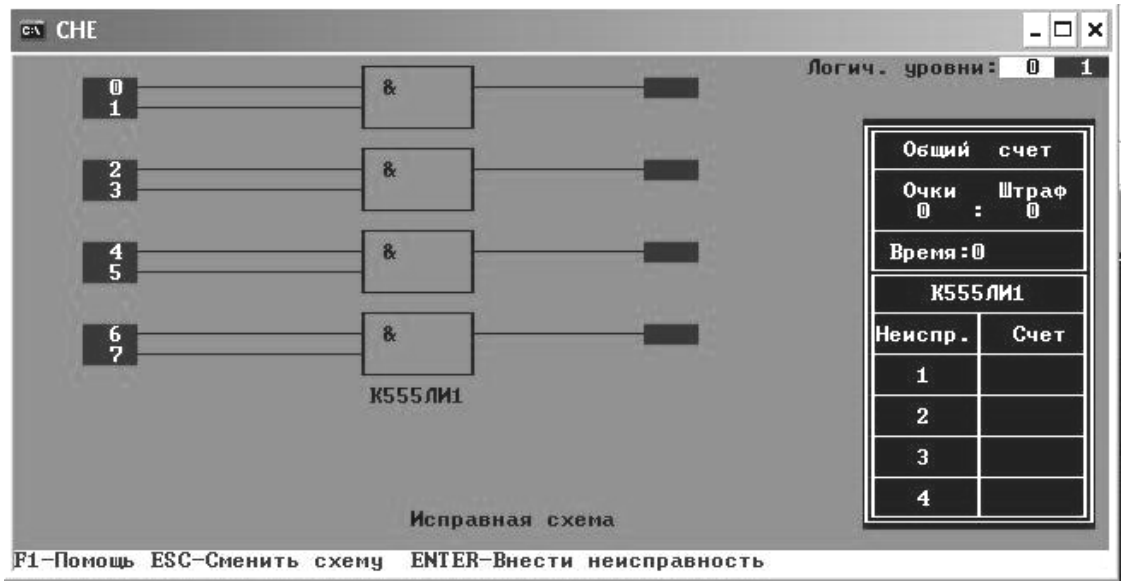

Рис.3. Экран для изучения микросхемы К555ЛИ1

Дальнейшие действия выполнить по каждой микросхеме, входящей в состав тем из главного меню. Все необходимые пояснения приведены в поле экрана изучаемой микросхемы.

2. РАБОТА В ЛАБОРАТОРИИ

2.1. Изучить общие положения устройства учебника.

2.2. Изучить все микросхемы, входящие в разделы учебника.

2.3. По каждой изучаемой микросхеме дать краткое её предназначение.

2.4. После изучения микросхем, производящих логические, арифметические действия привести примеры выполнения операций (дешифраторы, сумматоры, арифметико-логическое устройство).

3.СОДЕРЖАНИЕ ОТЧЁТА

3.1. По каждому разделу учебника дать кратко характеристику состава микросхем.

3.2. Привести условное обозначение изучаемых микросхем (**прямоугольники модели микросхем приводить не нужно**).

3.3. Назначение изучаемой микросхемы, таблицу состояния, примеры операций, которые можно реализовать при помощи изученной микросхемы.

3.4. Выводы (перечислить разделы и состав изученных микросхем.

ЛАБОРАТОРНАЯ РАБОТА №3

ИССЛЕДОВАНИЕ АРХИТЕКТУРЫ ПРОЦЕССОРА INTEL 8086

Цель работы: исследование и изучение при помощи системной программы DEBUG архитектуры шестнадцатиразрядного процессора с высокой производительностью. Исследуются программно доступные регистры, их назначение, система команд, способы адресации операндов.

1. ОБЩИЕ СВЕДЕНИЯ

В основу лабораторной работы положена персональная ЭВМ, работающая в реальном режиме. Процессор Intel 8086 можно считать ядром для всех последующих разработок фирмы Intel, поэтому все эксперименты могут быть выполнены на ЭВМ, построенных на последующих разработках кристаллов процессоров. Память ЭВМ в реальном режиме работы разбита на сегменты, которые адресуются через программно доступные сегментные регистры.

На рис 1. представлена укрупненная структурная схема процессора БИС К1810ВМ86.

В состав этой БИС входят:

Сегментные регистры

CS - сегмент кодов, SS - сегмент стека,
DS - сегмент данных, ES - дополнительный сегмент

Разрядность их равна 16 битам.

Смещение внутри каждого сегмента осуществляется при помощи:
сегмента кодов и содержимого регистра IP (программный сегмент, из которого процессор читает коды команд, дешифрирует и исполняет) (разрядность 16 бит);
в сегменте стека смещение осуществляется при помощи регистра SP (16 бит);
в сегментах данных смещение осуществляется при помощи различных регистров, которые решает использовать программист.

Регистры общего назначения

AX, BX, CX, DX - шестнадцатиразрядные, каждый из которых разбит на восьмиразрядные:

AH, AL; BH, BL; CH, CL; DH, DL.

Регистровые указатели:

SP - указатель стека, BP - указатель базы (разрядность 16 бит)

Индексные регистры:

SI - смещение в сегменте данных источника,

DI - смещение в сегменте данных приемника.

Для обработки цепочек (строк).

Регистр командного указателя: IP - содержит смещение на команду.

Флаговый регистр (регистр признаков) шестнадцатиразрядный и содержит следующие признаки:

	FH								FL							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F:	X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

OF – переполнение, ZF – нуль, DF - направление обработки цепочек, AF - полуперенос для выполнения команд десятичной коррекции для обслуживания двоично-десятичной арифметики, IF – аппаратные прерывания (разрешены или запрещены), PF – четность, TF - пошаговый режим (включен или выключен), CF – перенос из восьмого разряда в девятый, SF – знак. Остальные разряды не задействованы и считаются резервными для последующих процессоров.

Абсолютный (физический) адрес процессор формирует из содержимого сегментных регистров и смещения, находящегося в одном из регистров, хранящих смещение.

Например, адрес команды формируется из содержимого сегмента кодов (CS) и командного указателя (IP). Внутри процессора физического адреса не существует. А существует только логический адрес, с которым и работает программист. Логический адрес записывается так - CS:IP.

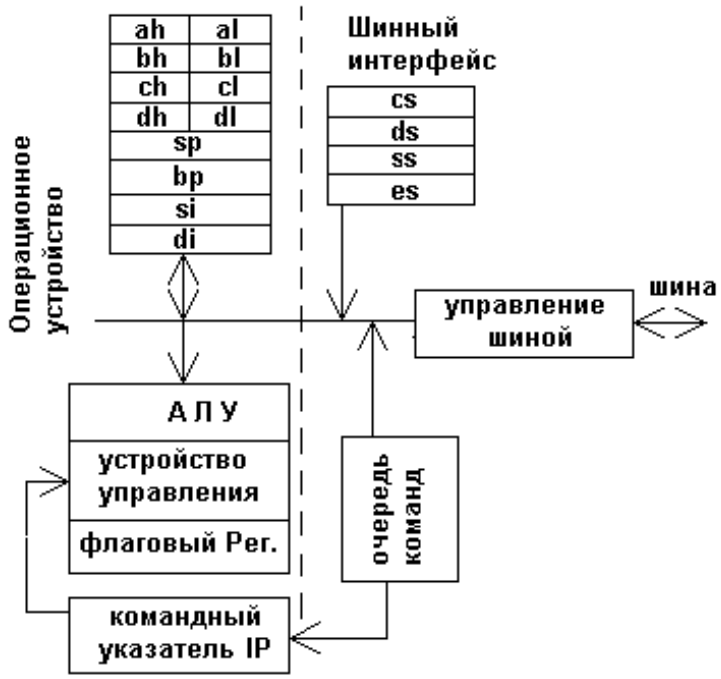

Рис.1. Укрупненная структурная схема процессора

Шины адреса и данных в этом процессоре мультиплексированы. Адресные шины имеют разрядность 20 бит. Пусть в CS записано 04AFH, а в IP - 0023H. Абсолютный (физический) адрес команды процессор сформирует на двадцатиразрядной адресной шине по следующему правилу: содержимое CS умножается на 16 (дописывается шетнадцатиричный 0 к содержимому CS), а затем осуществляет операцию сложения с содержимым регистра IP.

```
  04AF0
+
  00023
-------
  04B13H
```

1.1. Системная программа DEBUG

Системная программа DEBUG является составной частью операционной системы DOS. Программа позволяет: просматривать память, вводить программы и осуществлять трассировку их выполнения. То есть программа DEBUG позволяет решать задачу изучения архитектуры (показывается текущее состояние регистров и флагов) и проводить отладку программ.

Основные команды DEBUG: D,Q,E,R,T.

D (Dump) - просмотр памяти. Формат D N1:N2 - где N1 - содержимое сегментного регистра; N2 - содержимое регистра смещения.

В базовой системе ввода/вывода (BIOS), которая находится в ПЗУ, в ячейках 413H и 414H хранится информация об объеме основной памяти. Для просмотра ячейки 413H команда D имеет вид:

D 40:13.

Первые два байта, появившиеся на экране в результате выполнения этой команды в шестнадцатиричном формате, содержат объем памяти в килобайтах, причем представлены они в последовательности: младший, старший. То есть читать надо, переставив местами байты.

Серийный номер ЭВМ расположен по физическому адресу FE000H. Командой D FE00:0 можно просмотреть содержимое памяти с этого адреса.

Дата программирования ПЗУ BIOS находится, начиная с ячейки FFFF5H. Команда D FFFF:05 покажет содержимое ПЗУ (месяц/дата/год).

Просмотр содержимого памяти с выбором адреса через сегмент кода и сегмент данных, соответственно, осуществляется по командам

D CS:100, D DS:000.

Команда Q (Quit) - выход из DEBUG.

Команда E (Enter) - ввод. Предназначена для ввода машинных кодов команд, начиная с указанного смещения. Формат команды

E CS:100 ␣ байт ␣ байт ␣ байт ␣ …..

Завершение ввода - клавиша Enter.

Смещение 100 стандартное начальное смещение требование операционной системы, так как она использует верхние адреса сегмента как рабочие ячейки, при помощи которых производится управление исполнением приложения, расположенного в этом сегменте. Команда E может быть применена и для записи данных в сегмент данных DS:[ ].

Примеры ввода кодов в память программ и память данных:

E CS:100 8b d8 03 [enter] - осуществит последовательный ввод в ячейки ОЗУ. Начальное смещение равно IP=100. Содержимое CS операционная система DOS (затем и Windows) инициализирует всегда на сегмент, свободный для исполнения запускаемого приложения;

E DS:0000 23 01 25 00 00 00 [Enter] - последовательно будут записаны все байты, начиная с ячейки DS *16 + 0000 по тому же правилу, которое показано выше.

Команда R (Register) - просмотр содержимого регистров. При выполнении этой команды на экране высвечивается содержимое всех регистров, логический адрес и мнемокод команды, находящейся по адресу CS:IP. Команда R позволяет изменять содержимое любого регистра. Для этого требуется выполнить следующее действие:

R CS [Enter]

DEBUG покажет текущее содержание CS и напечатает двоеточие. Это есть разрешение для набора нового содержимого CS. Набрать новое содержимое на клавиатуре и нажать Enter.

Команда T - трассировка. Позволяет просмотреть динамику выполнения программы, для чего необходимо последовательно выполнять команду T. Информация на экране такая же, как и при выполнении команды R.

При помощи DEBUG и BIOS можно просмотреть размер основной памяти по команде INT 12H. Это подпрограмма BIOS для просмотра размера основной памяти, подключенной к ЭВМосква: В результате работы этой подпрограммы в регистре AX будет размещено число, соответствующее объему основной памяти ОЗУ ЭВМосква: Для выполнения этой операции необходимо набрать программу:

E CS:100 CD 12 CB [Enter]

Применив команды R и T, просмотрим трассу выполнения подпрограммы INT12h до выполнения команды RETF (возврат из INT 12).

1.2. Специальные средства DEBUG

В операционной системе DOS, начиная с версии 2.0 и старше, можно использовать программу DEBUG для ввода команд на языке АССЕМБЛЕРА. Команда A (Assemble) - переводит DEBUG в режим приема команд ассемблера.

A100 [Enter] - при этом устанавливается начальный адрес ввода xxxx:100 (где xxxx содержимое CS, которое выделит операционная система для выполнения запускаемого приложения).

Теперь можно вводить программу в мнемокодах (то есть в образах команд программиста, а не в кодах процессора), то есть после каждой команды надо нажимать [Enter]. Например:

```
-a100
xxxx:0100 mov ch,16
xxxx:0102 mov ax,016f
xxxx:0105 ret
xxxx:0106
```

Команда U (Unassemble) - показывает машинные коды для команд ассемблера (в ассемблерах это файл с расширением .lst). Пусть необходимо осуществить ассемблирование, начиная от смещения 100 до 110, команда будет иметь вид:

```
-u100,105
xxxx:0100   B516          MOV     CH,16
```

```
xxxx:0102  B86F01      MOV   AX,016F
xxxx:0105  C3          RET
```

На экране появятся таблица логических адресов, коды команд и мнемокоды. Все рассмотренные выше команды исполнять можно.

2. ЗАДАНИЕ ДЛЯ ДОМАШНЕЙ ПОДГОТОВКИ

2.1. Познакомиться с командами системной программы DEBUG.

2.2. Подготовить протоколы предстоящих экспериментов с форматами команд.

2.3. Подготовить предлагаемые программы к вводу.

2.4. Познакомиться с программно доступными регистрами и способом формирования абсолютных (физических) адресов.

3. РАБОТА В ЛАБОРАТОРИИ

3.1. Определить объем основной памяти ОЗУ в ЭВМ, на которой Вы работаете.

3.2. Определить серийный номер Вашей ЭВМ (если он указан, если нет, то убедиться в этом).

3.3. Определить дату записи в ПЗУ BIOS.

3.4. Осуществить ввод программы, представленной в кодах в таблице 3.1.(использовать начальный адрес XXXX:0100, где XXXX-сегмент кодов, предложенный операционной системой для Ваших экспериментов). В данной программе использованы **команды прямой адресации операндов.**

Таблица 3.1

Программа с командами прямой адресации операндов

Команда	Комментарий
b82301	;переслать значение 0123h в AX
052500	;прибавить значение 0025h к AX
8bd8	;переслать содержимое AX в BX
03d8	;прибавить содержимое AX к BX
8bcb	;переслать содержимое BX в CX
2bc8	;вычесть содержимое AX из CX
90	;нет операции
cb	;возврат в DOS

3.5. Осуществить просмотр содержимого памяти команд для того, чтобы удостовериться в правильности введённых кодов программы. Отметьте на Damp-ах введенную Вами информацию.

3.6. Осуществить трассировку введенной программы. Начните трассировку **с команды R**, иначе первая команда программы будет пропущена. (НЕ ЗАБУДЬТЕ РАСПЕЧАТАТЬ ПРОТОКОЛ ТРАССИРОВКИ. ОН ПОТРЕБУЕТСЯ ДЛЯ ОТЧЕТА). **Здесь же на свободном месте протокола трассировки напишите абсолютные адреса, используемые ЭВМ для выполнения Вашей программы.**

3.7. Ввести данные для следующего опыта **в сегмент данных** со смещением 0000 (команда E DS:0000… 0123 и 0025 - данные для ввода).

НАПОМИНАНИЕ: ДВУХБАЙТНЫЕ ДАННЫЕ (СЛОВА) ЗАПИСЫВАЮТСЯ ПО ПРАВИЛУ: МЛАДШИЙ – В МЛАДШИЙ АДРЕС; СТАРШИЙ – В СТАРШИЙ АДРЕС!

3.8. Осуществить **просмотр памяти данных**, с целью удостовериться в правильности введённой информации. (D DS:0000 [Enter]). В Damp-е выделить введённую информацию.

3.9. Осуществить ввод программы из таблицы 3.2. **в сегмент кодов**, начиная с адреса CS:100, которая будет выполнять те же действия, что и первая, но **данные будут извлекаться из памяти данных**. Используются команды с другим способом адресации, косвенная адресация.

Таблица 3.2

Программа с командами косвенной адресации операндов

Команда	Назначение
A10000	;переслать слово (два байта), начинающееся в DS с ;смещением 0000, в регистр AX.
03060200	;сложить слово (два байта), находящееся в DS ; с смещением 0002, с содержимым регистра AX
A30400	;переслать содержимое регистра AX , в память ;с логическим адресом DS : 0004
CB	;вернуться в DOS

3.10. Осуществить просмотр памяти команд с целью удостовериться в правильности введённой информации. (D CS:100 [Enter]). В Damp-е выделить введённую информацию.

3.11. Осуществить просмотр содержимого регистров и сделать трассировку введенной программы. Получить протоколы трассировки. Сравнить полученные результаты с предыдущим примером. Здесь же, на свободном

месте протокола трассировки напишите абсолютные адреса, используемые ЭВМ для выполнения этого опыта. Обратите внимание на то, что команды с косвенным способом адресации операндов двухцикловые, значит, физических адресов будет по два на команду.

3.12. Осуществить ввод программы запуска подпрограммы из BIOS, пересылающей основной объем ОЗУ в AX. Ваши действия:

E CS:100 cd 12 cb [enter]

Осуществить трассировку выполнения подпрограммы, получить протокол трассировки. Полученный результат сравнить с ранее полученным результатом просмотра памяти ПЗУ. Здесь же, на свободном месте протокола трассировки напишите абсолютные адреса, используемые ЭВМ для выполнения Вашей программы. Обратите особое внимание на логические адреса, в которых выполняется подпрограмма INT 12.

3.13. Ввести ассемблерную программу (команда A100 [Enter])

```
MOV  AL,  25
MOV  BL,  32
ADD  AL,  BL
RET
```

Получить протокол действий DEBUG после ввода программы.

3.14. Выполнить команду U для введенной программы. Адреса начальный и конечный 100 и 106. Протестировать работу программы, обратив внимание на байтный характер операндов. В протоколах трассировки выделить изменение программно доступных регистров после выполнения команд.

4. СОДЕРЖАНИЕ ОТЧЕТА

4.1. Общие положения (по Вашему усмотрению).

4.2. Структурная схема процессора с комментариями по программно доступным регистрам.

4.3. Описание основных команд DEBUG.

4.4. Обработанные протоколы проведенных экспериментов. Протокол считается обработанным, если Вы можете показать, как найти адрес выполненной команды и по какому адресу находятся данные.

4.5. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 4

АССЕМБЛИРОВАНИЕ И ВЫПОЛНЕНИЕ ПРОГРАММ

Цель работы: показать процессы ассемблирования, компоновки и выполнения программ различных типов (.EXE и .COM) на высокопроизводительных ЭВМосква: Используется реальный режим работы ЭВМ, система команд шестнадцатиразрядная.

1. ОБЩИЕ СВЕДЕНИЯ

При разработке программ на ассемблере требуются: любой текстовый редактор для набора текста исходной программы, Ассемблер (в данном случае MASM) и программа-компоновщик LINK.

Если программы на языке БЕЙСИК-интерпретатор могут быть выполнены сразу после ввода исходного текста, то программы на АССЕМБЛЕРЕ и других компилирующих языках нуждаются перед выполнением в трансляции и компоновке.

Эти функции и выполняют программы MASM и LINK.

Шаг ассемблирования включает в себя трансляцию исходного текста в машинный объектный код и генерацию модуля с расширением .OBJ. Этот формат еще не готов к исполнению, необходим шаг компоновки - то есть преобразование модуля .OBJ в модуль .EXE (исполняемый модуль).

Программа LINK осуществляет следующее:

1. Завершает формирование в модуле .OBJ адресов, которые остались неопределенными после ассемблирования.

2. Компонует более одного отдельно ассемблированного модуля в одну загрузочную программу. Это две или более ассемблерных программ или ассемблерная программа и программы, написанные на языках высокого уровня, таких как ПАСКАЛЬ или БЕЙСИК.

3. Формирует исполняемый модуль .EXE командами загрузки для выполнения.

В случае необходимости редактирования программы .EXE требуется внести изменения в исходную программу и проделать весь путь до получения нового исполняемого модуля.

1.1. Ассемблирование программы

Для создания исходного текстового модуля удобнее использовать простые текстовые редакторы (Блокнот или Edit из Far Manager). Исходный мо-

дуль должен иметь расширение .ASM. Это избавит программиста от работы с расширениями. Для ассемблирования исходного модуля необходимо запустить программу Ассемблер MASM.EXE.

Работа с программой организована в режиме диалога:

```
source filename      [.ASM]          ;введите имя исходной программы;
object filename  [filename.OBJ]      ;имя  выходного  модуля .OBJ
source listing     [nul.LST]         ;если требуется листинг ассемблирования, то
                                     введите имя ;проекта программы Enter;
cross-reference    [nul.CRF]         ;если нужен листинг перекрестных ссылок,
                                     укажите путь ;и нажмите  Enter;
```

если хотите оставить значения по умолчанию, то в трех последних предложениях нажмите клавишу Enter.

Ассемблер преобразует мнемокоды в машинные коды и выдает на экран сообщение об ошибках. Ассемблер MASM является многопроходной программой, что означает, если в исходном тексте имеется хотя бы одна ошибка, выходной файл .OBJ не создаётся.

ПРИМЕЧАНИЕ: формировать листинг ассемблирования есть смысл, так как ассемблер расставляет сообщения об ошибках сразу после обнаружения, а значит быстрее можно определить тип ошибки.

1.2. Компоновка программы

Если в результате ассемблирования не обнаружено ошибок, то необходимо переходить к компоновке исполняемой программы при помощи LINK.

Для выполнения этого шага необходимо запустить программу LINK и ответить на следующие предложения:

```
object Modules      [.obj]   ;имя входного модуля .OBJ Enter;
file                         ;имя выходного модуля .EXE Enter;
[filename.EXE]
File              [nul.MAP]  ;MAP файл содержит таблицу имен, размеров сег-
                             ментов и ;ошибки, обнаруженные LINK
libraries           [.LIB]   ;для ответа на этот вопрос нажмите Enter
                             ;(описание библиотечных средств).
```

После получения исполняемого модуля .EXE его можно протестировать запуском его для исполнения либо при помощи программы DEBUG .

1.3. Правила составления исходной программы для получения исполняемой типа .EXE и .COM

При составлении исходной программы необходимо помнить, что правила её составления для получения исполняемой типа .EXE и типа. COM несколько различны.

Эти правила достаточно формальны и связаны с операционной системой, установленной на данном конкретном вычислителе.

Для .EXE программ эти правила следующие:

1. Вначале задается директива Ассемблеру о формате страницы листинга и имени исходной программы

```
PAGE 60,132
TITLE <имя программы>.(EXE)<Составление лабораторного варианта>
```

2. Резервирование стековой памяти. Резервируется не менее 32 слов. Делается это при помощи описания сегмента стека.

```
STACKSG SEGMENT PARA STACK 'Stack'
        DW    32  DUP(?)
STACKSG ENDS
```

3. Определение данных (память данных) выполняется при помощи описания сегмента данных (если будет использован дополнительный сегмент данных, то описание данных производится по тем же правилам)

```
DATASG   SEGMENT  PARA  'Data'
          A   DW  250  ;переменной  A  присваивать слово 250
          B   DW 125 ;переменной  B  присваивать  слово  125
          C   DW ? ;переменной C резервировать память под слово
DATASG   ENDS
```

Строка A определяет слово, содержащее десятичное значение 250, которое Ассемблер транслирует в шестнадцатиричное 00FA. Строка B определяет слово с десятичным значением 125. Строка C определяет слово с неизвестным значением, обозначенным знаком (?).

4. Сегмент кода имеет имя CODESG. Этот сегмент предназначен для размещения мнемокодов приложения. Открытие сегмента кодов делается следующим образом:

```
CODESG   SEGMENT  PARA  'Code'
   BEGIN PROC FAR          ;начало процедуры begin, пересылки длинные
ASSUME CS:CODESG, DS:DATASG, SS:STACKSG, ES:NOTHING
```

Особенности: директива ASSUME (установить для Ассемблера связь между сегментами и сегментными регистрами CS, DS, ES, SS) указывает на определение DATASG через регистр DS, STACKSG через регистр SS, дополнительный сегментный регистр не задействован ES (иначе этот сегмент необходимо определить).
Процедура FAR переводит Ассемблер в режим формирования адресов команд управления ДЛИННЫЙ (межсегментный CS:IP).
Далее идет обязательная процедура по организации стека

```
PUSH   DS        ;записать в стек текущее значение регистра DS
SUB    AX,AX     ;обнулить содержимое регистра AX
PUSH   AX        ; записать в стек текущее значение регистра AX
```

Затем две команды, обеспечивающие адресацию сегмента данных

```
MOV  AX, DATASG   ;записать в регистр AX адрес сегмента, который определила операционная система
MOV  DS, AX       ;переслать содержимое AX в регистр сегмент данных DS
```

Этой операцией наше разрабатываемое приложение будет исполняться в сегменте, выделенном операционной системой.

Далее идет текст исходной программы в мнемокодах.

Завершается кодовый сегмент следующим образом:

```
BEGIN    ENDP      ;конец процедуры BEGIN
CODESG   ENDS      ;конец сегмента кодового
END    BEGIN
```

При работе LINK автоматически генерируется особый формат для .EXE файлов, в котором присутствует специальный начальный блок (заголовок) размером не менее 512 байт.

При разработке исходного текста приложения типа .COM необходимо учитывать то обстоятельство, что такие приложения выполняются в пределах адресов одного сегмента (64К). Стек в .COM программах генерируется авто-

матически под управлением операционной системы. Размер .COM файла всегда меньше .EXE. Пример исходного текста для программы типа .COM приведен в таблице 1.1.

Таблица 1.1.

Пример исходной программы для исполняемой типа .COM

page 60,132	
title имя программы (.COM)	
codesg segment para 'code'	
assume cs:codesg,ds:codesg,ss:codesg,es:codesg	
;----------	
org 100h	;обязательный атрибут исходной программы
;----------	
begin: jmp main	
A dw 250	
B dw 125	
C dw ?	
main proc near	; near - короткие пересылки
mov ax,A	
add ax,B	
mov C,ax	
Ret	
main endp	
codesg ends	
end begin	

Как видно из таблицы 1.1. приложение состоит из одного сегмента кодов, в теле которого имеется блок определения переменных (A, B, C). Обязательный атрибут исходной программы типа .COM – org 100h. Это означает, что приложение должно начинаться в выделенном операционной системой сегменте со смещения 100h.

Для подготовки исполняемого модуля типа .COM необходимо выполнить те же процедуры, что и для модуля типа .EXE. То есть воспользоваться программами MASM и LINK. После редактора связей LINK получится программный модуль с расширением .EXE. Но исполняться правильно он не будет, так как исходный текст подготовлен под тип .COM. Требуется ещё одно действие по преобразованию модуля .EXE в .COM при помощи специальной программы EXE2BIN.EXE. Программа EXE2BIN.EXE в операционной системе DOS преобразует .EXE файлы (подготовленного под тип .COM) в формат BIN. Для получения исполняемого файла .COM .BIN файл средствами DOS

необходимо переименовать в COМосква: Формат команды преобразования в бинарный:

EXE2BIN.EXE < имя преобразуемого файла >

При работе EXE2BIN, в случае обнаружения ошибки, выдается сообщение о невозможности преобразования .EXE модуля в формат .BIN без указания конкретной причины. (EXE2BIN - преобразователь из .EXE в BIN (двоичный) формат (EXE-to-BIN)).

Для .COM файлов DOS автоматически определяет стек и устанавливает одинаковый сегментный адрес во всех четырех сегментных регистрах.

Если опущен ORG 100H, то на данные в префиксе программного сегмента будут установлены неправильные ссылки с непредсказуемым результатом при выполнении.

Попытка выполнить .EXE модуль, написанный для .COM формата, успеха не имеет. Современные операционные системы (Windows) поддерживают реальный режим. Поэтому все требования DOS, описанные в правилах подготовки исполняемых модулей типов .EXE и .COM будут выполнены.

1.4. Устройства ввода-вывода для проведения эксперимента

Для проведения эксперимента по проверке правильности составленной программы управления предлагается использовать клавиатуру и дисплей. В базовой системе ввода-вывода (BIOS), резидентно расположенной в ПЗУ каждой ПЭВМ, есть подпрограммы обращения к этим устройствам и всем, входящим в состав машины.

Все необходимые экранные и клавиатурные операции можно выполнить с помощью подпрограммы INT 10H, которая передает управление в BIOS.

Для выполнения более сложных операций существует прерывание более высокого уровня INT 21H, которое сначала передает управление в DOS. (Например, при вводе с клавиатуры потребуется подсчет введенных символов, проверка на максимальное число символов и проверка на символ "Enter").

1.4.1. Установка курсора

Обычный монитор имеет 25 строк и 80 столбцов. В каждую позицию экрана курсор может быть установлен программно. В таблице 1.2. приведены примеры положения курсора на экране.

Подпрограмма INT 10H включает в себя установку курсора в любую позицию и очистку экрана. Пример установки курсора на 5-ю строку и 12-й столбец имеет вид:

```
mov ah,02h          ; запрос на установку курсора
mov bh,00h          ; номер страницы экрана
mov dh,05h          ; строка 05
mov dl,12h          ; столбец 12
int 10h             ; передача управления в BIOS
```

Таблица 1.2

Положение курсора на экране

положение курсора	десятичный формат		шестнадцатиричный формат	
	строка	столбец	строка	столбец
верхний левый угол	00	00	00	00
верхний правый угол	00	79	00	4F
центр экрана	12	39/40	0C	27/28
нижний левый угол	24	00	18	00
нижний правый угол	25	79	18	4F

Для установки строки и столбца можно использовать одну команду

```
MOV dx,050ch        ;строка 5 столбец 12
```

1.4.2. Очистка экрана

Очистка экрана может начинаться с любой позиции и заканчиваться в любой другой с большим номером. Начальное значение строки и столбца заносится в регистр CX, конечное - в DX, значение 07 - в регистр BH и 0600H в AX. С этими параметрами можно запускать подпрограмму INT 10h.
Пример:

```
mov ax,0600h        ; AH 06 (прокрутка) AL 00 (весь экран)
mov bh,07h          ; нормальный атрибут (черно/белый)
mov cx,0000h        ; верхняя левая позиция
mov dx,184fh        ; нижняя правая позиция
int 10h             ;передача управления в BIOS
```

1.4.3. Экранные и клавиатурные операции в базовой версии DOS

Вывод на экран требует определения текстового сообщения в области памяти данных, установки в регистр AH значения 09h (вызов функции DOS) и указания команды DOS INT 21H. Конец сообщения определяется ограничителем ($).

Пример:

```
Imia    db  'имя покупателя?','$'
mov ah, 09              ; запрос вывода на экран
lea dx,Imia             ;загрузка в dx адреса сообщения
int 21h                 ;вызов прерывания DOS
```

1.4.4. Вывод на экран набора символов ASCII-кода

Большинство из 256 ASCII кодов имеют символьное представление и могут быть выведены на экран. Шестнадцатиричные коды 00 и ff не имеют символов и выводятся на экран в виде пробелов, хотя символ пробела имеет ASCII-код 20H.

В таблице 1.3. приведена программа (.COM типа), которая выводит на экран полный набор символов ASCII-кода. Программа использует три процедуры (подпрограммы): b10clr, c10set, d10disp. Первая очищает экран, вторая устанавливает курсор в положение 0,0, а третья выводит содержимое поля CTR, которое вначале инициализировано значением 00 и затем увеличивается на 1 при каждом выводе на экран, пока не достигнет шестнадцатиричного значения FF.

Таблица 1.3

Пример программы (COM) вывода на экран ASCII-кода

```
;-------------------------------------------------------------------
page   60,132                   ; формат страницы листинга
title allasc (COM)              Вывод на экран ASCII-символов
;-------------------------------------------------------------------
codesg segment  para  'code'    ; начало программы
assume cs:codesg,ds:codesg,ss:codesg,es:codesg
;-------------------------------------------------------------------
org  100h               ; начать программу со смещения 100h
;-----------------------------------------------
begin:  jmp  short  main
```

```
ctr     db   00,'$                    ; байт ctr равен 00
;---------------------------------------
; основная процедура
main  proc  near
call  b10clr                          ; процедура -очистить экран
call  c10set                          ; процедура - установить курсор
call  d10disp                         ; процедура - вывести символ на экран
Ret                                   ; возврат в операционную систему
main  endp                            ; конец основной процедуры
;----------------------------------
; процедура-очистка экрана
;----------------------------------
b10clr   proc                         ; начало процедуры b10clr
mov  ax,0600h
mov  bh,07h
mov  cx,0000h                         ; левая верхняя позиция экрана
mov  dx,184fh                         ; правая нижняя позиция экрана
int  10h
ret
b10clr   endp                         ; конец процедуры b10clr
; установка курсора в 00,00
;----------------------------------
c10set     proc                       ; начало процедуры c10set
mov  ah,02h
mov  bh,00h
mov  dx,0000h
int  10h
          ret
c10set     endp                       ; конец процедуры c10set
;----------------------------------
;вывод на экран ASCII символов
;----------------------------------
d10disp    proc                       ; начало процедуры d10disp
mov  cx,256                           ; 256 итераций
d20:                                  ; метка
 lea  dx,ctr                          ; пересылка в dx адреса байта ctr
mov  ah,09h                           ; функция вывода символа
int  21h
inc  ctr                              ; увеличить байт ctr на единицу
```

loop d20	; уменьшить CX, если не 0 то на метку d20
ret	
d10disp endp	; конец процедуры d10disp
codesg ends	
end begin	; конец программы

1.4.5. Ввод данных с клавиатуры

Область ввода требует наличия списка параметров, содержащего спецификацию полей, которые необходимы при выполнении команды INT. Во-первых, должна быть определена максимальная длина вводимой информации, для предупреждения пользователя звуковым сигналом. Символы, превышающие максимальную длину, игнорируются. Во-вторых, в списке параметров должно быть определено поле, куда команда возвращает действительную длину введенного текста в байтах.
Пример:

```
namepar  label  byte          ; список параметров
maxlen   db   20              ; максимальная длина
actlen   db   ?               ; реальная длина
namefld  db  20 dup(' ')      ; введенные символы
```

Для запроса на ввод необходимо поместить в регистр AH номер функции - 10 (0AH), загрузить адрес списка параметров (NAMEPAR в примере) в регистр DX и выполнить
INT 21H:

```
mov  ah,0ah                   ; запрос функции ввода
lea  dx,namepar               ; загрузка адреса списка параметров
int  21h                      ; ;вызвать функцию ввода DOS
```

Если пользователь ввел имя BROWN и нажал Enter, то список параметров будет содержать следующую информацию:

```
десятичные и символьные      20  5  B  R  O  W  N  #
шестнадцатиричные          14 05 42 52 4f 57 4e 0d 20 20
```

Первый байт - максимальная длина вводимой информации (maxlen db20). Во второй байт списка параметров подпрограмма INT 21H заносит длину введенного имени - 05.

Пример программы ввода имен и вывода их в центре экрана приведен в таблице 1.4.

Таблица 1.4

Программа ввода имен и вывода их в центре экрана

```
;-----------------------------------------------------------------
page 60,132
title ****** (exe) Ввод и вывод
;-----------------------------------------------------------------
stacksg segment para 'stack'        ; формирование сегмента стека
dw  32 dup(?)                       ; минимальный объем стека для .exe
stacksg ends                        ; конец формирования сегмента стека
;-----------------------------------------------------------------
datasg segment para 'data'          ; формирование сегмента данных
namepar  label  byte                ; имя списка параметров
maxnlen  db 20                      ; максимальная длина имени
namelen  db ?                       ; число введенных символов
namefld  db 20 dup(' '),'$'         ; имя и ограничитель
primpt    db  'Name?','$'           ; приглашение
datasg   ends                       ; конец формирования сегмента данных
;-----------------------------------------------------------------
codesg segment para 'code'          ; начало содового сегмента
begin proc  far                     ; начало процедуры begin пересылки длинные
assume cs:codesg,ds:datasg,ss:stacksg,es:datasg
push   ds
sub    ax,ax
push   ax
mov    ax,datasg                    ; установка сегмента
mov    ds,ax                        ; данных
mov    es,ax                        ; установка дополнительного сегмента данных
a20:                                ; метка
call   q10clr                       ; процедура очистки экрана
mov    dx,0000                      ;установка курсора в 0
call   q20curs                      ; процедура установки курсора
call   b10prmp                      ; процедура выдать приглашения
call   d10inpt                      ; процедура ввода имени
call   q10clr                       ; процедура очистки экрана
```

cmp namelen,00	; имя введено?
je a30	; нет - выйти
call e10code	; выдать сигнал и ограничитель '$'
call f10cent	; центрирование и вывод
jmp a20	; возврат на метку a20
a30:	; метка
ret	; вернуться в DOS
;------------------------------	
b10prmp proc near	; начало процедуры вывода на экран
mov ah,09	; функция вывода на экран
lea dx,primpt	; установка адреса вывода приглашения
int 21h	; прерывание, выполняющее вывод
ret	
b10prmp endp	; конец процедуры вывода приглашения
;------------------------------	
d10inpt proc near	
mov ah,0ah	; функция ввода
lea dx,namepar	; установка адреса вводимой информации
int 21h	; прерывание, выполняющее ввод
ret	
d10inpt endp	; конец процедуры ввода
;------------------------------	
e10code proc near	; процедуры установки сигнала и ограничителя
mov bh,00	;замена символа 0d на 07
mov bl,namelen	
mov namefld[bx],07	
mov namefld[bx+1],'$'	;установить ограничитель
ret	
e10code endp	; конец процедуры
;------------------------------	
begin endp	
codesg ends	

2.ЗАДАНИЕ ДЛЯ ДОМАШНЕЙ ПОДГОТОВКИ

2.1. Познакомиться с разделом "общие сведения" и правилами подготовки исходных файлов для исполнения программ .EXE и .COM типов.
2.2. Изучить последовательность действий для компоновки программ.

2.3. Подготовить тексты исходных файлов для исполняемых программ типов .EXE и .COM по пунктам 1.4.1., 1.4.2. и 1.4.3. (НАПОМИНАНИЕ: наиболее подходящие для подготовки текстов исходных модулей редакторы текстов БЛОКНОТ или FarEdit).
2.4. Подготовить текст исходного файла для программы из таблицы 1.4. пункт 1.4.4. для исполняемого .EXE типа.
2.5. Изучить программу из пункта 1.4.5. (таблица 1.4) и написать начало этой же программы для исполняемой типа .COMосква:

3. РАБОТА В ЛАБОРАТОРИИ

3.1. Набрать текст исходной программы, составленной по пункту 2.3. (оба варианта, для .EXE и .COM).
3.2. Произвести ассемблирование этих исходных программ. В случае обнаружения Ассемблером ошибок, внести исправление в редакторе и снова повторить ассемблирование. (Для отчета файлы .LST распечатать).
3.3. Скомпоновать программы для исполнения. .EXE модуль, предназначенный для .COM типа обработать программой EXE2BIN.
3.4. Полученные исполняемые программы запустить в работу методами Windows или Far Menedger. Убедиться в работоспособности Ваших программ.
3.5. Подготовить при помощи редактора исходные файлы для программ из таблицы 1.2. (.EXE и .COM типов).
3.6. Проделать с ними действия, указанные в пунктах 3.2., 3.3. и 3.4.
3.7. При помощи редактора подготовить исходную программу из таблицы 1.3. (.EXE или .COM типа по Вашему выбору), ассемблировать ее, скомпоновать и протестировать методом пуска.
НАПОМИНАНИЕ: НЕ ЗАБУДЬТЕ СДЕЛАТЬ РАСПЕЧАТКИ ИСХОДНЫХ И ВЫХОДНЫХ ФАЙЛОВ .ASM И .LST ОБЯЗАТЕЛЬНО ВСЕ, А ПО ОПЫТУ 3.7. .EXE ИЛИ .COM ОДИН, ЛЮБОЙ ПО ВАШЕМУ ВЫБОРУ.

4. СОДЕРЖАНИЕ ОТЧЕТА

4.1. Фрагменты "ОБЩЕЙ ЧАСТИ" на Ваше усмотрение.
4.2. Листинги исходных программ по пункту 3.2.
4.3. Листинги программ .LST и Print Screen экрана при выполнении Ваших программ. Прокомментировать распечатки.
4.4. Листинги исходных программ по пункту 3.6. и Print Screen экрана при выполнении Ваших программ. Прокомментировать распечатки.
4.5. Листинги программ из таблицы 3 пункт 3.7. и Print Screen экрана при выполнении Вашей программы. Прокомментировать распечатки.

4.6. Каким образом выполняются процедуры управления периферийным устройством дисплей в ПЭВМосква:
4.7. Каким образом выполняются процедуры управления периферийным устройством клавиатура в ПЭВМосква:

ЛАБОРАТОРНАЯ РАБОТА № 5

ИССЛЕДОВАНИЕ ПРОГРАММИРУЕМОГО ПАРАЛЛЕЛЬНОГО ВВОДА/ВЫВОДА. В КАЧЕСТВЕ ПЕРИФЕРИЙНОГО УСТРОЙСТВА – ИСТОЧНИК ЗВУКА

Цель работы: знакомство, приобретение навыков программирования и использования БИС программируемого параллельного ввода/вывода 1810ВВ55 (Intel 8255), как канала управления периферийным устройством - источник звука.

1. ОБЩАЯ ЧАСТЬ

Для знакомства и исследования микросхемы 1810ВВ55 предлагается использовать встроенный в системный блок ПЭВМ источник звука динамик, предназначенный для сопровождения действий оператора звуковыми сигналами. Динамик программно управляется при помощи двух программируемых БИС - ППИ 1810ВВ55 и таймера 1810ВИ54.

На рис.1.1 показана структурная схема подключения динамика к выходам ППИ и ПТ.

Канал 2 ПТ связан с динамиком компьютера. Канал 2 может быть программно отсоединен от громкоговорителя. Динамик не будет генерировать звук до определенных установок микросхемы интерфейса с периферией 8255 (отечественные аналоги 580ВВ55 и 1810ВВ55).

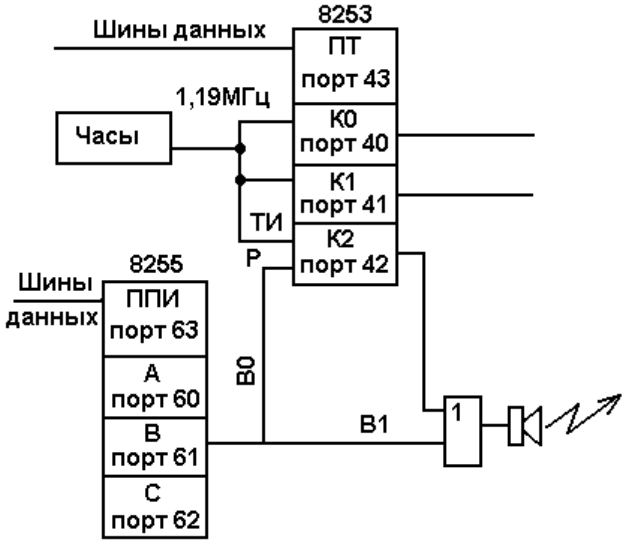

Рис.1.1. Структурная схема подключения источника звука в ПЭВМ

В этой лабораторной работе будем применять для создания звуковых эффектов только ППИ. Структурная схема ППИ представлена на рис.1.2. ШД может быть соединена с одним из трех каналов A, B, C. Все каналы восьмиразрядные, направление приема или передачи по каналу программируется со стороны программ, то есть управляется программистом. Поэтому ППИ называют «программируемый» интерфейс. Программирование осуществляют перед началом обмена, эту процедуру называют инициализация ППИ. Два канала ППИ A и B могут принимать или передавать байт данных, а канал C программно может быть запрограммирован еще и на прием или передачу информации по своим тетрадам, то есть разряды C0 - C3 – ввод, C; - C7 – вывод или наоборот. Каждый бит канала C может быть использован как стробирующий сигнал, который формируется программно, то есть возможно побитное программное управление.

ППИ может быть запрограммирован на три режима работы: Режим 0 (P0) – режим простого обмена, когда все три канала могут быть использованы для приема или передачи информации без стробирующих сигналов между ЦПЭ и ВНУ; Режим 1 (P1) – режим обмена информацией по каналам A и B под управлением стробирующих сигналов, формируемых по каналу C (сигналы «квитанции» – такой метод обмена еще называют квитированием информации или «рукопожатия»); Режим 2 (P2) – режим обмена информацией по каналу A в двух направлениях под управлением сигналов, формируемых по каналу C (двунаправленная магистраль A). Если посмотреть стандартные схемы современных параллельных интерфейсов и протоколы обмена по ним, то можно увидеть тесную связь с конструкцией ППИ.

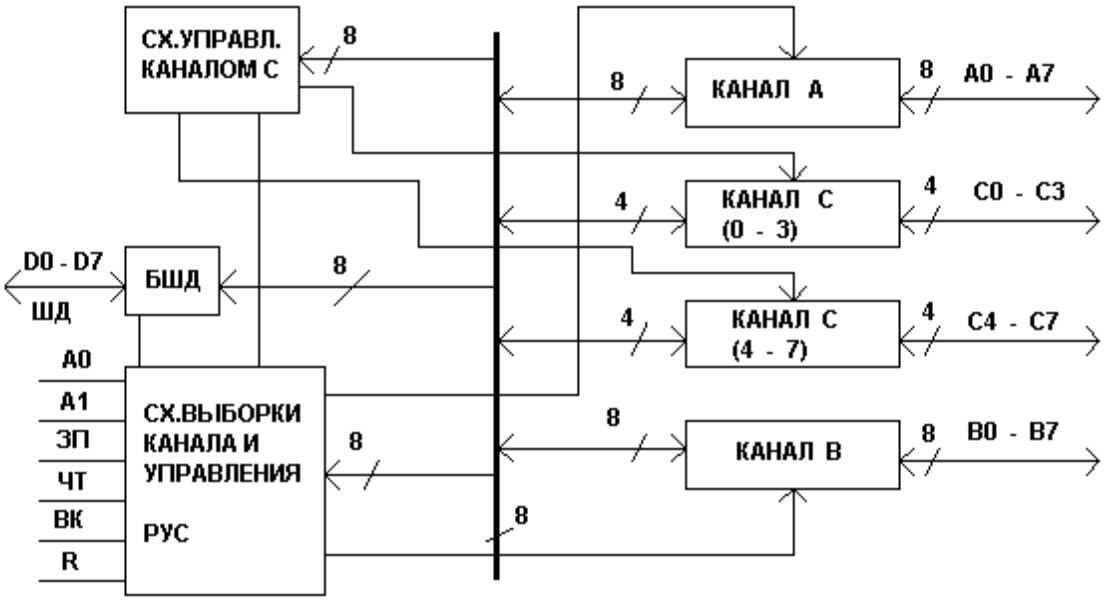

Рис.1.2. Структурная схема ППИ

На структурной схеме обозначено: СХ. УПРАВЛ. КАНАЛОМ С – схема управления каналом С; КАНАЛ А, КАНАЛ В – восьмиразрядные неделимые каналы обмена информацией в двух направлениях; КАНАЛ С – может работать как восьмиразрядный в двух направлениях, так и с разделением направления приема или передачи информации по тетрадам с возможностью побитного программного управления; БШД – буфер шины данных предназначен для буферизации принимаемого или передаваемого байта и связан с шиной данных системы; СХ.ВЫБОРКИ КАНАЛА И УПРАВЛЕНИЯ – схема выборки канала и управления предназначена для управления обменом по сигналам, поступающим из системы, по которым производится выбор нужного канала, включается установленное ранее направление передаваемого байта и формируется внутренний сигнал обмена; РУС – регистр управляющего слова, предназначен для хранения управляющего слова ППИ, в котором указано: режим работы ППИ, какой канал в каком направлении должен передавать байт информации. D0-D7 - двунаправленная магистраль данных для передачи данных , управляющих слов и информации состояния, A0-A7 - двунаправленная магистраль данных канала А, B0-B7 - двунаправленная магистраль данных канала В, C0-C7- двунаправленная магистраль данных канала С, ВК - выбор кристалла; A0, A1 - входы , необходимые для выбора одного из каналов А , В , С или регистра управляющего слова (адрес канала или РУС); ЧТ (чтение) – вход, на который подается сигнал управления чтением информации из канала, адрес которого установлен на входах A0, A1 (от ВНУ к ЦПЭ); ЗП (запись) - вход управления записью информации из ЦПЭ к ВНУ; R - вход, используемый для начальной установки схемы (сброс). При подаче сигнала на этот вход содержание всех внутренних регистров устанавливается на нуль. Как уже отмечалось, перед началом обмена информацией необходимо запрограммировать ППИ на требуемый режим и направление приема/передачи информации по каналам. Инициализация проводится при помощи двух команд:
загрузить в аккумулятор байт управляющего слова ППИ; переслать УС в регистр управляющего слова ППИ.
Адресация в ППИ осуществляется по входам A0, A1:

A1	A0	Канал
0	0	Канал А
0	1	Канал В
1	0	Канал С
1	1	РУС

Для нахождения байта управляющего слова (УС) используют информацию о формате управляющего слова ППИ, в котором содержится назначение каждого бита УС. Формат УС ППИ представлен на рис.1.3.

Сигнал разрешения для работы канала 2 ПТ формируется установкой младшего бита (B0) порта B с адресом 61H, который является регистром порта B микросхемы 8255. **Сброс этого бита переводит сигнал разрешения в состояние нельзя каналу 2 работать.** Бит B1 порта 61H связан с динамиком и может использоваться для генерации звука.

Генерирование звука с помощью ППИ состоит во включении и выключении с желаемой частотой бита порта B, который связан с динамиком (бит B1). Порт B имеет адрес 61H (хотя AT не имеет микросхемы интерфейса с периферией 8255 как таковой, он использует для этой цели тот же адрес порта и тот же бит).

Если программа переключает значение бита с максимально возможной частотой, то частота слишком высокая и человеческое ухо не способно услышать воспроизводимый звук. Поэтому необходимо формировать частоты, воспринимаемые ухом и воспроизводимые динамиком. В таблице 1.1 приведены частоты нот первой октавы.

Частоты на октаву выше можно получить, удваивая эти значения, на две октавы выше - еще раз удваивая частоты. И наоборот, частоты на октаву ниже равны, приблизительно, половине этих значений (хорошо настроенное пианино точно не следует арифметическим интервалам). ***Помните, что бит B0 порта B управляет сигналом разрешения канала 2 ПТ. Поэтому этот бит должен быть сброшен в нуль на время выполнения эксперимента с ППИ.*** На рис. 1.3. показано, как можно установить частоту звука при помощи ППИ.

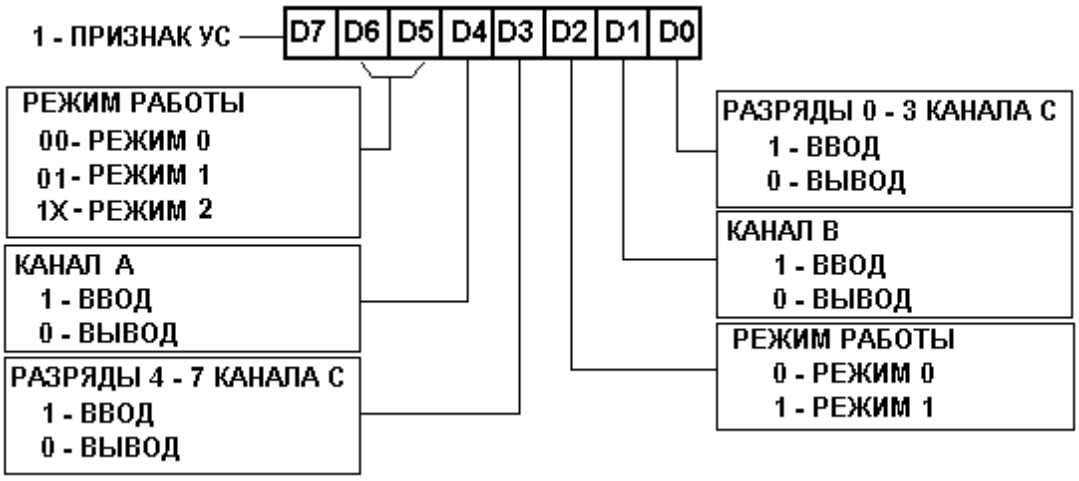

Рис.1.3. Формат управляющего слова ППИ

Таблица 1.1

Частоты первой октавы, начиная с ноты С(до)

Нота	Частота в Гц
С(до)	523,3
D(ре)	587.3
Е(ми)	659,3
F(фа)	698,5
G(соль)	784,0
А(ля)	880,0
В(си)	987,7

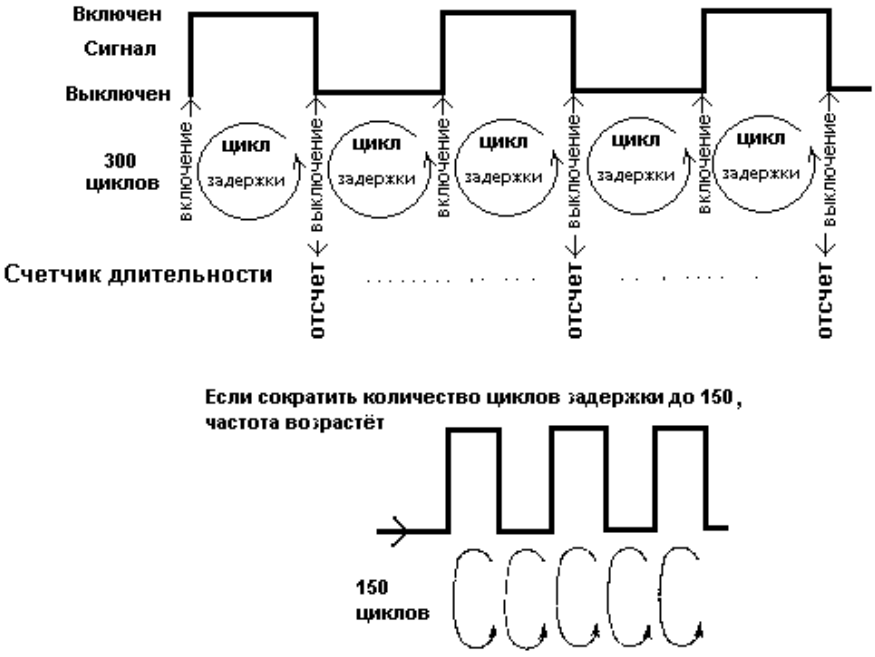

Рис.1.4. Алгоритм программного формирования звука ИМС 8255

В примере программы (табл. 1.2) введены две переменные. Одна, обозначенная "FREQ", используется в качестве счетчика в цикле формирования паузы. Переменная же "NUM_CES" устанавливает продолжительность тона, то есть сколько раз должен быть повторен процесс включения и выключения бита В1 порта В. Чем больше это число, тем дольше звучит данный звук.

Отметим, что для этой процедуры аппаратные прерывания должны быть запрещены. Причина этого в том, что прерывание таймера происходит с такой частотой и регулярностью (18,2 раза в секунду), что оно будет существенно влиять на частоту формируемого звука. Имейте в виду, что пока пре-

рывания запрещены, счетчик времени суток BIOS не будет работать. ОДНАКО, В СОВРЕМЕННЫХ ОС, РАБОТАЮЩИХ С ГЛОБАЛЬНЫМИ СЕТЯМИ, ПРОГРАММНЫЕ ВОЗДЕЙСТВИЯ НА ПРОЦЕССОР (запрет или разрешение прерываний)НЕ ДОПУСКАЮТСЯ. Потому «чистый» звук мы не услышим, но наши программные действия абсолютно справедливы. И если бы не было сложной ОС, мы имели бы «чистый» звук. Эффект «скрежетания» будет, конечно, из-за того, что аппаратные прерывания будут обслуживаться, но заметно влиять мы сможем только на время «скрежетания».

Таблица 2

Исходный текст программы генерирования звука на ИМС 8255

NUM_CES	**EQU 1000**	**;длительность звучания**
FREQ	**EQU 300**	**; длительность полупериода**
	CLI	**;запрет прерываний**
	MOV DX,NUM_CES	**;длительность тона в DX**
	IN AL,PORT B	**;получаем значение из порта B**
	AND AL,11111110B	**;отключаем динамик от таймера**
NEXT_CLE:	**OR AL,00000010B**	**; включаем динамик**
	OUT PORT B,AL	**;посылаем команду и порт B**
	MOV CX,FREQ	**;задержка на полпериода в CX**
FIR_HALF:	**LOOP FIR_HALF**	**;делаем задержку**
	AND AL,11111101B	**;выключаем динамик**
	OUT PORT B,AL	**;посылаем команду в порт B**
	MOV CX,FREQ	**;задержка на полпериода в CX**
SEC_HALF:	**LOOP SEC_HALF**	**;делаем задержку**
	DEC DX	**;вычитаем единицу из счетчика**
	JNZ NEXT_CLE	**;если 0, то надо заканчивать**
	STI	**;разрешить прерывания**

В таблице 1.3 приведен пример, практически повторяющий гудок, который выдается при старте системы. Плавные переходы тонов производятся за счет непрерывного изменения частоты. Такой звуковой эффект можно сделать более выразительным, если немного уменьшать или увеличивать длительность каждого сегмента тона. Проще всего применять метод генерации звука, управляемый микросхемой интерфейса с периферией 8255. Меняйте значение бита B1 порта B между 0 и 1, используя программный мультивибратор.

Таблица 1.3

Программа генерирования звука

; — гудок динамика		
	MOV DX,800	**;счетчик числа циклов**
	IN AL,61H	**;читаем порт B ИМС 8255**
	AND AL,0FEH	**;выключаем бит таймера 8253**
NEXTC:	**OR AL,2**	**;готовим бит B1=1 ППИ**
	OUT 61H,AL	**;устанавливаем бит B1 порта B**
	MOV CX,150	**;длительность первой половины**
CYCLEU:	**LOOP CYCLEU**	**;задержка, пока уровень высокий**
	AND AL,0FDh	**;готовим бит B1=0 ППИ**
	OUT 61h,AL	**;сбрасываем бит B1 порта B**
CYCLED:	**LOOP CYCLED**	**;задержка, пока сигнал низкий**
	DEC DX	**;уменьшаем счетчик циклов**
	JNZ NEXTC	**;повторяем цикл, пока DX не 0**

В начале каждого нового пустого цикла, за счет загрузки значения в CX, слегка изменяйте это значение. В таблице 1.4 приведен пример программного генерирования плавного изменения тонов и длительности одновременно.

Таблица 1.4

Генерирование плавно изменяющихся тонов

;— запрет микросхемы таймера		
PB	**EQU 61 H**	**;адрес порта B микросхемы 8255**
	IN AL,PB	**;адрес порта B микросхемы 8255**
	AND AL, 0FEh	**; устанавливаем бит B0=0**
	OUT PB,AL	**; запрещаем работать таймеру**
;— установка частоты и длительности звука		
	MOV BX,9000	**; начальное значение счетчика**
	MOV DX,3000	**;длительность звука 3000 циклов**
REPEAT:		**;сюда возвращаемся после цикла**
;— установка бита динамика		
	OR AL,00000010B	**; подготавливаем бит B1=1 порта B**
	OUT PB,AL	**; устанавливаем бит B1=1**
	MOV CX,BX	**; установка счетчика для 1/2 периода**
CYC1:	**LOOP CYC1**	**; временная пауза на 1000 циклов**
; — сброс бита динамика		
	AND AL,11111101B	**; подготовка бита B1=0 порта B**

	OUT PB,AL	**; установка бита B1=0 порта B**
	MOV CX,BX	**; установка счетчика**
CYC2:	**LOOP CYC2**	**; временная пауза на 1000 циклов**
;— переход к следующему циклу		
	DEC BX	**;увеличиваем частоту переключений,**
	DEC BX	**; уменьшая содержимое счетчика**
	DEC DX	**; уменьшаем длительность генерирования**
	JNZ REPEAT	**; если DX не 0, то новый цикл**

Ассемблер позволяет генерировать нечистые тона, когда интервал, в течение которого динамик включен, не равен интервалу, в течение которого он выключен. Такое нарушение симметрии может приводить к жужжащим и брякающим звукам. Когда отношение этих интервалов составляет, скажем, 50 к 1, получаем жужжание. Если увеличить отношение еще в 10 - 20 раз, то жужжание переходит в отдельные брякающие звуки. В любом случае звук генерируется микросхемой интерфейса с периферией 8255. В таблице 5 представлен пример формирования жужжащих звуков при помощи ППИ 1810ВВ55 (8255).

Таблица 1.5

Программа генерирования жужжащих звуков

NUM_CES	**EQU 300**	**; длительность звучания тона**
FREQ1	**EQU 50**	**; время включенного состояния динамика**
FREQ2	**EQU 3200**	**; время выключенного состояния динамика**
PORT B	**EQU 61H**	**; адрес порта B микросхемы 8255**
	CLI	**; запрет прерываний**
	MOV DX,NUM_CES	**; DX считает длительность тона**
	IN AL,PORT B	**; читаем состояние порта B**
AND AI,11111110B		**; готовим отключение динамика от таймера**
NEXT_CLE:	**OR AL,00000010B**	**; готовим включение динамика**
	OUT PORT B,AL	**; включаем динамик**
	MOV CX, FREQ1	**; установка задержки для импульса**
FIRST_HALF:	**LOOP FIRST_HALF**	**; длительность импульса**
	AND AI,11111101B	**; готовим выключение динамика**
	OUT PORT B,AL	**; выключили динамик**
	MOV CX, FREQ2	**; установка задержки для паузы**
SEC_HALF:	**LOOP SEC_HALF**	**; длительность паузы**
	DEC DX	**; уменьшаем число циклов**
	JNZ NEXT CLE	**; если 0, то пора заканчивать**
	STI	**; разрешаем прерывания**

Для создания брякающих звуков можно использовать этот же код, но надо заменить значение FREQ2 на величину около 40000.

НАПОМИНАНИЕ: *помните, современные операционные системы поддерживают многозадачный режим выполнения приложений. Значит, в момент выполнения наших опытов, выполняются и другие приложения. Таким образом, команды запрещения обслуживания аппаратных прерываний не будут исполняться, а это приведёт к неверному формированию звуковых сигналов. Потому что времена импульсов и пауз будут переменными. Единственное, что нам удастся услышать - это разное время генерирования звуков.*

2. РАБОТА В ЛАБОРАТОРИИ

2.1. Разработать и исследовать программу генерирования однотонного звука при помощи схемы параллельного интерфейса 8255. (обращайте внимание на время звучания, только оно нами ЗАМЕТНО управляется)
2.2. Разработать и исследовать программу генерирования звука, когда ЭВМ ничем другим не занимается (8255).
2.3. Разработать и исследовать программу плавного перехода тонов.
2.4. Разработать и исследовать программы звуковых эффектов («жужжания» и «брякания»).

3. СОДЕРЖАНИЕ ОТЧЕТА

3.1. Общая часть по Вашему усмотрению.
3.2. Блок схему подключения динамика к ЭВМ с указанием значений B0 и B1 для управления через ППИ.
3.3. Алгоритмы программ всех экспериментов с .asm и .lst форматами.
3.4. Покажите, каким образом программно отключается работа канала 2 программируемого таймера.
3.5. По какому принципу программно формируется сигнал управления периферийным устройством, встроенным в системный блок ПЭВМосква:
3.6. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 6

ИССЛЕДОВАНИЕ ПРОГРАММИРУЕМОГО ТАЙМЕРА ДЛЯ УПРАВЛЕНИЯ ПЕРИФЕРИЙНЫМ УСТРОЙСТВОМ – ИСТОЧНИК ЗВУКА

Цель работы: ознакомление с работой программируемого таймера в составе ПЭВМ, основными режимами его работы, способами программирования и перепрограммирования.

1. ОБЩАЯ ЧАСТЬ

Программируемый таймер (ПТ) обозначается КР580ВИ53 и представляет собой однокристальное трехканальное программируемое устройство, предназначенное для получения программно-управляемых временных задержек и выполнения времязадающих функций в микропроцессорных системах. Применение ПТ в системе позволяет повысить эффективность программирования процессов управления и синхронизации внешних устройств, особенно в реальном масштабе времени, т.к. подсчет временных интервалов может производиться параллельно с работой процессора согласно программе управления.

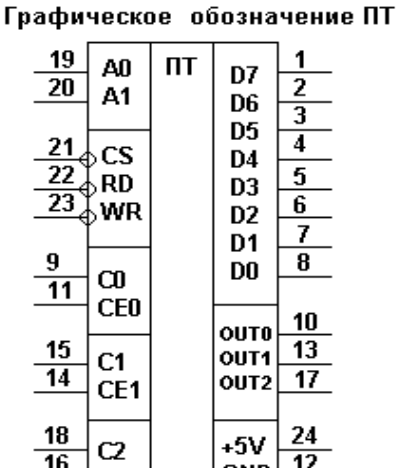

Рис.1.1. Графическое обозначение ПТ

Условное графическое обозначение ПТ приведено на рис.1.1. ПТ может применяться и как самостоятельное устройство при условии выполнения требований, предъявляемых к электрическим и временным параметрам. ПТ по входам и выходам совместим с микросхемами ТТЛ серий. Обмен информацией с центральным процессорным элементом (ЦПЭ) осуществляется по 8 разрядному двунаправленному каналу однобайтными и двухбайтными числами.

Разрядность счетчиков в каждом из трех каналов - 16. Счет может выполняться в двоичном или двоично-десятичном коде. Частота синхронизации каналов от 0 до 2 МГц. Максимальное значение счета в двоичном коде - два в шестнадцатой степени, а в двоично-десятичном - десять в четвертой степени.

1.1. Устройство ПТ

ПТ состоит из следующих функциональных узлов (рис.1.2)

1. Буфер канала данных.
2. Схема выбора канала.
3. Схемы управления чтением "на лету".
4. Схемы канала 0.
5. Схемы канала 1.
6. Схемы канала 2.

Буфер канала данных Д7-Д0 состоит из восьми тристабильных двунаправленных буферов, предназначенных для сопряжения ПТ с шиной данных ЦПЭ. Данные через буфер канала данных принимаются или передаются при выполнении команд ввода или вывода.

Схема выбора канала предназначена для формирования сигналов управления каналами 0, 1, 2, внутренними и внешними передачами данных, приемом управляющих слоев и для выполнения других функций управления. На схему поступают входные сигналы ЧТ, ЗП с шины управления ЦПЭ и сигналы А0, А1, ВМ с адресной шины ЦПЭ.

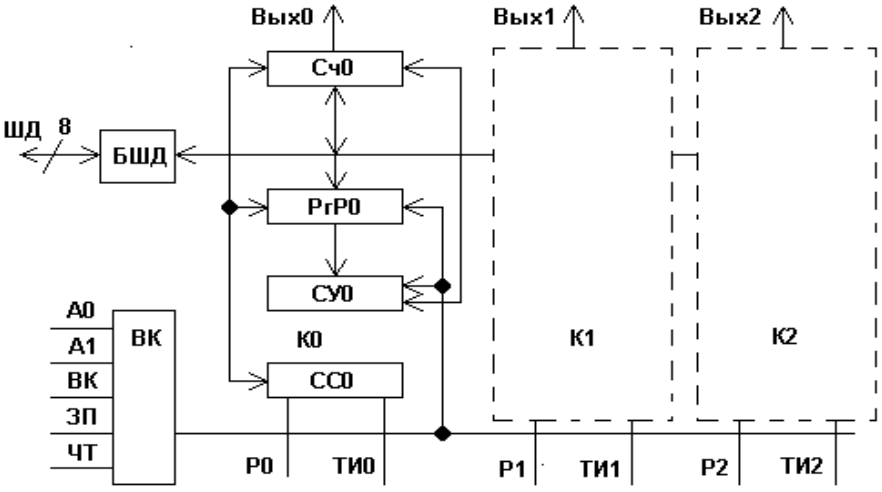

Рис.1.2. Структурная схема ПТ К580ВИ53

До тех пор пока ПТ не выбрана (ВМ = 1), никакие операции записи или чтения в ПТ невозможны, однако сигнал ВМ на работу счетчиков каналов не влияет. Схема управления чтением "на лету" позволяет прочитать содержимое счетчиков каналов, не прерывая их текущего счета. Схемы каналов 0,1,2 идентичны, т.е. все каналы одинаковы, независимы, могут работать и в разных режимах.

В состав каждого канала входит 16 разрядный вычитающий счетчик. Скорость вычитания (декодированная) определяется частотой подаваемых на входы ТИ0, ТИ1, ТИ2 импульсов. В каждом канале находится свой регистр режима, предназначенный для хранения кода управляющего слова (УС). Код

УС задает режим работы канала, определяет тип счетчика (двоичный или двоично-десятичный) и последовательность загрузки данных в счетчик. Информацию в регистр можно только записывать. Прочитать содержимое этого регистра невозможно.

16 разрядный вычитающий счетчик выполняет счетные операции над однобайтными или двухбайтными числами. Счетчики всех каналов независимы и могут иметь различные режимы работы и различные типы счета. Загрузка в счетчик нулевого значения позволяет получить максимально возможную величину счета. Информация из счетчика передается в буферный регистр и при необходимости может быть прочитана в аккумулятор ЦПЭ.

1.2. Программирование ПТ

Перед использованием ПТ в процессе решения задач управления необходимо проинициализировать ПТ, т.е. произвести программирование каждого из используемых каналов. Для приведения каждого канала ПТ в требуемое состояние, соответствующее выбранному режиму, и для загрузки его информацией о типе счета необходимо загрузить в ПТ некоторый набор управляющих слов УС. Эти УС программируют режим, очередность загрузки, тип счета. После программирования ПТ готов к выполнению задач, связанных с отсчетом времени. Режимы работы каждого канала определяются содержимым регистра режима, то есть содержимым УС. После записи УС в регистр режима выбранного канала он переводится в один из шести основных режимов работы:

1. Режим 0 (прерывание терминального счета).
2. Режим 1 (ждущий мультивибратор).
3. Режим 2 (генератор частоты импульсный, несимметричный мультивибратор)
4. **Режим 3 (Генератор меандра, симметричный мультивибратор).**
5. Режим 4 (одиночный программно формируемый строб).
6. Режим 5 (одиночный аппаратно формируемый строб).

Поскольку в данной лабораторной работе нас будет интересовать только режим 3 (генератор меандра), то подробно будем рассматривать только этот режим. Временные диаграммы работы ПТ в режиме 3 (генератор Меандра) показаны на рис.1.3. Этот режим во всем аналогичен режиму 2, за исключением того, что длительность положительного и отрицательного полупериодов выходного сигнала канала ПТ для четных чисел N равна периоду входной частоты, умноженной на половину N, а для нечетных - положительный полупериод равен периоду входной частоты, умноженной на половину N, отрицательный - периоду входной частоты умноженной на (N-1)/2.

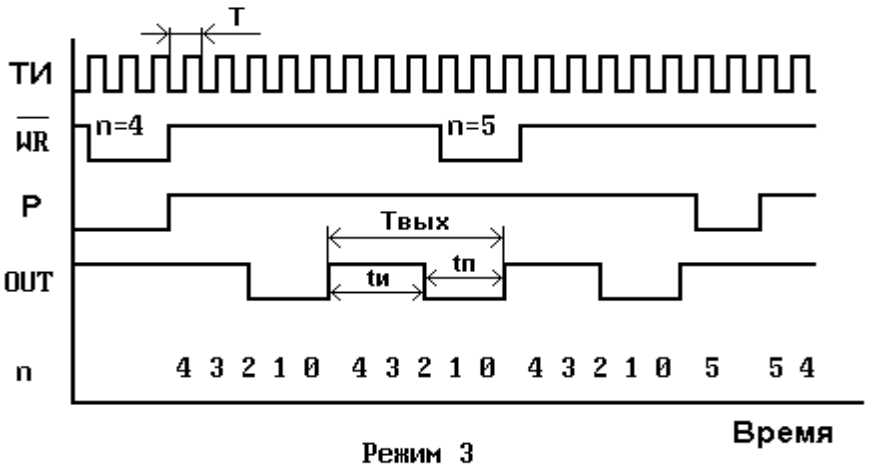

Рис.1.3. Временные диаграммы работы ПТ в режиме 3

Формат управляющего слова приведен на рис.1.4.

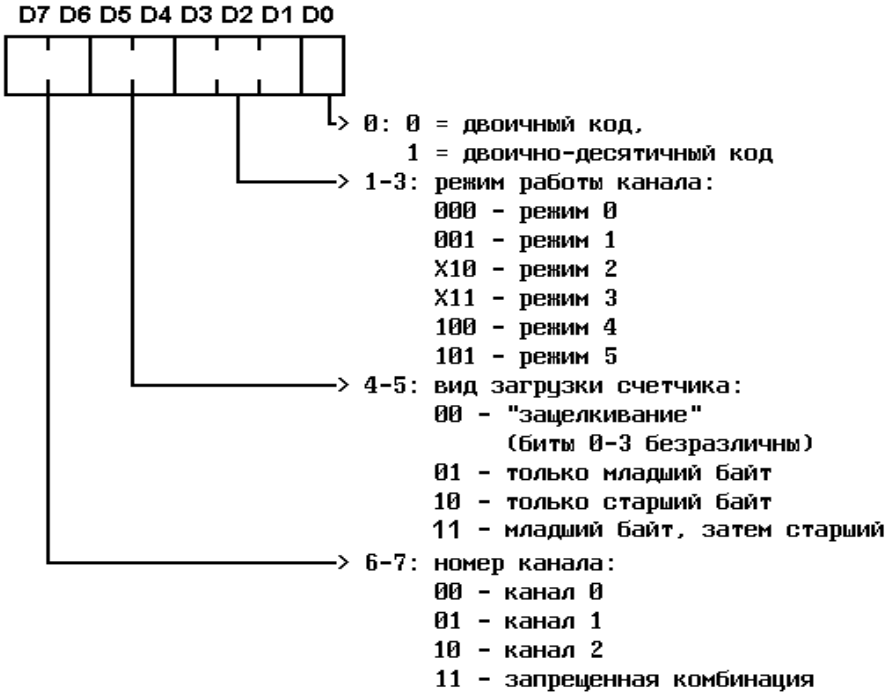

Рис.1.4. Формат управляющего слова ПТ К580ВИ53

1.3. Пример программирования ПТ

Пусть требуется запрограммировать ПТ в режим 0 и загрузить в счетчик двухбайтное число 1254Н, тип счета двоичный. Для программирования выбран канал 2.

Используя формат управляющего слова (рис.1.4), формируем управляющее слово для нужного режима, получили:

Д7	Д6	Д5	Д4	Д3	Д2	Д1	Д0	
1	0	1	1	0	0	0	0	- B0

Это УС должно быть загружено в регистр режима (регистр для хранения УС). Допустим, что в ПЭВМ адреса каналов следующие: канал 0 (К0) - Е0, канал 1 (К1) - Е1, канал 2 (К2) - Е2, регистр режима Е3.

Тогда для программирования канала 2 необходимо УС = B0H записать в адрес Е3. Затем младший байт числа -54h, за ним старший - 12h, оба этих байта должны быть загружены в адрес Е2.

1.4. Использование программируемого таймера в ПЭВМ

Если необходимо получить какие-либо сложные звуки, то необходимо программировать микросхему таймера 580ВИ53 (1810ВИ54,8253). Канал 2 этой микросхемы прямо связан с динамиком компьютера. Когда этот канал программируется на режим 3 (симметричный мультивибратор), таймер формирует прямоугольные импульсы заданной частоты. Динамик имеет не один, а два входа для генерации звука. На рис. 1.5. показано, что, кроме микросхемы таймера, сигнал на динамик поступает также с микросхемы параллельного ввода/вывода 580ВВ55(8255).

Каждый из трех каналов микросхемы таймера 8253 (8254 для AT) состоит из шестнадцатиразрядных регистров (отечественные аналоги 580ВИ53 и 1810ВИ54). **Доступ к ним осуществляется через один порт; номера портов 40H, 41H и 42H соответствуют каналам К0, К1, К2. Адрес регистра режима 43H.**

Когда значение числа в счётчике достигает нуля, канал выдает выходной сигнал и затем новая копия содержимого регистра задвижки пересылается в регистр таймера счетчика, после чего процесс повторяется.

Чем меньше число в регистре счетчика, тем больше выходная частота канала. Все три канала всегда активны, то есть их работа не зависит от поведения процессора. Текущее значение любого из регистров счетчика может быть прочитано в любой момент времени, что не влияет на счет. Каждый канал имеет две входные (ТИ - тактовые импульсы и Р - разрешение работы канала) и одну выходную линии.

Выходная линия выводит импульсы канала, возникающие в результате подсчета. Назначение этих сигналов варьируется в зависимости от типа IBM PC.

Каждый импульс канала 0 инициирует прерывание таймера (INT 8) и именно это прерывание увеличивает показание счетчика.

Это аппаратное прерывание, поэтому оно обрабатывается всегда независимо от того, чем занят процессор, если только разрешены аппаратные прерывания. Выходная линия используется также для синхронизации некоторых

дисковых операций, поэтому если вы изменили ее значение, то вам необходимо восстановить первоначальное значение перед обращением к диску.

Канал 1 управляет обновлением памяти на всех машинах, поэтому с ним лучше не экспериментировать. Выходная линия этого канала связана с микросхемой прямого доступа к памяти, и ее импульс заставляет микросхему DMA обновить всю память.

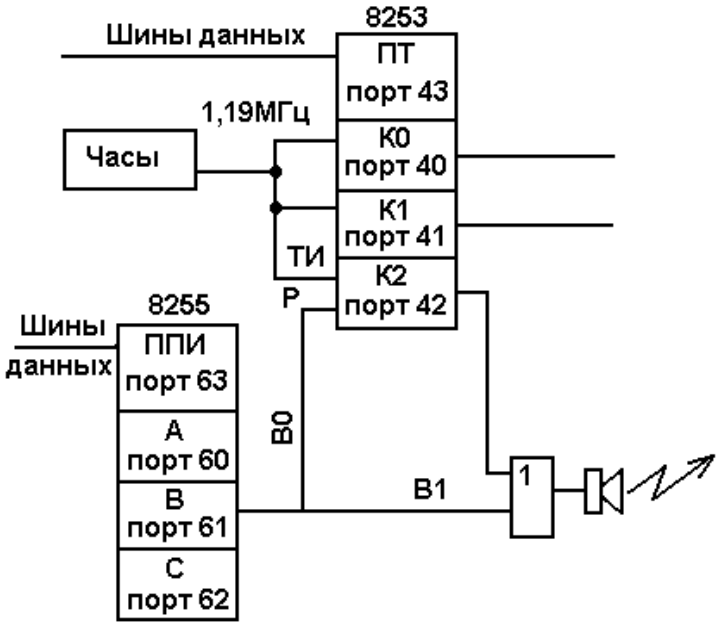

Рис.1.5. Схема подключения динамика в ПЭВМ

Канал 2 связан с громкоговорителем ком-пьютера. Он формирует прямоугольные импульсы для генерации звука. Программисты имеют возможность управлять вто-рым каналом. Простые звуки могут генериро-ваться одновременно с другими программными операциями, а более слож-ные звуковые эффекты могут быть достигнуты за счет использования процессора. Канал 2 может быть отсоединен от громкоговорителя программным способом (линия В0 порта 61Н). Однако динамик не будет генерировать звук до определенных установок микросхемы интерфейса с периферией.

Две входные линии для каждого канала состоят из линии часов, передающей сигнал от микросхемы системных часов и линии, называемой разрешения (Р), которая включает и выключает сигнал от часов. Сигналы разрешения всегда открыты для сигналов часов по каналам 0 и 1. Но может быть закрытым для канала 2. Что позволяет производить некоторые специальные манипуляции со звуком.

Микросхема таймера может применяться непосредственно для временных операций, но это редко бывает удобным. Ввод с часов производится с частотой 1,19318 МГц (даже на АТ, где системные часы идут быстрее, микросхема таймера получает сигнал с частотой 1,19 МГц). Поскольку максимальное число, которое может храниться в 16 битах, равно 65535, то обнуление счетчика таймера произойдет через 65535 периодов входной частоты канала, если такое число было записано в счетчик канала. Значит, максимальное время, формируемое одним каналом таймера, равно

$$65535/(1{,}19*10^{6})\text{сек.}, \text{ или } 18{,}2 \text{ Гц.}$$

Поэтому в большинстве временных операций используется счетчик времени суток BIOS. Для подсчета времени читается значение времени суток и сравнивается с некоторым ранее запомненным значением для определения числа импульсов, прошедших с того момента. Специальный способ, описанный в [1], позволяет использовать счетчик времени суток для операций в реальном времени.

Для управления динамиком от программируемого таймера программно необходимо установить биты управления порта с адресом 61H в следующее состояние: B0 = 1, разрешить работу каналу 2 (порт 42H) таймера; B1 = 0, снять влияние на динамик со стороны программируемого параллельного ввода/вывода.

Таким образом, для программирования микросхемы 8253 надо выполнить три основных шага. После того как третий шаг завершен, запрограммированный канал немедленно начинает функционировать по новой программе.

1. Послать в командный регистр (порт 43H) байт, представляющий цепочку битов, которые выбирают канал, статус чтения/записи, режим операции и форму представления чисел.

2. Для канала 2 надо разрешить сигнал от часов, установив в 1 бит B0 порта с адресом 61h. (Когда бит B1 этого регистра установлен в 1, канал 2 управляет динамиком).

3. Вычислить значение счетчика от 0 до 65535, поместить его в AX и послать сначала младший (AL), а затем старший байт (AH) в регистр канала 2 (порт 42H).

Каналы таймера работают постоянно. По этой причине программы всегда должны восстанавливать начальные установки регистров 8253 перед завершением опытов с таймером. В частности, если при завершении программы генерируется звук, то он будет продолжаться даже после того, как операционная система получит управление и загрузит другую программу. Имейте это в виду при написании процедуры выхода по «Ctrl-Break».

1.5. Генерация тона

Как генерировать звук, когда компьютер не занят ничем другим? Для этого достаточно запрограммировать микросхему таймера, которая работает независимо от процессора. При использовании ППИ для генерирования звука процессор непосредственно управляет динамиком, поэтому программе приходится выполнять работу, которую может выполнять микросхема таймера.

Частота задается в герцах (от 37 до 32767), а длительность в импульсах счетчика времени суток BIOS (от 0 до 65535), причем в секунду формируется 18,2 импульса. Частоты первой октавы, начиная с ноты C(до), таковы:

Таблица 1.1

Нота	Частота в Гц
C(до)	523,3
D(ре)	587.3
E(ми)	659,3
F(фа)	698,5
G(соль)	784,0
A(ля)	880,0
B(си)	987,7

Поскольку микросхема таймера 8253 работает независимо от процессора, очень легко генерировать звук, который издается одновременно с выполнением других операций. Для этого необходимо просто запрограммировать канал 2 этой микросхемы для генерации определенной частоты, а затем перепрограммировать микросхему для выключения звука. Запрограммируйте канал 2 таймера. Работа таймера с динамиком должна быть предварительно разрешена через порт B микросхемы интерфейса с периферией 8255 (адрес 61H).

Вычислите требуемое значение счетчика для загрузки канала 2, разделив 1,19 МГц на требуемую частоту. Звук будет продолжаться до тех пор, пока не будет снят сигнал разрешения канала 2. Для этого необходимо сбросить бит B0 порта B в 0, иначе звук будет продолжаться бесконечно и может быть прекращен только при перезагрузке компьютера. Для точного регулирования длительности звука можно использовать счетчик времени суток BIOS. В данном примере генерируется частота 440 Гц. Звук прекращается после нажатия любой клавиши на клавиатуре. В таблице 1.2 приведен исходный текст программы генерирования звука при помощи таймера 8253.

Таблица 1.2

Исходный текст программы генерирования звука таймером 8253

; — разрешение канала 2 установкой порта B микросхемы 8255		
PORT B	EQU 61H	;установка адреса порта B
	IN AL,PORT B	; чтение его значения
	OR AL,3	;установка двух младших битов
	OUT PORT B,AL	;посылаем байт в порт B
; — установка регистров ввода/вывода		
COM	EQU 43H	;адрес командного регистра
CHAN 2	EQU 42H	;адрес канала 2
	MOV AL,10110110B	;цепочка битов для канала 2
	OUT COM,AL	;засылка в командный регистр
; — засылка числа в счетчик таймера		
	MOV AX,2705	;счетчик → 1190000/440 =2705
	OUT CHAN 2,AL	;посылаем младший байт в канал 2
	MOV AL,AH	;сдвигаем младший байт и AL

	OUT CHAN 2,AL	;посылаем старший байт в канал 2
; — ждем нажатия клавиши		
	MOV AH,1	;номер функции прерывания INT 21h
	INT 21H	;вызываем прерывание 21h
; — выключение звука		
	IN AL,PORT B	;получаем байт из порта B
	AND AL,11111100B	;сбрасываем два младших бита порта B
	OUT PORT B,AL	;посылаем байт обратно

1.6. Генерирование набора тонов

Построение звуков в ассемблере требует большой работы. Может быть использован любой из двух методов генерации звука, рассмотренных ранее. Для обоих методов надо просто генерировать один тон в течение заданного времени, затем следующий и т.д. Каждая звуковая строка формируется из двух строк данных, одна из которых содержит частоты последовательных тонов, а другая хранит их длительности (при условии, что требуются разные длительности). Продолжительность звучания определяется с помощью счетчика времени суток BIOS.

В примере (таблица 1.3) для генерации звука используется микросхема таймера 8253. Здесь просто исполняются 8 нот, но небольшая модификация может сильно расширить возможности этой процедуры. Имеются три строки данных. Первая устанавливает длительность каждой ноты как кратное произвольного периода задержки (изменяя этот период задержки, можно изменять темп). Вторая строка содержит частоты каждой из 8 нот; эти значения должны быть помещены в регистр задвижки канала 2 микросхемы 8253 для исполнения желаемых тонов.

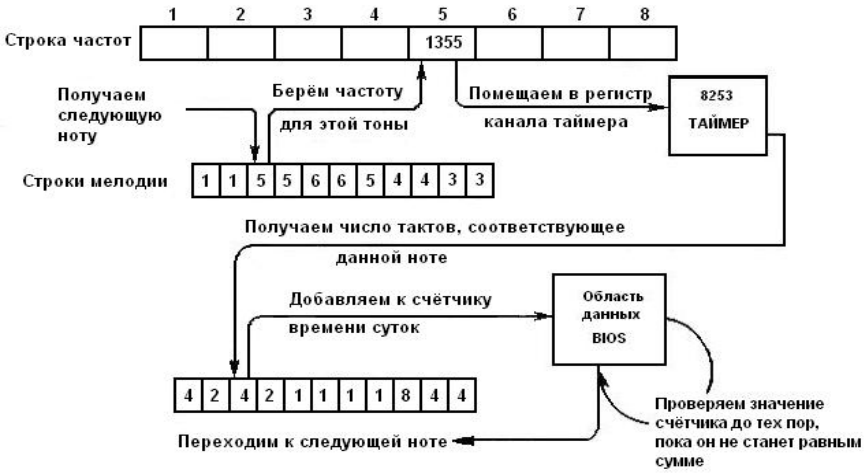

Рис. 1.6. Алгоритм исполнения строки нот

Третья строка содержит мелодию в виде кодовых номеров от 1 до 8, которые соответствуют восьми частотам. Эта строка завершается кодом 0FFH, который служит признаком конца мелодии. Процедура просто читает очередную ноту мелодии, находит соответствующую частоту и помещает ее в канал 2. Затем длительность для этой ноты помещается в счетчик цикла задержки, который использует счетчик времени суток. Когда задержка кончается, переходим к обработке следующей ноты. На рис. 1.6. показана работа этой процедуры.

Таблица 1.3

Программа генерирования набора тонов

; — в сегменте данных		
BEAT	DB 10,9,8,7,6,5,4,3,2	; длительность нот
FREQ	DW 2280,2031,1809,1709	;таблица частот
	DW 1521,1353,1207,1139	
MELODY	DB 1,2,3,4,5,6,7,8,0FFH	;номер частоты ноты
; — инициализация		
PORT_B	EQU 61H	
COM_REG	EQU 43H	
LATCH2	EQU 42H	
	IN AL,PORT_B	;получаем текущий статус
	OR AL,00000011B	;разрешаем динамик и таймер
	OUT PORT B,AL	;заменяем байт
	MOV SI,0	;инициализируем указатель
	MOV AL,0B6H	;установка для канала 2
	OUT COM_REG,AL	;посылаем в командный регистр
; — смотрим ноту, получаем ее частоту и помешаем в канал 2		
NEXT:	LEA BX,MELODY	;берем смещение для мелодии
	MOV AL, [BX] [SI]	;берем код n-й ноты строки
	CMP AL,0FFH	;проверка на конец строки
	JE NO_MORE	;если конец, то на выход
	CBW	;переводим в слово
;получение частоты		
	MOV BX,OFFSET FREQ	;умещение таблицы частот
	DEC AX	;начинаем отсчет с 0
	SHL AX,1	;умножаем на 2, так как словная таблица
	MOV DI,AX	;адресуем через DI
	MOV DX,[BX] [DI]	;получаем частоту из таблицы
;начинаем исполнение ноты		

	MOVAL.DL	;готовим младший байт частоты
	OUT LATCH2,AL	;посылаем его
	MOV AL,DH	;готовим старший байт частоты
	OUT LATCH2,AL	;посылаем его
; — создание цикла задержки		
	MOV AH,0	;номер функции чтения счетчика
	INT 1AH	;получаем значение счетчика
	MOV BX,OFFSET BEAT	;смещение таблицы длин
	MOV CL, [BX] [SI]	;берем длину очередной ноты
	MOV CH,0	
	MOV BX,DX	;берем младшее слово счетчика
	ADD BX,CX	;определяем момент окончания
ST_SOU:	INT 1AH	;берем значение счетчика
	CMP DX,BX	;сравниваем с окончанием
	JNE ST_SOU	;неравны - продолжаем звук
	INC SI	;переходим к следующей ноте
	JMP NEXT	
; — завершение		
NO_MOR:	IN AL,PORT_B	;получаем статус порта B
	AND AL,0FCH	;выключаем динамик
	OUT 61H,AL	;заменяем байт

1.7. Генерация строки тонов одновременно с другими операциями**

** Означают, что этот опыт можно делать по личной инициативе для расширения собственных интересов.

Для решения такой задачи нужна генерация звука через микросхему 8253, так как метод, использующий микросхему 8255, занимает процессор. Соответственно только строки чистых музыкальных тонов могут производиться таким методом. Всякого рода звуковые эффекты при этом недоступны. Программы, работающие в реальном времени, модифицируют прерывание таймера, которое останавливает процессор 18,2 раза в секунду, чтобы изменить показание счетчика времени суток. Расширение процедуры прерывания сравнивает новое значение счетчика времени суток со значением, показывающим время завершения генерации тона, и когда это значение достигнуто,

прерывает звук, начинает генерацию другого тона и устанавливает время его окончания.

Приведенная процедура является развитием процедуры, показанной в предыдущем параграфе, на случай реального времени. Она требует понимания, как перепрограммировать прерывание таймера. На эту процедуру должен указывать вектор прерывания и тогда она будет выполняться 18,2 раза в секунду в те моменты, когда будет обновляться значение счетчика времени суток BIOS. Обычно выполняется только несколько строчек, которых достаточно, чтобы определить, что время изменения звука еще не наступило и процедура освободит процессор для решения других задач.

Счетчик времени суток BIOS применяется для измерения длительности каждой ноты. При переходе от одной ноты к другой длительность новой ноты вычисляется как число импульсов счетчика и это значение добавляется к текущему его значению. Каждый раз при вызове процедуры проверяется текущее значение счетчика времени суток, и когда ожидаемое время, наконец, наступает, выполняется набор операций по поиску новой ноты, программированию ее частоты в канале 2 микросхемы 8253 и установлению нового значения счетчика длительности. Добавочный код требуется для обработки специальных случаев первой и последней нот в строке. В таблице 1.4 приведена программа генерирования последовательности тонов одновременно с другими операциями.

Таблица 1.4

Программа генерирования строки тонов
одновременно с другими операциями

; — в сегменте данных		
BEAT	DB 10,9,8,7,6,5,4,3,2	;длительность нот
FREQ	DW 2280,2031,1809,1709	;таблица частот
	DW 1521,1355,1207,1139	
MELODY	DB 1,2,3,4,5,6,7,8,0FFH	;номер частоты в таблице
HOLDIP	DW 0	;запоминаем оригинальный
HOLDCS	DW 0	;вектор прерывания
SOU_NOW?	DB 1	;звук включен?
FIR_NOTE?	DB 1	;первая нота?
END NOTE	DW 0	;счетчик конца ноты
WHICH NO	DW 0	;указатель на текущую ноту
; — инициализация вектора прерывания изменение вектора		
	PUSH DS	;сохраняем регистр
	MOV AX,SEG MEL2	;сегмент процедуры

	MOV DS,AX	;помещаем в DS
	MOV DX,OFFSET MEL2	; смещение процедуры
	MOV AL,1CH	;номер вектора прерывания
	MOV AH,25H	;функция установки вектора
	INT 21 H	;изменение вектора
	POP DS	; восстановление регистра
; — программа работает дальше, постоянно вызывая процедуру ; — в конце программы восстанавливаем вектор прерывания		
	MOV DX,0FF53H	; восстанавливаем оригинальные
	MOV AX,0F000H	;значения для вектора 1CH
	MOV DS,AX	
	MOV AL,1CH	;номер прерывания
	MOV AH,25H	;функция установки вектора
	INT 21 H	; восстанавливаем вектор
	RET	
; — это само прерывание		
MEL2	PROC FAR	
	PUSH AX PUSH BX PUSH CX	;сохраняем изменяемые регистры
	PUSH DX PUSH DI PUSH SI PUSH DS	
	MOV AX,SS:[114]	; берем начальный DS из стека
	MOV DS,AX	;восстанавливаем его
	CMP SOU_NOW?,1	; нужен ли звук?
	JE PLAY IT	; если нет, то выход из прерывания
	JMP NOT NOW	
PLAY IT:	CMP FIR NOTE?,0	; это первая нота?
	JE TIM CHK	; если нет, то па установку времени
; — инициализация		
PORT B	EQU 61 H	; определяем имена портов
COM REG	EQU 43H	
LATCH2	EQU 42H	
	IN AL,PORT B	;берем статус порта B
	OR AL,00000011B	;разрешаем динамик и таймер
	OUT PORT B,AL	;посылаем байт обратно
	MOV SI,0	;указатель на строки

	MOV AL,0B6H	;инициализация канала 2 таймера
	OUT COM_RBG,AL	;посылаем в командный регистр
	MOV FIR_NOTE?,0	;сбрасываем флаг первой ноты
; — ищем ноту, получаем ее частоту, посылаем в канал 2		
NEXT:	LEA BX, MELODY	;берем смещение строки мелодии
	MOV SI,WHICH_NO	; указатель на текущую ноту
	MOV AL, [BX] [SI]	;код текущей ноты строки
	CMP AL,0FFH	;проверяем признак конца
	JE NO_MOR	;если да, то на конец
	CBW	;иначе - в словный формат
; получаем частоту		
	MOV BX, OFFSET FREQ	;смещение таблицы частот
	DEC AX	;начинаем отсчет с нуля
	SHL AX,1	;умножаем на 2, так как словная таблица
	MOV DI,AX	;адресуемся через DI
	MOV DX,[BX][DI]	; получаем частоту из таблицы
;начинаем исполнение ноты		
	MOV AL,DL	;готовим младший байт частоты
	OUT LATCH2,AL	;посылаем и регистр канала таймера(42h)
	MOV AL,DH	;готовим старший байт
	OUT LATCH2,AL	;посылаем его в порт 42h
; — пустой цикл, определяющий длительность нот		
TIME_IT:	MOV AH,0	;функция чтения счетчика
	INT 1AH	;получаем значение счетчика
	MOV BX,OFFSET BEAT	;смещение строки длин нот
	MOV CL, [BX] [SI]	; длительность текущей ноты
	MOV CH,0	
	MOV BX,DX	;младшее слово значения счетчика
	ADD BX,CX	; добавляем длину в импульсах
	MOV END_NOTE,BX	; запоминаем время окончания
TIM_CHK:	MOV AH,0	;функция чтения счетчика
	INT 1AH	;читаем счетчик
	CMP DX,END NOTE	; сравниваем с кодом завершения
	JNE NO NOW	;если неравно, то выходим
	MOV SI, WHICH NO	;иначе, берем следующую ноту
	INC SI	; увеличиваем номер ноты
	MOV WH1CH_NO,SI	; запоминаем его
	JMP NEXT_ NOTE	;начинаем следующую ноту

; — завершение процедуры		
NO_MOR:	IN AL,PORT_B	;читаем состояние порта B
	AND AL,0FCH	; выключаем динамик
	OUT 61h,AL	;возвращаем байт
	MOV SOU_NOW?,0	;восстанавливаем переменные
	MOV FIR_NOTE?,1	
NO NOW:	POP DS POP SI POP Dl POP DX POP CX POP BX POP AX	;восстанавливаем регистры
	IRET	;возврат из прерывания
	MEL2 ENDP	

2. РАБОТА В ЛАБОРАТОРИИ

2.1. Подготовить и исследовать программу генерирования однотонного звука при помощи схемы таймера 8253.
2.2. Подготовить и исследовать программу плавного перехода тонов.
2.3. Подготовить и исследовать программу генерирование звука одновременно с другими действиями (8253).
2.4.** Подготовить и исследовать программу звуковых эффектов.

3. СОДЕРЖАНИЕ ОТЧЕТА

3.1.Общая часть по Вашему усмотрению.
3.2. Блок - схему подключения динамика к ЭВМ с указанием значений B0 и B1.
3.3. Управляющее слово для Вашего варианта программирования ПТ.
3.4. Алгоритмы программ всех экспериментов с .asm и .lst форматами.
3.5. Как программируется таймер (инициализация таймера).
3.6. Как в программе управления ПТ изменяется частота генерирования звука.
3.7. Выводы.

ЛАБОРАТОРНАЯ РАБОТА № 7

ИССЛЕДОВАНИЕ ПРОГРАММИРУЕМОГО КОНТРОЛЛЕРА ПРЕРЫВАНИЙ

Цель работы: изучение способов обслуживания аппаратных прерываний с применением программируемых контроллеров прерываний, знакомство с правилами программирования, обслуживания и управления.

1.ОБЩИЕ СВЕДЕНИЯ

Для управления аппаратными прерываниями во всех типах ПЭВМ IBM PC используется микросхема программируемого контроллера прерываний Intel 8259A или ее программно-совместимый аналог. Поскольку в каждый момент времени может поступить не один запрос, микросхема имеет схему приоритетов. Существует 8 уровней приоритетов, максимальный приоритет соответствует запросу 0 (ЗПР0). Дополнительные 8 уровней (ЗПР 8 - ЗПР 15) обрабатываются второй микросхемой 8259A, подключенной каскадно; этот второй набор уровней имеет приоритет между ЗПР 2 и ЗПР 3 (рис. 1.1) [3]. Запросы на прерывания по входам ЗПР 0 ÷ 7 соответствуют векторам прерывания от INT 8 до INT F. Запросы на прерывания по входам ЗПР 8 ÷ 15 обслуживаются векторами INT 70 - INT 77. Исключением из общего правила является аппаратное немаскируемое прерывание NMI, которое имеет максимальный приоритет, поскольку является немаскируемым прерыванием.

Микросхема 8259A имеет три однобайтовых регистра, которые управляют восемью линиями аппаратных прерываний (рис.1.2). Регистр запроса на прерывание (IRR) устанавливает соответствующий бит, когда на линии прерывания уровень сигнала изменяется от низкого к высокому. Затем микросхема автоматически проверяет, не обрабатывается ли другое прерывание. При этом используется информация из регистра обслуживания (ISR). Дополнительная цепь отвечает за схему приоритетов. Перед вызовом прерывания проверяется регистр маски прерываний (IMR), чтобы узнать, разрешено ли в данный момент прерывание данного уровня. Если необходимые условия обработки запроса прерывания выполнены, то контроллер 8259A выдает процессору сигнал ЗПР (INT), процессор завершает выполнение текущей команды и проверяет бит разрешения прерываний в регистре признаков (флаги).

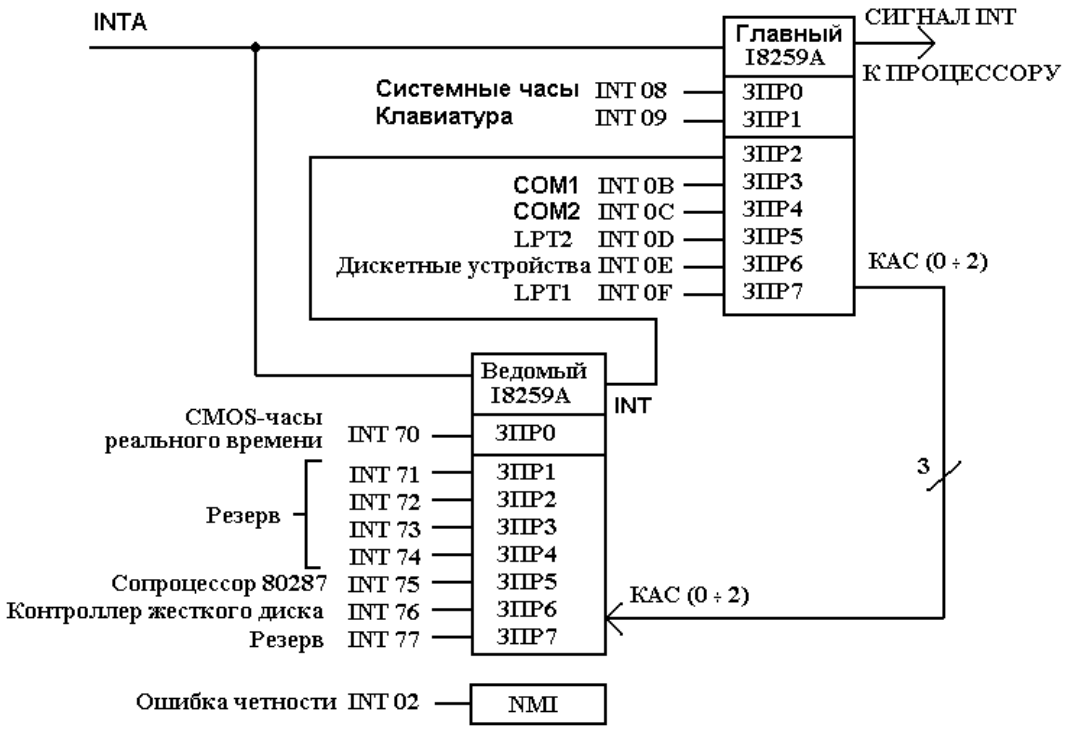

Рис.1.1. Структурная схема системы прерываний IBM PC

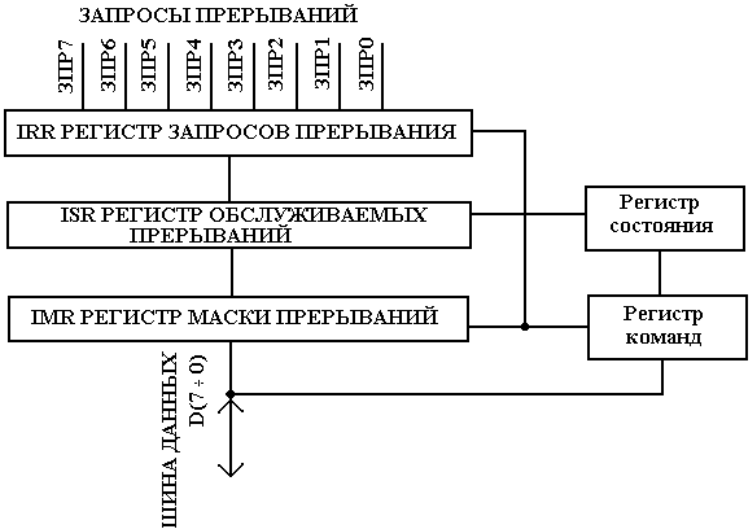

Рис.1.2. Программная модель контроллера 8259A

Если этот бит установлен, то процессор начинает выполнять цикл чтения кода команды с шины данных, сопровождая свои действия сигналом подтверждения прерывания ППР (INTA). По этому сигналу контроллер прерываний формирует на шине данных код команды INT, а по второму сигналу ППР – байт, определяющий номер прерывания (вход запроса). Его значение зависит от номера входа ЗПР у контроллера прерываний, которое в данный момент должно быть обслужено. Процессор сохраняет в стеке логический адрес прерывания (CS:IP) и содержимое флагового регистра (словосостояние программы

PSW). Затем процессор вычисляет адрес ячейки ОЗУ, предназначенный для обслуживания этого запроса (номер команды умножается на 4 в шестнадцатиричной системе счисления). Начиная с этого адреса, в ОЗУ загружен длинный логический адрес начала подпрограммы обслуживания этого запроса (запись логического адреса начала подпрограммы делается на этапе инициализации данного запроса). Этот адрес процессор пересылает в CS и IP , что является скрытой командой JMP CS:IP которую процессор не читал. После завершения обслуживания в подпрограмме необходимо командой управления OCW1/EOI в контроллере 8259A сбросить бит регистра ISR с максимальным приоритетом, в противном случае обработка прерываний для уровня с текущим приоритетом и уровней с более низкими приоритетами будет блокирована.

Доступ к внутренним регистрам контроллеров осуществляется через порты 20h, 21h для ведущего контроллера и A0h, A1h - для подчиненного. Порты 21h, A1h предназначены для доступа к регистрам маски, формат регистров маски показан на рис.1.3.

Рис.1.3. Формат регистра маски

Порты 20h и A0h используются для управляющих команд, формат команд показан на рис.1.4, 1.5. Форматы внутренних регистров контроллера IRR, ISR соответствуют формату регистра маски. Формат регистра состояний показан на рис.1.6. На этапе начального пуска программой BIOS выполняется инициализация контроллера прерывания. В процессе инициализации устанавливаются значения векторов прерываний и дисциплина обслуживания приоритетных уровней.

Рис.1.4. Формат команды управления OCW 0

В лабораторной работе используется аппаратное прерывание INT 8 от микросхемы таймера. Стандартный обработчик данного прерывания поддерживает системные часы операционной системы MS DOS, контролирует длительность паузы при выполнении операций с гибким диском, а также вызывает программное прерывание INT 10 предназначенное для подключения обработчиков прерываний, разработанных пользователем.

Векторы прерываний, используемые операционной системой, образуют таблицу векторов и располагаются в начальной области оперативной памяти, начиная с нулевого физического адреса (00000h - 003ffh).

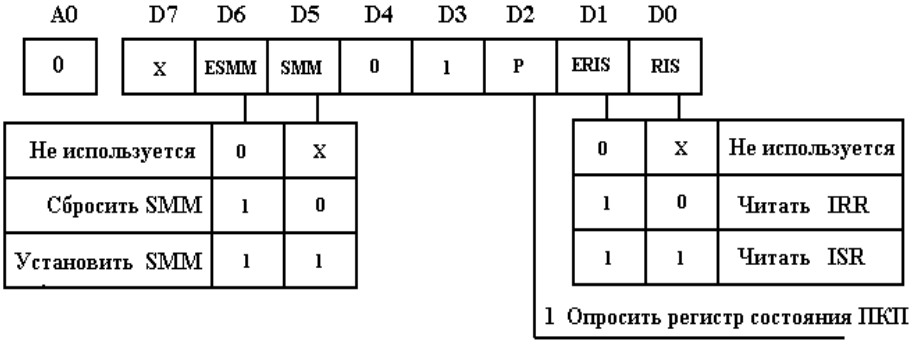

Рис.1.5.Формат команды управления OCW1

Рис.1.6. Формат регистра состояния контроллера прерываний в режиме опроса

Таблица векторов занимает 1К оперативной памяти и содержит 256 векторов. На каждый вектор отводится четыре байта. Вектор прерываний имеет следующий формат в оперативной памяти:

Ячейка +2	CS -- сегмент точки входа в обработчик прерываний
Ячейка +0	IP - смешение точки входа в обработчик прерываний

2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Загрузить отладчик AFDPRO или TIC .
2. С помощью HELP (F4) изучить команды отладчика.
3. Прочитать содержимое регистров маски (команды I “порт”, O “ порт“):
21h - главный контроллер;

A1h - подчиненный контроллер.
Пользуясь схемой (рис. 1.1) расшифровать содержимое регистров маски, результаты выписать в отчет.
4. Установить маску на клавиатуру, не изменяя бит маски остальных источников прерываний, код новой маски записать в порт 20. Убедиться, что клавиатура отключена.
5. Набрать, начиная с адреса 100, текст программы:

```
      CLI
      MOV CX,FFFF
L1:   NOP
      NOP
      LOOP L1          ; при наборе вместо L1 поставить адрес метки L1
      MOV AL,0A        ; команда управления «читать IRR»
      OUT [20], AL     ; загрузка в ПКП кода 0Ah для чтения регистра запросов
M:    IN AL, [20]      ; чтение регистра IRR

      MOV BL,AL        ; переслать в BL значение регистра IRR
      MOV AL,0B        ; команда управления «читать ISR»
      OUT [20], AL     ; загрузка в ПКП кода 0Bh для чтения регистра состояния
      IN AL, [20]      ; чтение регистра ISR
      STI              ; разрешить прерывания
L2:   NOP      ; установить контрольную точку останова на этой команде
```

Программа позволяет читать внутренние регистры контроллера, используя команду управления OCW1. Расшифровать полученные результаты чтения регистров IRR, ISR.
6. Изучить возможности AFDPRO или TIC для установки контрольных точек программы (нажать клавиши F5, F4). Установить контрольную точку останова на метке L2 (метка – применить команду передачи управления на адрес команды, на которую необходимо сделать передачу).
7. Многократно запуская программу по команде g100, фиксировать в отчете значения регистров IRR и ISR ведущего контроллера.
8. Установить в программе, в команде, помеченной меткой M, режим опроса регистра состояния контроллера.
9. Запустить программу по команде g100, зафиксировать в отчете значения регистра признаков (флагов) и регистра ISR. При зависании машины выполнить перезагрузку и вернуться в AFDPRO или TIC. Объяснить причину зависания.
В отчете расшифровать содержимое внутренних регистров IRR, ISR и слова состояния контроллера.

10. Рассчитать адрес вектора прерывания системного таймера по формуле

АДРЕС = номер прерывания (см. рис.1.1) * 4;

Записать в отчет адрес вектора и его содержимое. Запомнить текущее содержимое регистра CS. Записать в CS значение из вектора. В окне дизассемблера найти точку входа в обработчик прерываний таймера и записать в отчет три первых команды обработчика прерываний системного таймера. Восстановить прежнее содержимое регистра CS.
11. Рассчитать адрес вектора системного прерывания INT 1Ch. Записать в отчет адрес вектора и его содержимое. Запомнить текущее содержимое регистра CS. Записать в CS значение из вектора. В окне дизассемблера найти точку входа в обработчик прерываний
INT 1Ch и записать в отчет три первых команды обработчика. Восстановить прежнее значение регистра CS.
12. Набрать текст программы обработки прерываний таймера, начиная с адреса CS:100:

```
PUSH  AX              ; отправить в стек AX
ADD  CS: [30], 1      ;
MOV  AL,20            ; загрузить в AL код 20h – конец обслуживания
OUT  [20], AL         ; записать в ПКП  код управления
POP  AX               ; восстановить из стека AX
I RET                 ; возврат из INT
```

Начиная с адреса CS:120 набрать текст:

```
NOP
JMP 120
```

13.Замаскировать прерывание системного таймера.
14. Заменить вектор прерываний таймера на новый. Показать преподавателю.
15. Сбросить маску прерываний таймера.
16. В пошаговом режиме, выполнять программу, начиная с адреса CS:100, контролировать содержимое ячейки CS:0030.
17. Замаскировать прерывания системного таймера. Восстановить прежний обработчик, прерываний таймера.
18. Разрешить прерывания таймеру.

3. ОФОРМЛЕНИЕ ОТЧЕТА

Отчет должен содержать:
1. Титульный лист, цель работы.

2. Назначение, правила доступа, коды управляющих команд и содержимое внутренних регистров контроллеров прерываний.
3. Значение маски, использованное для останова системного таймера.
4. Адреса векторов прерывания системного таймера и клавиатуры.
5. Текст начальных команд обработчика прерываний таймера и системного прерывания INT 1Ch .
6. Текст программы обработки прерываний с комментариями.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Одиноков, В. В. Программирование на ассемблере : учебное пособие [для вузов] по специальностям: 090105 - "Комплексное обеспечение информационной безопасности автоматизированных систем", 090102 - "Компьютерная безопасность", 090106 - "Информационная безопасность телекоммуникационных систем" / В. В. Одиноков, В. П. Коцубинский . – Москва : Горячая линия-Телеком , 2011 . – 278, [1] с.
2. Пирогов, В. Ю. Ассемблер для Windows / Владислав Пирогов . – 4-е изд., [перераб. и доп.] . – Санкт-Петербург : БХВ-Петербург , 2012 . – 873 с.
3. Абель,П. Язык ассемблера для IBM PC и программирования /П.Абель. – Москва: Высш. шк., 1992. - 447с.
4. Микро-ЭВМ: пер.с англ./под ред. А. Дирксена. – Москва: Энергоатомиздат, 1982. –328с.
5. Джордейн, Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT /Р. Джордейн. – Москва: Финансы и статистика, 1992. –544 с.

ЭВМ И ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА
Методические указания для выполнения лабораторных работ

Подписано в печать . Формат 60 × 84/16
Бумага писчая. Печать офсетная.
Усл. п. л. 4. Тираж экз. Заказ №

РИО ВоГУ 160000, г. Вологда, ул. С. Орлова, 6.