Задание на лабораторную работу N4 по дисциплине

"Организация ЭВМ и систем"

Тема: Написание собственного прерывания.

1. Краткие сведения.

Прерывание - это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств (нажатие клавиши, обработка сигнала таймера, ввод-вывод строки и т.д.).

Когда возникает прерывание, процессор прекращает выполнение текущей программы (если ее приоритет ниже) и запоминает в стеке вместе с регистром флагов адрес возврата (CS:IP) - места, с которого будет продолжена прерванная программа. Затем в CS:IP загружается адрес программы обработки прерывания и ей передается управление.

Адреса 256 программ обработки прерываний, так называемые векторы прерывания, имеют длину по 4 байта (в первых двух хранится значение IP , во вторых - CS) и хранятся в младших 1024 байтах памяти.

Программа обработки прерывания должна заканчиваться инструкцией IRET (возврат из прерывания), по которой из стека восстанавливается адрес возврата и регистр флагов.

Программа обработки прерывания - это отдельная процедура, имеющая структуру:

SUBR\_INT PROC FAR ; SUBR\_INT – имя, задаваемое пользователем

PUSH AX ; сохранение изменяемых регистров

<действия по обработке прерывания>

MOV AL, 20H ; корректное завершение

OUT 20H, AL ; обработчика

POP AX ; восстановление регистров

IRET

SUBR\_INT ENDP

Две строки перед POP AX и IRET (завершение обработчика) необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное.

*Замечание:* в лабораторной работе действиями по обработке прерывания могут быть обработка и вывод на экран строки текста, задержка вывода сообщений, включение звукового сигнала, снятие отсчета времени и т. п.

Часто возникает необходимость создания собственной или новой подпрограммы обработки прерывания, доступной из любой программы, с целью замены или дополнения функций существующих прерываний, а также использование холостых прерываний, телом которых является IRET. При этом адрес нового (собственного) обработчика прерывания может размещаться по номеру существующего вектора прерывания.

Соответственно программа, использующая новую программу обработки прерывания, при своем завершении должна восстанавливать оригинальные векторы прерываний. Для этого используется функция 35 прерывания 21H, которая сохраняет текущее значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. В этом случае головная программа должна содержать инструкции (для примера заменяется вектор прерывания 1Сh):

; -- в сегменте данных

KEEP\_CS DW 0 ; для хранения сегмента

KEEP\_IP DW 0 ; и смещения вектора прерывания

; -- в начале программы

MOV AH, 35H ; функция получения вектора

MOV AL, 1CH ; номер заменяемого вектора

INT 21H

MOV KEEP\_IP, BX ; запоминание смещения

MOV KEEP\_CS, ES ; и сегмента

Для задания адреса собственного обработчика прерывания с заданным номером в таблицу векторов прерываний используется функция 25H прерывания 21H, которая устанавливает вектор прерывания на указанный адрес нового обработчика (сегмент – в DS, смещение – в DX)

PUSH DS

MOV DX, OFFSET SUBR\_INT ; смещение для процедуры в DX

MOV AX, SEG SUBR\_INT ; сегмент процедуры в АХ

MOV DS, AX ; помещаем в DS

MOV AH, 25H ; функция установки вектора

MOV AL, 1СH ; номер вектора

INT 21H ; меняем прерывание

POP DS

В конце программы нужно восстановить старый вектор прерывания.

CLI

PUSH DS

MOV DX, KEEP\_IP

MOV AX, KEEP\_CS

MOV DS, AX

MOV AH, 25H

MOV AL, 1CH

INT 21H ; восстанавливаем вектор

POP DS

STI

**2. Варианты заданий**

Шифры, определяющие варианты заданий приведены в таблицах.

Задания гр.1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № студ. | Шифр задания | № студ. | Шифр задания | № студ. | Шифр задания | № студ. | Шифр задания |
| 1 | a | 8 | d | 15 | c | 22 | b |
| 2 | b | 9 | a | 16 | d | 23 | c |
| 3 | c | 10 | b | 17 | a | 24 | d |
| 4 | d | 11 | c | 18 | b | 25 | a |
| 5 | a | 12 | d | 19 | c | 26 | b |
| 6 | b | 13 | a | 20 | d | 27 | c |
| 7 | c | 14 | b | 21 | a | 28 | d |

Задания гр.2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № студ. | Шифр задания | № студ. | Шифр задания | № студ. | Шифр задания | № студ. | Шифр задания |
| 1 | d | 8 | a | 15 | b | 22 | c |
| 2 | c | 9 | d | 16 | a | 23 | b |
| 3 | b | 10 | c | 17 | d | 24 | a |
| 4 | a | 11 | b | 18 | c | 25 | d |
| 5 | d | 12 | a | 19 | b | 26 | c |
| 6 | c | 13 | d | 20 | a | 27 | b |
| 7 | b | 14 | c | 21 | d | 28 | a |

Задания гр.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № студ. | Шифр задания | № студ. | Шифр задания | № студ. | Шифр задания | № студ. | Шифр задания |
| 1 | b | 8 | c | 15 | a | 22 | d |
| 2 | d | 9 | b | 16 | c | 23 | a |
| 3 | a | 10 | d | 17 | b | 24 | c |
| 4 | c | 11 | a | 18 | d | 25 | b |
| 5 | b | 12 | c | 19 | a | 26 | d |
| 6 | d | 13 | b | 20 | c | 27 | a |
| 7 | a | 14 | d | 21 | b | 28 | c |

Буква в шифре задает номер и назначение заменяемого вектора прерывания:

a – 1Сh - прерывание от системного таймера - генерируется автоматически операционной системой 18 раз в сек;

b - 60h - прерывание пользователя - должно генерироваться в программе;

c - 23h - прерывание, генерируемое при нажатии клавиш Control+C; иначе программа должна завершаться.

d - 16h - прерывание от клавиатуры (по заданному  
скан-коду клавиши делать действия, указанные ниже).

Цифра определяет действия, реализуемые программой обработки прерываний:

1. Исключение латинских букв и цифр, введенных во входной строке при формировании выходной строки.

2. Формирование выходной строки из номеров позиций во входной строке, на которых расположен символ "D".

3. Преобразование всех заглавных латинских букв входной строки в строчные, а восьмеричных цифр в инверсные, остальные символы входной строки передаются в выходную строку непосредственно.

4. Исключение заглавных русских букв и восьмеричных цифр, введенных во входной строке, при формировании выходной строки.

5. Инвертирование введенных во входной строке цифр в десятичной системе счисления (СС) и преобразование строчных русских букв в заглавные, остальные символы входной строки передаются в выходную строку непосредственно.

6. Выдача звукового сигнала c заданной высотой и длительностью звучания

(управление динамиком через таймер).

7. Преобразование введенных во входной строке десятичных цифр в восьмеричную СС, остальные символы входной строки передаются в выходную строку непосредственно.

8. Формирование номера введенной латинской буквы по алфавиту и номера позиции ее первого вхождения во входной строке и выдача их на экран.

9. Выполнить вывод на экран заданного количества (3-5) одинаковых сообщений, задержка между которыми возрастает в 2 раза, начиная от 1 сек.

10. Выдача двух звуковых сигналов c заданной высотой и длительностью звучания 2 и 4 сек (управление динамиком через таймер).

11. Преобразование всех строчных латинских букв входной строки в заглавные, а десятичных цифр в инверсные, остальные символы входной строки передаются в выходную строку непосредственно.

12. Исключение заглавных латинских букв и десятичных цифр, введенных во входной строке, при формировании выходной строки.

13. Инвертирование введенных во входной строке цифр в восьмеричной СС

и преобразование заглавных русских букв в строчные, остальные символы входной строки передаются в выходную строку непосредственно.

14. Преобразование введенных во входной строке шестнадцатиричных цифр в десятичную СС, остальные символы входной строки передаются в выходную строку непосредственно. Полученные десятичные числа разделять пробелами.

15. Формирование номера введенной русской буквы по алфавиту и номера позиции ее первого вхождения во входной строке и выдача их на экран.

16. Исключение заглавных русских букв и восьмеричных цифр, введенных во входной строке, при формировании выходной строки.

17. Заменить введенные во входной строке латинские буквы на десятичные числа, соответствующие их номеру по алфавиту, остальные символы входной строки передать в выходную строку непосредственно.

18. Выполнить чтение и вывод на экран отсчета системных часов (в тиках, где 1 тик = 55 мсек). С учетом указанного в прерывании 1А числа тиков в час, мин, сек результат представить в виде чч:мм:сс .

19. Исключение русских букв и цифр, введенных во входной строке, при формировании выходной строки.

20. Выдача двух звуковых сигналов c заданной длительностью звучания и высотами звука, отличающимися в 2 раза (управление динамиком через таймер).

21. Преобразование всех заглавных латинских букв входной строки в

строчные, а десятичных цифр в инверсные, остальные символы входной строки передаются в выходную строку непосредственно.

22. Исключение строчных русских букв и десятичных цифр, введенных во входной строке, при формировании выходной строки.

23. Преобразование всех строчных латинских букв входной строки в

заглавные, а шестнадцатиричных цифр в инверсные, остальные символы входной строки передаются в выходную строку непосредственно.

24. Инвертирование введенных во входной строке цифр в десятичной СС и преобразование заглавных русских букв в строчные, остальные символы входной строки передаются в выходную строку непосредственно.

25. Выполнить чтение и вывод на экран отсчета часов реального времени из памяти CMOS (в формате BCD) – использовать прерывании 1А.

26. Инвертирование введенных во входной строке цифр в шестнадцатиричной системе счисления (СС) и преобразование строчных русских букв в заглавные, остальные символы входной строки передаются в выходную строку непосредственно.

27. Исключение строчных латинских букв и восьмеричных цифр, введенных во входной строке, при формировании выходной строки.

28. Выходная строка формируется из инвертированных номеров позиций, на которых во входной строке расположен символ "Ю".

***Замечание:*** для исключения возможного взаимного влияния системных и

пользовательских прерываний рекомендуется отвести в программе под стек не менее 1К байт.

**3. Краткие сведения о командах обработки строк.**

Для обработки строковых данных ассемблер имеет пять групп команд обработки строк:

— MOVS — переслать один байт или одно слово из одной области памяти в другую;

— LODS — загрузить из памяти один байт в регистр AL или одно слово в регистр AX;

— STOS — записать содержимое регистра AL или AX в память;

— CMPS — сравнить содержимое двух областей памяти, размером в один байт или в одно слово;

— SCAS — сравнить содержимое регистра AL или AX с содержимым памяти.

Каждая команда имеет модификации, указывающие размер операнда: байт (B), слово (W), двойное слово (D). Например: MOVSB, MOVSW, MOVSD.

Все команды работы со строками считают, что строка-источник находится по адресу DS:SI, а строка-приемник – по адресу ES:DI. Регистры SI и DI содержат относительные адреса, указывающие на необходимые области памяти (для их загрузки можно использовать команду LEA). Команды MOVS, STOS, CMPS и SCAS требуют инициализации регистра ES (обычно адресом, аналогичным адресу в регистре DS).

Все строковые команды за один раз выполняют операцию только с одним элементом строки (байтом, словом или двойным словом). Для выполнения операций над всеми элементами строки необходимо задать один из префиксов повторения операций: REP (Повторять), REPE (Повторять, пока равно), REPZ (Повторять, пока ноль), REPNE (Повторять, пока не равно), REPNZ (Повторять, пока не ноль). Префиксы используют регистр СХ в качестве счетчика повторений, уменьшая его при каждом выполнении на 1.

В следующем примере выполняется пересылка 20 байт из STRING1 в STRING2. Пусть оба регистра DS и ES уже инициализированы адресом сегмента данных:

STRING1 DB 20 DUP('\*')

STRING2 DB 20 DUP(' ')

CLD ;Сброс флага DF для направления передачи слева направо

MOV CX,20 ;Счётчик на 20 байт

LEA DI,STRING2 ;Адрес области "куда"

LEA SI,STRING1 ;Адрес области "откуда"

REP MOVSB ;Переслать данные

В заключение приведем таблицы кодирования цифровых и буквенных символов.

1. Кодовая таблица 1251 (Windows)

|  |  |  |
| --- | --- | --- |
| Символ | Десятичный код | 16-ричный код |
| 0 | 48 | 30h |
| … | … | … |
| 9 | 57 | 39h |
| … | … | … |
| A | 65 | 41h |
| … | … | … |
| Z | 90 | 5Ah |
| … | … | … |
| a | 97 | 61h |
| … | … | … |
| z | 122 | 7Ah |
| А | 192 | C0h |
| … | … | … |
| Я | 223 | DFh |
| а | 224 | E0h |
| … | … | … |
| я | 255 | FFh |
| Ё | 168 | A8h |
| ё | 184 | B8h |

1. Кодовая таблица 866 (MS DOS)

|  |  |  |
| --- | --- | --- |
| Символ | Десятичный код | 16-ричный код |
| А | 128 | 80h |
| … | … | … |
| Я | 159 | 9Fh |
| а | 160 | A0h |
| … | … | … |
| п | 175 | AFh |
| р | 224 | E0h |
| … | … | … |
| я | 239 | EFh |
| Ё | 240 | F0h |
| ё | 241 | F1h |
| Коды цифр, а также строчных и заглавных латинских букв совпадают с аналогичными кодами таблицы 1251 | | |

**4. Необходимые вспомогательные функции.**

***4.1. Ввод строки символов***

Ввод строки символов реализуется с помощью функции 0Ah прерывания 21h, которая позволяет вводить строку длиной до 254 символов, выдавая эхо на терминал. Эта функция продолжает ввод символов до тех пор, пока не нажата клавиша "Enter". DS:DX указывает на строку, куда помещаются вводимые символы.

|  |  |  |
| --- | --- | --- |
| Вход: | АН | 0Ah |
| DS:DX | адрес буфера для ввода строки |
| Выход: | = | буфер содержит введенную строку |

Первый байт буфера должен содержать максимальное число символов для ввода (от 1 до 254). Второй байт буфера будет содержать длину реально введенной строки без учета последнего символа CR (клавиша Enter). С третьего байта записывается вводимая строка, включая последний символ CR (0Dh).

Таким образом, для получения 50 - символьной строки надо отвести под буфер минимум 53 байта памяти:

– в первый байт памяти записать число ASCII 51;

– после ввода 50 символов второй байт дескриптора будет содержать ASCII 50;

– 53-й байт отведённой памяти будет содержать ASCII 13 (0Dh).

**Пример 1.**

; - - - в сегменте данных

STRING DB 53 DUP(?) ; область для 50 символов

; - - - получение строки с клавиатуры

LEA DX, STRING ; DS:DX - адрес строки

MOV BX, DX

MOV AL,51 ; установка длины строки (+1 для CR)

MOV [BX], AL ; посылаем в первый байт буфера длину строки

MOV AH, 0AH ; номер функции

INT 21H

; - - - получаем в AH реальную длину строки

MOV AH, [BX]+1 ; теперь реальная длина в AH

**Пример 2.**

; Программа считывает строку символов и выводит ее на экран.

;------------------------------------------------------------------------

model small

.stack 100h

.data

crlf DB 0Dh, 0Ah, '$' ; символы перехода на начало новой строки

buf DB 6 ; выделяем буфер для ПЯТИ вводимых символов

.code

start:

mov ax, @data

mov ds, ax

MOV DX, offset buf ; DX = адрес буфера

MOV AH, 0Ah ; AH = номер функции 0Аh

INT 21h ;вызов функции 0Ah

;Если введенную с клавиатуры строку необходимо вывести на экран, то в

;конец строки нужно записать символ $. Для этого нужно получить адрес

;буфера, узнать длину реально введенной строки и прибавить к полученному

;значению 2, т.к. первые 2 байта содержат служебную информацию. Это

;значение будет адресом байта, который следует за последним символом

;строки. В этот адрес и записывается символ $

MOV DХ, offset buf ; DХ = адрес буфера

MOV BX, 0h ; ВХ = 0

MOV BL, [DХ+1] ; BL = длинa строки

MOV BYTE PTR [DХ+BX+2], '$' ; DХ+BX+2 - адрес, который следует

;за последним символом строки

MOV DX, offset buf ; DX = адрес буфера

ADD DX, 2h ; прибавляем к DX число 2, т.к. первые

; 2 байта не содержат символов строки

MOV AH, 9h ; вызываем функцию вывода строки

INT 21h

;------------------------------------------------------------------------

mov AX,4C00h

int 21h

end start

***4.2. Управление динамиком и генерация звука.***

Внутри корпуса IBM PC есть динамик и программа может управлять звуками, генерируемыми этим динамиком. Для этого нужно управлять некоторыми выходными битами микросхемы 8255 (параллельный программируемый интерфейс) и генератора частоты звука в микросхеме 8253 (системный таймер).

Программируемый периферийный интерфейс микросхемы 8255 (PPI) имеет три входных или выходных порта. IBM PC инициализирует микросхему 8255 так, чтобы получилось два входных  порта - 60H и 62H - и один порт вывода, 61H.

Входной порт 60H в первую очередь вводит значения с клавиатуры, а кроме того порт 60h служит буфером между микропроцессором и устройством ввода-вывода; он передает данные микропроцессору только тогда, когда последний запрашивает их командой IN. Все остальное время вводной порт задерживает данные и не допускает, чтобы они повлияли на работу микропроцессора.

Порт 61H - это порт вывода микросхемы 8255 в машинах фирмы IBM. Всякий выходной порт временно запоминает данные, выводимые программой. Если бы аппаратура не запоминала данные, они бы пропали примерно в течение микросекунды. Такое запоминание данных позволяет сохранять их значение в порте до тех пор, пока они снова не будут изменены программой. То есть, когда мы выводим значение, меняющее положение диффузора динамика, оно остается неизменными до тех пор, пока его не изменит программа.   
 На рис.1 показано значение битов порта 61H. Эти данные взяты из технического описания. При изучении управления динамиком имеют значение только биты 0 и 1.

  Бит           Значение   
      -----------------------------------------------------   
         0      Порт 2 таймера (управление динамиком)   
         1      Прямое управление динамиком   
         2      Мультиплексный порт 62H   
         3      Управление мотором кассетного магнитофона   
         4      Включение контроля доступа на системной   
                  плате памяти   
         5      Включение контроля доступа в памяти   
                  каналов ввода-вывода   
         6      Временной контроль клавиатуры   
         7      Мультиплексный порт 60H сброса ввода с клавиатуры   
                 
            Рис. 1. Значение битов порта 61H

На рис. 2 показана программа (Microsoft (R) Macro Assembler Version 5.00),

которая управляет динамиком двумя разными способами.

Первый способ, помеченный в листинге меткой DIRECT, непосредственно управляет динамиком. Бит 1 выводного порта 61H подключен к динамику. Всякий раз, когда программа меняет значение этого бита, диффузор динамика двигается либо наружу, либо внутрь. Быстро меняя значение этого бита, программа генерирует звук. Это иллюстрирует первая часть программы на рис. 2, она меняет значение бита 1, порождая высоко-частотный тон. Скорость, с которой программа меняет бит 1, определяет частоту тона. Взяв на себя непосредственное управление динамиком, вы должны прежде всего работать с выводным портом микросхемы 8255 системной   платы.

TITLE Dynamic  
            STACK SEGMENT STACK   
                     DW    64 DUP (?)   
       STACK ENDS   
            CODE  SEGMENT   
                          ASSUME      CS:CODE   
          SOUND PROC    FAR   
                  PUSH  DS          ; Адрес возврата   
              MOV   AX, 0   
              PUSH  AX

  ;-----      Задание режима работы динамика   
                    SUB   CX, CX            ; Счетчик цикла   
                  IN    AL, 61H   
                   AND   AL, 0FEH   
                     OUT   61H, AL     ; Установка разряда 0 порта 61H в 0 для   
                                                           ;  задания прямого управления динамиком

;-----  Прямое управление динамиком

  DIRECT:   
                       OR    AL, 2   
                      OUT   61H, AL     ; Включить динамик   
                   AND   AL, 0FDH   
                      OUT   61H, AL     ; Выключить динамик   
                       LOOP  DIRECT

            ;-----  Управление динамиком через таймер  
                      MOV   AL, 10110110b   
                        OUT   43H, AL     ; Установка режима для 2-го канала   
            MOV BX, frequency; частота в гц(напр.1000)→в делитель BX

MOV AX, 34DDh

MOV DX, 12h ; (DX,AX)=1193181

DIV BX ; AX=(DX,AX)/BX

                         OUT   42H, AL ; Занесение  
                        MOV   AL, AH ; высоты звука  
                         OUT  42H, AL    ; в порт динамика   
                            IN    AL, 61H   
                          MOV  AH, AL   
                          OR    AL, 3   
                          OUT   61H, AL ; Запуск динамика и включение звука

             MOV CX, 000Fh ; Задержка в 1 сек,

MOV DX, 4240h ; во время которой

MOV AH, 86h ; работает

INT 15h ; динамик

                            MOV   AL, AH   
                            OUT   61H, AL     ; Выключение динамика   
                           RET   
              SOUND ENDP   
            CODE  ENDS   
                                  END

Рис. 2. Управление динамиком

Как показано на рис.2, установка бита 0 в положение 0 включает прямое управление динамиком, блокируя механизм генерации звука микросхемой 8253, который используется во второй части программы.   
     Обратите внимание на то, как программа сбрасывает бит 0. Команда OUT включает все 8 бит порта 61H. Способа изменить только бит 0, оставив остальные биты нетронутыми, не существует. Если в программе нужно изменить только бит 0, она должна считать из порта текущее значение других разрядов. К счастью, микросхема 8255 допускает прямое программное чтение выводных портов.   
    Последовательность команд

IN    AL , 61H   
      AND   AL , 0FEH   
      OUT   61H, AL   
 читает текущий код из выводного порта, затем команда AND сбрасывает младший бит, а команда OUT посылает результат в выводной порт. Если бы программа вывела в порт просто число 0, динамик работал бы верно, но клавиатура была бы выключена. Работая с любым портом вывода побитовой настройки, стройте программу так, чтобы она не влияла ни на один из других бит, если только вы не собираететсь изменять и их.   
        Оставшаяся часть первой части программы на рис.2 изменяет значение бита 1 выходного порта. Исходное значение порта 61H находится уже в регистре AL, так что программе не нужно читать его при каждом выполнении цикла. Регистр CX используется таким образом, чтобы выполнить цикл 64K раз. Если при выполнении программы не удается услышать звук, генерируемый программой, попытайтесь вставить несколько добавочных команд NOP в цикл DIRECT. Это снизит частоту тона.   
  
 Вторая часть программы на рис. 2 для генерации тональности использует системный таймер 8253. Микросхема 8253 фирмы Intel содержит три 16-битовых счетчика, которые могут быть использованы в системе для счета или задания временных интервалов. В один из счетчиков программа загружает 16-битовое значение. Содержимое счетчика уменьшается на единицу по каждому импульсу от таймера; частота импульсов, подводимых с таймера ко всем трем каналам, равна 1.19МГц. Это означает, что содержимое счетчика уменьшается на  единицу каждые 0,84 микросекунды. Каждый из трех каналов имеет выход. Командами управления определяют способ, которым микросхема 8253 ведет счет.

Выходы трех каналов счетчика-таймера подключаются к различным узлам системной платы. Канал 0 подключается к контроллеру прерываний 8259 и используется для порождения прерывания подсчета времени суток. Канал 1 соединен с контроллером прямого доступа к памяти (ПДП или DMA) 8237, и использовать этот канал схемы 8253 нельзя, так как смена кода в этом счетчике с большой вероятностью уничтожит вашу программу и все другие данные в памяти системы.

Наконец, канал 2 подключен к динамику для генерации звука. Для установки канала 2 таймера программа должна послать код 0B6H в порт 43H - управляющий порт микросхемы таймера 8253, назначение бит которого приведены на рис.3.

Структура бит порта по адресу 43h

|  |  |  |  |
| --- | --- | --- | --- |
| Канал  7 - 6 | Операция  5 - 4 | Режим  3-1 | 0 |

0 - бит указывает какая система счисления используется - двоичная (0) или BCD (1) (двоично-кодированная десятичная).

Биты 1 - 3 задают режим работы таймера:

* 000 - запрос прерывания IRQ0 при достижении нуля;
* 001 - мультивибратор;
* 010 - генератор импульсов;
* 011 - операции считывания информации канала 0 и подсчета поступивших импульсов; также генератор прямоугольных импульсов **(основной режим);**
* 100 - программно запускаемый одновибратор;
* 101 - аппаратно запускаемый одновибратор;

Биты 4 - 5 определяют возможные операции:

* 00 - «защелкивание» значения таймера – фиксация текущего значения счетчика, все остальные биты, кроме выбора номера канала (включая биты 0 – 3), игнорируются, и выполняется запись текущего значения счетчика во внутренний регистр, содержимое которого затем может быть прочитано в аккумулятор.
* 01 - чтение или запись старшего байта счетчика;
* 10 - чтение или запись младшего байта счетчика;
* 11 - чтение или запись последовательно младшего и старшего байтов.

Биты 6 -7 - номер канала, который будет программироваться:

* 00 - канал 0 ;
* 01 - канал 1 ;
* 02 - канал 2 .

Загрузкой кода 0B6H в порт 43H канал 2 таймера-счетчика настраивается на работу в качестве делителя частоты. Таймер делит исходную частоту - в данном случае 1.19МГц - на 16-битовое число, которое программа загружает в регистр канала 2. Регистр канала 2 расположен по адресу порта 42H. Программа в примере загружает в регистр канала число 1000. Это означает, что на выходе вы услышите частоту 1190Гц.

На самом деле, вы услышите основную частоту 1190 Гц плюс обертоны, вызванные прямоугольной формой сигнала таймера.

Заметим, что число 1000 - 16-битовое, в то время как порт 42H - 8-битовый. Команда установки режима работы, которую мы послали в порт 43H, сообщила микросхеме 8253, что в нее будет выводиться 16-битовое число в виде двух 8-битовых. Сначала посылается младший значащий байт, а за ним следует старший. Такая двухшаговая процедура загружает в канальный регистр требуемое значение.

Далее программа должна дать управляющему порту 61H такую установку, чтобы он пропускал сигнал на динамик. Для этого программа устанавливает равными 1 биты 0 и 1 управляющего порта.   
    Заметим, что программа в начале сохраняет первоначальное значение кода из порта 61Н и восстанавливает его в конце. Это отключает динамик по окончании звука. Если этот способ недостаточен, - например, если программа генерирует звук тогда, когда не совсем ясно, был ли выключен динамик - можно выключить его, сбросив в нуль бит 1 порта 61H.