Contents

Com-port

CRC

Порты

Программы

Адреса портов

```
com1 equ 3f8h
com2 equ 2f8h
com3 equ 3e8h
com4 equ 2e8h
```

Установка DCD

```
mov dx, 3f8h + 6
in al, dx
or al, 10000000b
out dx, al
```

Установка DTR

```
mov dx, 3f8h + 4
in al, dx
or al, 00000001b
out dx, al
```

Установка RTS

```
mov dx, base+4
mov al, 00000010b
out dx, al
```

Ждать DCD

```
mov dx, 3f8h + 6
check:
in al, dx
test al, 00100000b
jz check
```

Ждать DTR

```
mov dx, 3f8h + 4
check:
in al, dx
test al, 00000001b; DTR
jz check
```

Инициализация порта

```
;control mode
mov dx, 3f8h = + 3
mov al, 10000000b
out dx, al
;sending speed
mov dx, 3f8h
mov al, low_speed
out dx, al
mov dx, 3f8h + 1
mov al, high_speed
out dx, al
;sending control word
mov dx, 3f8h + 3
mov al, 0000111b
out dx, al
```

Получить 1 байт

```
receive_byte macro adr
local check, not_break, not_error, finish
check:
       mov dx, port_adr + 5
       in al, dx
       test al, 00010000b; break
       jz not_break
       print message_break
       jmp finish
not_break:
    test al, 00000100b
       jz not_error
       print message_parity
not_error:
       test al, 00000001b
       jz check
       mov dx, port_adr
       in al, dx
       mov [adr], al
finish:
endm
```

Получать байты, пока не будет break

```
receive_data macro
local check, not_break, not_error, finish
       lea di, data
check:
       mov dx, port_adr + 5
       in al, dx
       test al, 00010000b; break
       jz not_break
       print message_break
       jmp finish
not_break:
    test al, 00000100b
       jz not_error
       print message_parity
not_error:
       test al, 00000001b
       jz check
       mov dx, port_adr
       in al, dx
       cmp al, manual_break
       jz finish
       mov ds:[di], al
       inc di
       inc data_count
       cmp data_count, 64
       jz finish
       jmp check
finish:
endm
```

Отправить 1 байт

```
send_byte macro adr
local send
send:
       mov dx, port_adr
       mov al, adr
       out dx, al
       mov dx, port_adr + 5
       in al, dx
       test al, 00100000b
       jz send
endm
```

Отправить много байт

```
send_data macro
local send
       mov cx, 64
       lea di, data
send:
       mov dx, port_adr
       mov al, ds:[di]
```

```
       out dx, al
success:
       mov dx, port_adr + 5
       in al, dx
        bt ax, 5
        jnc success
       inc di
       loop send
endm
```

Выдача break

```
mov dx, port_adr + 3
in al, dx
or al, 01000000b
out dx, al
```

#18. Напишите два варианта настройки порта COM2 на скорость 9600 б/с, 1 стоповый бит, 5-разрядный формат, проверку на нечётность.

```
Port  equ   2F8h
....
     mov    al, 80h
     mov    dx, Port+3   ;
Управляющий порт
     out    dx, al       ; Режим
настройки
     mov    al, 0Ch
     mov    dx, Port
     out    dx, al       ;
Передача кода скорости
     mov    dx, Port+3   ;
Управляющий порт
     mov    al, 1000b    ;
Задание скорости, чётности и
т.п.
.......................
     xor    ah, ah       ;
Инициализация COM2
     mov    dx, 2
     mov    al, 111 . 11 .  0
.  00
               Spd   O/E   StB
5Bt
     int    14h
```

#1. Напишите фрагмент программы, которая ожидает сигнала RI и по его приходу выдаёт сигналы RTS и DTR. Все прерывания замаскированы.

```
     mov    dx, 3F8h + 6
LL1: in     al, dx
     test   al, 40h
     jz     LL1
     mov    al, 3
     mov    dx, 3F8h + 4
     out    dx, al
```

#17. Используя BIOS, напишите программу инициализации порта COM2 для передачи текста повести с максимальной скоростью.

```
    Mov    AH, 00h
    Mov    DX, 2
    Mov    AL, 11111001b      ; 6 бит на символ
!
    Int    14h
```

```
;напишите фрагмент программы,
;который передает по порту COM1
;байт 88h в течении 12-ти секунд
com1  equ 3F8h
com2  equ 2F8h

base  equ com1

write base+4,88h
;задержка
mov al,0
out 70h,al
in al,71h
add al,12h
daa
cmp al,59h
jbe L
sub al,60h
das
L:
mov bl,al
M:
in al,71h
cmp al,bl
jnz M

write base+4,0

;написать фрагмент программы, которая
;автоматически настраивает COM1-порт на
скорость,
;с которой на него поступает поток байтов

readbyte macro symbol
  local @@loc
  push ax
  push dx

@@loc:
  read mus,base+5
  mov al,mus
  test al,1h
  jz @@loc
```

```
  read symbol,base

  pop dx
  pop ax
endm

DATA SEGMENT use16
hi_sp db ?
lo_sp db ?
mus db ?

DATA ENDS
```

```
read al,base
read al,base
read al,base

readbyte hi_sp    ;speed
readbyte lo_sp    ;speed

write base+3,80h   ;setup port
write base,lo_sp   ;low byte of speed
write base+1,hi_sp  ;hi byte of speed
```

Часы реального времени

int 1ah

```
INT 1Ah,  00h (0)        Read System-Timer Time Counter                  all
    Reports the current time of day, and whether 24 hours has passed since
    1) the last power-on, 2) the last system reset, or 3) the last system-
    timer time read or set.
       On entry:      AH         00h
       Returns:       CX         High-order part of clock count
                      DX         Low-order part of clock count
                      AL         0 if 24 hours has not passed; else 1
   ---------------------------------------------------------------------------
        Notes:         The following formulas convert the clock count to
                      the time of day:
                                 Hour      = Clock / 65543 (1007h)
                                 Remainder = Clock MOD 65543
                                 Minutes   = Remainder / 1092 (444h)
                                 Remainder = Remainder MOD 1092
                                 Second    = Remainder / 18.21
                                 Remainder = Remainder MOD 18.21
                                 Hundredths = CINT(Remainder * 100)
                      The "system timer" (as distinguished from the real-
                      time clock) is the timer that's set when the system
                      is started. This time is temporary, lasting only as
                      long as the system is turned on.
                      The clock count may also be read as a 4-byte integer
                      at memory location 0:046C. This 4-byte value is
                      equal to the 4-byte integer in CX:DX after Service
                      00h has been called.

                      After the call, the flag (at 0:0470h) stating
                      whether 24 hours has passed or not, is cleared.

                      When TIME is typed at the command line, DOS gets the
                      time by means of this service.

                      Counts occur at the rate of 18.2 per second.
```

Программы

Задержка на 10 секунд

```
Mov ax, 40h
Mov es, ax
Mov di, 6ch
Mov eax, es:di
```

```
Label:
Mov ebx, ed:di
Sub ebx, eax
Sub ebx, 182
Jnz Label
```

Считать количество секунд с 00:00:00

```
get_seconds macro mem_cell
       pusha
       mov ah, 02h
       int 1ah
       mov bl, dh
       mov al, ch
       xor ah, ah
       aam 16
       aad 10
       mul seconds_in_hour
       mov (word ptr mem_cell)+2, dx
       mov word ptr mem_cell, ax
       mov al, cl
       xor ah, ah
       aam 16
       aad 10
       mul seconds_in_minute
       mov edx, mem_cell
       add edx, eax
       mov al, bl
       xor ah, ah
       aam 16
       aad 10
       add edx, eax
       mov mem_cell, edx
       mov ecx, mem_cell
       popa
endm
```

Считать тики прерыванием

```
get_ticks macro mem_cell
       pusha
       mov ah, 00h
       int 1ah
       mov (word ptr mem_cell)+2, cx
       mov word ptr mem_cell, dx
       popa
endm
```

Считать тики напрямую

```
get_ticks2 macro mem_cell
       pusha
       mov ax, 40h
       mov es, ax
       mov di, 6ch
       mov eax, es:di
       mov mem_cell, eax
       popa
endm
```

#13. Напишите фрагмент программы установки будильника на 12:20:45 микросхемы часов реального времени с использованием BIOS.

```
     mov    ah, 7
     int    1Ah
     mov    ah, 6
     mov    ch, 12h        ;
Время заносится
     mov    cl, 20h        ; в
BCD
     mov    dl, 45h        ;
     int    1Ah
```

#03. Написать программу выдачи с выхода 2-го канала м/с таймера синхроимпульсов скважностью 2 частотой 10КГц, если на вход CLK2 поступают импульсы частотой 1.9МГц.

```
     1.9МГц/10КГц = 190 = 0BEh
– константа
     mov     al, 10110110b ;
     out     43h,al
     mov     al, 0BEh
     out     42h, al
     xor     al, al
     out     42h, al
```

#64. Написать программу, формирующую импульс на выходе 2-го канала микросхемы таймера через 8 мс, если на вход CLK2 поступают импульсы частотой 1,9МГц.

$$Const = f \cdot t = 15200$$

```
     Const    dw   15200

     mov     al, 10110000b ;
режим 0,
     out     43h,al
     mov     ax, Const
     out     42h, al
     mov     al, ah
     out     42h, al
```

FAT

Загрузочный сектор логического диска

Смещение в секторе	Размер	Содержание
00	3	Инструкция перехода на программу загрузки
03	8	Аббревиатура операционной системы
0Bh	2	Число байтов в секторе (всегда 512)
0Dh	1	Число секторов в кластере
0Eh	2	Размер системной области (включая этот сектор)
10h	1	Число таблиц FAT (чаще всего 2)
11h	2	Число описателей файлов в корневом каталоге (в FAT32 - 0)
13h	2	Общее число секторов на диске (если 0, то размер - в поле со смещением 20h)
15h	1	Тип устройства
16h	2	Размер одной FAT в секторах (0 в FAT32)
18h	2	Число секторов на дорожке
1Ah	2	Число головок
1Ch	4	Абсолютный номер этого сектора
20h	4	Размер диска в секторах

Загрузочный сектор FAT12

Смещение	Размер	Содержимое
24h	1	Номер дисковода для функций BIOS
25h	1	Зарезервировано
26h	1	Сигнатура - 29h
27h	4	Дата/время создания диска
2Bh	11	Метка диска - текстовая строка
36h	8	Аббревиатура файловой системы

Описатель файла

Смещение	Размер	Содержимое
00	8	Имя файла
08	3	Расширение файла
0Bh	1	Атрибуты файла
0Ch	1	Зарезервировано
0Dh	1	Сотые доли секунды создания файла
0Eh	2	Время создания файла
10h	2	Дата создания файла
12h	2	Дата последнего обращения к файлу
14h	2	Старшее слово первого кластера файла
16h	2	Время последней записи в файл
18h	2	Дата последней записи в файл
1Ah	2	Младшее слово первого кластера файла
1Ch	4	Размер файла в байтах

Программы

FAT

#1. На жёстком диске организованно 10 логических дисков. Сколько передвижений головки надо выполнить, чтобы в наихудшем случае добраться до файла, если последний локализован в корневом каталоге?

За 4-передвижения считывается последний диск, 5 – доступ к файлу

#24. Дорожка жёсткого диска содержит 13 секторов, фактор чередования равен 3-м. Укажите порядок расположения секторов на дороже.

1	2	3	4	5	6	7	8	9	10	11	12	13
1	10	6	2	11	7	3	12	8	4	13	9	5

#32. Скорость вращения жёсткого диска составляет 10 обр./с . Файл занимает 8 смежных секторов, расположенных на 17-и секторной дорожке с фактором чередования 3. Определить время считывания файла

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	**7**		**2**	**8**		**3**			**4**			**5**			**6**	

1	7	13	2	8	14	3	9	15	4	10	16	5	11	17	6	12

$\Rightarrow$ необходимо сделать 2 оборота. Время считывания файла $t = \dfrac{2\ [\text{об}]}{10\ [\text{об}/c]} = 0{,}2\ [\text{с}]$

#31. Напишите фрагмент программы, которая определяет число файлов, хранящихся в корневом каталоге на гибком диске (1.44M). Считать, что их меньше 32.

```
Data      db   1024 dup (?)
FileCount db  0
       mov    ah, 2     ; Чтение
       mov    dl, 0     ; Дисковод A
       mov    ch, 0     ; Цилиндр
       mov    cl, 2     ; Сектор 19 (абс) - начало корневого каталога
       mov    al, 2     ; Читать два сектора
       lea    bx, Data
       int    13h
; Анализ каталога
       mov    cx, 32
       lea    bx, Data
LL1:
       mov    al, [bx]
       or     al, al
       jz     End
       add    bx, 32
       cmp    al, 0E5h
       je     LL1
       inc    FileCount
       loop   LL1
End:
```

#16. Из корневого каталога считан описатель файла. Определить время создания файла и номер начального кластера.

```
     49 52 5F 31 20 20 20 20 45 58 45 20 00 00 00 00
     00 00 00 00 00 00 6B 55 D7 22 0F 00 0C 0A 00 00
     ................................................
     DirEntry   struc
```

```
        FileName db 8 dup(?)
        Ext      db 3 dup(?)
        Attr     db ?
        Reserved db 10 dup(?)
        Time     dw ?
        Date     dw ?
        Cluster  dw ?
        FileSize dd ?
      DirEntry    ends
```

I	**R**	**_**	**1**					E	X	E					
49	52	5F	31	20	20	20	20	45	58	45	20	00	00	00	00
00	00	00	00	00	00	6B	55	D7	22	0F	00	0C	0A	00	00
						Time		Date		Cls		Size			

```
Time Pack 556Bh,    Date Pack 22D7h
     Время      01010.101011.01011 = 556Bh    (5.6.5)
                  Час   Мин   Сек = 01011•2
     Время        10  : 43  :  22

     Дата       0010001.0110.10111   = 22D7h  (7.4.5)
          1980 + Год    Мсц  День
     Дата         1997   6     23
```

```
      Начальный кластер 0Fh
      Резмер   0Ah
```

#17. Из корневого каталога считано 32 байта - описателя файла. Определить дату создания файла и его обьем в байтах: 49 52 5F 31 20 20 20 20 41 58 45 21 00 00 00 00 00 00 00 00 00 00 DE 83 3D 16 15 00 00 87 00 00.

Считать корневой каталог

```
read_root macro
       pusha
       mov ah, 2
       mov al, 14
       mov cx, 2h
       mov dx, 100h
       lea bx, root
       int 13h
       popa
endm
```

Перевести адрес сектора из физического в логический

```
convert_sector macro
local skip, fin
       pusha
       mov ax, sector_number
       xor dx, dx
       mov cx, 18
       shl cx, 1
       div cx
       mov track, al

       mov ax, dx
       xor dx, dx
       push cx
       mov cx, 18
       div cx
       pop cx
       mov head, al
       mov sec, dl
;corrections
       mov ax, sector_number
       xor dx, dx
       mov cx, 36
       div cx
       cmp dl, 0
       jnz skip
       inc head
       dec track
       add sec, 12h
       jmp fin
skip:
       mov ax, sector_number
       xor dx, dx
       mov cx, 18
       div cx
       cmp dl, 0
       jnz fin
       dec head
       add sec, 12h
fin:
       popa
endm
```

Читать сектор

```
read_sector macro
local nth
       mov ax, sector_number
       cmp ax, 0
       jz nth
       convert_sector
       mov ah, 2
       mov dl, 0; A
       mov dh, head
       mov ch, track
       mov cl, sec
       mov al, 1
       lea bx, sector
       int 13h
nth:
endm
```

Писать сектор

```
write_sector macro
local nth
       mov ax, sector_number
       cmp ax, 0
       jz nth
       convert_sector
       mov ah, 3
       mov dl, 0; A
       mov dh, head
       mov ch, track
       mov cl, sec
       mov al, 1
       lea bx, sector
       int 13h
nth:
endm
```

Отформатировать дискету

```
        ;RESET
        mov ax, 0000h
        mov dl, [disk_drive]
        int 13h

        ;PREFORMAT
        mov ax, 1800h
        mov ch, 80
        mov cl, 18
        mov dl, [disk_drive]
        int 13h

        ;RESET
        mov ax, 0000h
        mov dl, [disk_drive]
        int 13h
```

Отформатировать дорожку

```
        mov ah, 05h
        mov al, 18
        mov cl, 00
        mov bx, offset buffer ;db
512 dup (?)
        mov ch, [track]
        mov dh, [head]
        mov dl, [disk_drive]
        int 13h
```

Запись на дорожку

```
        mov ah, 03h
      mov bx, offset write_buf
        mov al, 18           ; 18
sectors
        mov dl, [disk_drive]    ;
floppy disk
        mov dh, [head]          ;
zero head (up)
        mov ch, [track]         ;
13th track
        mov cl, 1               ;
from 1 sector
        int 13h
```

Чтение с дорожки

```
        mov ah, 02h             ;
2nd function
      mov bx, offset read_buf
        mov al, 18             ; 18
sectors
        mov dl, [disk_drive]    ;
floppy disk
        mov dh, [head]          ;
zero head (up)
        mov ch, [track]         ;
13th track
        mov cl, 1               ;
from 1 sector
        int 13h
```

#31. Напишите фрагмент программы, которая определяет число файлов, хранящихся в корневом каталоге на гибком диске (1.44М). Считать, что их меньше 32.

```
Data      db   1024 dup (?)
FileCount db  0
       mov    ah, 2     ; Чтение
       mov    dl, 0     ;
Дисковод A
       mov    ch, 0     ;
Цилиндр
       mov    cl, 2     ; Сектор
19 (абс) – начало корневого
каталога
       mov    al, 2     ; Читать
два сектора
       lea    bx, Data
       int    13h
; Анализ каталога
       mov    cx, 32
       lea    bx, Data
LL1:
       mov    al, [bx]
       or     al, al
       jz     End
       add    bx, 32
       cmp    al, 0E5h
       je     LL1
       inc    FileCount
       loop   LL1
End:
```

```
;считать первый сектор последнего файла корневого каталога
.386
 model small
show  macro f
      push ax
      push dx
      mov dx,offset f
      mov ah,9
      int 21h
      pop dx
      pop ax
endm
data segment use16;
 BUFFER  DB   512 DUP(?)   ;создаем буфер
     cyl db 0 ; цилиндр
     head db 0 ; головка
     sec db 0  ; сектор
data  ends
assume  cs:code ,ds:data, ss:stack
code segment use16
main:
     mov ax,data ;
     mov ds,ax;
     ;---читаем сектора
     MOV  AX,SEG BUFFER       ;
     MOV  ES,AX
     MOV  BX,OFFSET BUFFER    ;
     mov ah,2
     mov al,1
     mov cx,2h
     mov dx,100h
     INT  13H             ;
      mov di, offset buffer
      scan:
      add di,32
      cmp byte ptr es:[di-8],0
      je eom
      cmp byte ptr es:[di-8],0E5h
      je scan
      mov bx,di
      jmp scan
      eom:
      add bx,01Ah
      mov ax,[bx]
      add ax,020h
      pusha
      mov bx,offset cyl
      mov cl, 36;секторов в цилиндре
      div cl
      mov BYTE PTR[bx], al
      mov al, ah
      xor ah, ah
      lea bx, head
      mov cl, 18 ; секторов на дорожке
      div cl
      mov BYTE PTR[bx], al
      lea bx,sec
      mov BYTE PTR[bx], ah
      mov ah, 2 ; читаем сектор
    mov al, 1 ;к-во секторов
    lea bx, cyl
    mov ch, BYTE PTR[bx] ;номер дорожки
    lea bx, sec
    mov cl, BYTE PTR[bx]; сектор, с кот. нач. чит.
    lea bx, head
    mov dh, BYTE PTR[bx]; номер головки
    mov dl, 0                ;гибкий диск
    MOV  AX,SEG BUFFER       ;
    MOV  ES,AX
    lea bx, buffer; разм. прогр. данных
    int 13h
    exit:   mov ax,4C00h
    Int 21h
```

```
 code ends
 stack 200h
 end main
```

#4. Написать фрагмент программы, который выполняет сброс контроллера, включает двигатель 1-го НГМД и, если есть второй НГМД, то его также включает.

```
     cli
     mov   al, 0001100b            ; Сброс контроллера
     mov   dx, 3F2h
     out   dx, al

     mov   al, 00111100b           ; Включение моторов
     out   dx, al
     sti
```

#48. Заполните таблицу данных, которые нужно передать при выполнении командной фазы форматирования дорожки, на которой расположен 155-й сектор. Форматировать на 720Кбайт та, чтобы чтение-запись выполнялись бы возможно медленнее.

```
 Фаза команды:
     01001101   (4Dh)    – команда форматирования
     00000100            – 155 сектор находится на Head  1
 N      2                – код длины сектора 512
 SC     9                – к-во секторов
 GPL    1Bh              – длина GPL   (для 1.44M – 06Ch)
 D      0FEh             – Заполнитель
```

#37. Написать фрагмент программы, который выполняет фазу чтения результата операции записи на 176-й сектор гибкого диска, отформатированного на 1.44M. Если запись прошла успешно, установить флаг C равным 0, иначе установить этот флаг в 1.

; Чтение al из порта с проверкой готовности

```
InAL   macro
       local   m1
       mov     dx,3F4h
m1:
       in      al,dx
       and     al,11000000b
       cmp     al,11000000b
       jne     m1
       inc     dx
       in      al,dx
endm
...............
       InAL                          ; Чтение ST0
       and     al,11000000b          ; Маска[Нормальное завершение]
       mov     ah,al
       InAL                          ; Чтение ST1
       and     al,10110110b          ;
       or      ax,ax
       jnz     Error
       clc
       jmp     Next
Error:
       stc
Next:
       InAL                          ; Чтение ST2
```

```
       InAL                        ; Чтение C
       InAL                        ; Чтение H
       InAL                        ; Чтение R
       InAL                        ; Чтение N
```

#60. Написать фрагмент программы, который передвигает головки на 2 дорожки (в сторону увеличения номера) по сравнению с их текущей позицией.

```
; Вывод al в порт с проверкой готовности
OutAL  macro   Command
       local   m1
       mov     dx,3F4h
m1:
       in      al,dx
       and     al,11000000b
       cmp     al,10000000b
       jne     m1
       inc     dx
       mov     al,Command
       out     dx,al
endm
.........................
       OutAL   11001111b          ; Относительное позиционирование
головки
       OutAL   0
       OutAL   2
```

#1. Какой процент информационной ёмкости используется непосредственно при записи полезной информации в гибком диске 3.5", отформатированной на 720Кбайт?

Количество секторов = 80•9•2

Начало дорожки: GAP(80) + SYNC(11) + IAM(4) + GAP(50) = 146

Размер сектора: SYNC(12) + IDAM(4) + CYL(1) + HEAD(1) + SEC(1) + NL(1) + CRC(1) + GAP(22) + SYNC(12) + DATA_AM(4) + 512 + CRC(1)+GAP(68)= 640

Общий объём: (Количество секторов)•(Размер сектора)+(Количество дорожек)•(Начало дорожки)=944960 байт

Процент информационной ёмкости: (720•1024/944960)•100%=78%

#17. Сколько команд обращения процессора к контроллеру гибкого диска (IN и OUT) надо выполнить, чтобы скопировать полностью информацию с одной дискеты на другую?

Дискета 1.44: к-во дорожек 80•2=160

Чтение: команды чтения дорожки OUT – 9
IN – 7
Итого Out 9•160=1440
Итого IN 7•160=960

Перемещение головок (Seek): OUT = 3 • 80 = 240
Sense Interrupt Status OUT = 1 • 80 = 80 IN = 2 • 80 =160

Итого для ЧТЕНИЯ:

OUT = 1440+240+80 = 1760

IN = 960+160 = 1120

Для записи аналогично
Ответ:
OUT = 1760•2 = 3520
IN = 1120•2 = 2240
Или 5760

#50. Напишите фрагмент программы, задающий стандартные для дискеты 3.5" (1.44M) временные параметры.

Используем макрос OutAL из задачи 60

```
OutAL  3        ; Команда Specify
OutAL  0AFh     ; SRT  | HUT
OutAL  2        ; HLT = 2,  ND=0
```

#47. Напишите фрагмент программы, который определяет число накопителей, подключённых к КГМД и номер активного в настоящий момент накопителя.

```
Port     equ 3F0h
one_mes  db 'Один накопитель $'
two_mes  db 'Два накопителя $'
fisrt    db 'Активен первый $'
second   db 'Активен второй $'

MSG  macro S
       lea   dx, S
       mov   ah, 9
       int   21h
endm
       mov     dx, Port
       in      al, dx      ; SRA
       test    al, 40h
       jz      One
       MSG     two_mes
       jmp     @@1
One:   MSG     one_mes
@@1:   mov     dx, Port+1
       in      al, dx
       test    al, 20h
       jz      F1
       MSG     first
       jmp     exit
F1:
       MSG     second
exit:
```

#48. Напишите фрагмент программы, который реализует командную фазу чтения 44-го сектора.

Используем макрос OutAL из задачи 60

```
OutAl   66h    ;  65h – запись
OutAl   0      ;
OutAl   1      ; cylinder
OutAl   0      ; head
OutAl   9      ; логическая нумерация секторов – от 0
OutAl   2      ; размер сектора
OutAl   18     ; к-во секторов
OutAl   1Bh    ; GPL
OutAl   0FFh   ; DTL
```

```
        WaitInt

        InAl            ; st0
        inal            ; st1
        inal            ; st2
        inal            ; cyl
        inal            ; head
        inal            ; sect
        inal            ; byte in sect
```

#5. Напишите фрагмент программы установки на нулевую дорожку головок жёсткого диска с выдачей сообщения, если возникли ошибки. Временной параметр движения принять равным 3-м.

```
     cli
     mov   dx, 1F7h
L1:  in    al, dx   ; порт команд / байта состояния
     bt    ax, 7    ; проверка
     jnc   L1       ;   готовности

     mov   al, 0001.0011b    ; рекалибровка с временным
               Rec   Spd     ; параметром передвижения 3
     out   dx, al
     sti

; Ожидание выполнения
L2:  in    al, dx
     bt    ax, 7
     jc    L2
     bt    al, 0
     jc    Error
  ..............
Error:
```

#01. Заполните таблицу данных, которые нужно передать в рамках командной фазы форматирования дорожки, на которой расположен 99-й сектор. Форматировать на 1.44 Мбайт.

					01001101b				
					00000100b				
					2				
					18				
					2Ah				
					F6h				
C	2	2	2	2	2	2	2	2	2
H	1	1	1	1	1	1	1	1	1
R	1	3	5	7	9	11	13	15	17
N	2	2	2	2	2	2	2	2	2
C	2	2	2	2	2	2	2	2	2
H	1	1	1	1	1	1	1	1	1
R	2	4	6	8	10	12	14	16	18
N	2	2	2	2	2	2	2	2	2

{Первая строка таблицы дописана позднее.}

#05. С использованием BIOS выполнить чтение данных, расположенных начиная с 129 сектора по 155 сектор гибкого диска, отформатированного на 1.44 Мбайта. Если при чтении обнаружена ошибка - выдать сообщение.

```
CODESEG
        Mov     AH, 02h
        Mov     AL, 16
        Mov     DL, 00h
        Mov     DH, 1
        Mov     CH, 3
        Mov     CL, 3
        Les     BX, [Buf]
        Int     13h
        Jc      Error
        Mov     AH, 02h
        Mov     AL, 11
        Mov     DL, 00h
```

```
        Mov     DH, 0
        Mov     CH, 4
        Mov     CL, 1
        Les     BX, [Buf1]
        Int     13h
        Jc      Error
        . . .
Error:  Mov     AH, 09h
        Lea     DX, [Msg]
        Int     21h

DATASEG
Msg     DB      'Ошибка$'
```

КПД

```
dma_read macro file_name
       cli
       mov al, 5h
       out 0ah, al
       out 0ch, al
       mov al, 01110110b
       out 0bh, al

       mov al, 14; sectors
       mov dx, 1f2h
       out dx, al
       mov al, 2; sector start
       mov dx, 1f3h
       out dx, al
       mov al, 0; dor st
       mov dx, 1f4h
       out dx, al
       mov al, 1; dor ml
       mov dx, 1f5h
       out dx, al
       mov al, 1; head
       mov dx, 1f6h
       out dx, al
       mov al, 25h; read 5 times
       mov dx, 1f7h
       out dx, al

       xor eax, eax
       mov ax, ds
       shl eax, 4
       xor ebx, ebx
       lea bx, root
       add eax, ebx

       out 4h, al
       xchg al, ah
       out 4h, al
       shr eax, 16
       out 81h, al
       xchg al, ah
       out 81h, al

       mov ax, 7168
       out 5h, al
       xchg al, ah
       out 5h, al

       mov al, 2h
       out 0ah, al
       sti
endm
```

#40. Напишите фрагмент программы, который настраивает 2 канал КПД на приём с контроллера гибкого диска 512 байт с адреса 1C000h. Контроллер не имеет буфера.

```
outp  macro P, D
      mov  al, D
      out  P, al
endm

    mov    ax, 1C00h
    mov    es, ax
    xor    di, di
    mov    cx, 512
    ............
     cli
     outp   0Ah, 011b   ;
остановка второго канала DMA
     out    0Ch, al     ; сброс
указателя последовательности
байт

     xor    ebx, ebx
     mov    bx, ds
     shl    ebx, 4
     xor    eax, eax
     mov    ax, di
     add    ebx, eax   ;
Физический адрес в EBX

     outp   4, bl
     outp   4, bh      ; запись
адреса

     shr    ebx, 16
     mov    al, bl
     mov    dx, 81h
     out    dx, al     ; Запись
страницы

     outp   5, cl
     outp   5, ch      ; Count

     outp   0Bh, 10000110b ;
(46h) : FDC to Memory
                              ;
если 4Ah : Memory to FDC

     outp   0Ah, 2          ;
Разрешение работы с каналом.
```

Мышь

int 33h

```
INT 33,0 - Mouse Reset/Get Mouse Installed Flag
 AX = 00
   on return
   AX = 0000  mouse driver not installed
        FFFF  mouse driver installed
   BX = number of buttons
   - resets mouse to default driver values:
     .  mouse is positioned to screen center
     .  mouse cursor is reset and hidden
     .  no interrupts are enabled (mask = 0)
     .  double speed threshold set to 64 mickeys per second
     .  horizontal mickey to pixel ratio (8 to 8)
     .  vertical mickey to pixel ratio (16 to 8)
     .  max width and height are set to maximum for video mode

INT 33,1 - Show Mouse Cursor
 AX = 01
   returns nothing
   - increments the cursor flag;  the cursor is displayed if flag
   is zero;  default flag value is -1

INT 33,2 - Hide Mouse Cursor
 AX = 02
   returns nothing
   - decrements cursor flag; hides cursor if flag is not zero

INT 33,3 - Get Mouse Position and Button Status
 AX = 03
   on return:
   CX = horizontal (X) position  (0..639)
   DX = vertical (Y) position  (0..199)
   BX = button status:
      |F-8|7|6|5|4|3|2|1|0|  Button Status
        |  | | | | | | | `---- left button (1 = pressed)
        |  | | | | | | `----- right button (1 = pressed)
        `------------------- unused
   - values returned in CX, DX are the same regardless of video mode

INT 33,4 - Set Mouse Cursor Position
 AX = 04
   CX = horizontal position
   DX = vertical position
   returns nothing
   - default cursor position is at the screen center
   - the position must be within the range of the current video mode
   - the position may be rounded to fit screen mode resolution

INT 33,5 - Get Mouse Button Press Information
   AX = 5
        BX = 0  left button
             1  right button
        on return:
        BX = count of button presses (0-32767), set to zero after call
        CX = horizontal position at last press
        DX = vertical position at last press
        AX = status:
                |F-8|7|6|5|4|3|2|1|0|  Button Status
                  |  | | | | | | | `---- left button (1 = pressed)
                  |  | | | | | | `----- right button (1 = pressed)
                  `------------------- unused
```

```
INT 33,6 - Get Mouse Button Release Information
   AX = 6
        BX = 0  left button
             1  right button
        on return:
        BX = count of button releases (0-32767), set to zero after call
        CX = horizontal position at last release
        DX = vertical position at last release
        AX = status
                |F-8|7|6|5|4|3|2|1|0|  Button status
                  |  | | | | | | | `---- left button (1 = pressed)
                  |  | | | | | | `----- right button (1 = pressed)
                  `------------------- unused

INT 33,7 - Set Mouse Horizontal Min/Max Position
   AX = 7
        CX = minimum horizontal position
        DX = maximum horizontal position
        returns nothing
        - restricts mouse horizontal movement to window
        - if min value is greater than max value they are swapped

INT 33,8 - Set Mouse Vertical Min/Max Position
   AX = 8
        CX = minimum vertical position
        DX = maximum vertical position
        returns nothing
        - restricts mouse vertical movement to window
        - if min value is greater than max value they are swapped

INT 33,9 - Set Mouse Graphics Cursor
   AX = 9
        BX = horizontal hot spot (-16 to 16)
        CX = vertical hot spot (-16 to 16)
        ES:DX = pointer to screen and cursor masks (16 byte bitmap)
        returns nothing
        - screen mask is AND'ed to screen Cursor Mask is XOR'ed
        - bytes 0-7 form the screen mask bitmap
        - bytes 8-F form the cursor mask bitmap

INT 33,A - Set Mouse Text Cursor
   AX = 0A
        BX = 00  software cursor
             01  hardware cursor
        CX = start of screen mask or hardware cursor scan line
        DX = end of screen mask or hardware cursor scan line
        returns nothing

INT 33,B - Read Mouse Motion Counters
   AX = 0B
        on return:
        CX = horizontal mickey count (-32768 to 32767)
        DX = vertical mickey count (-32768 to 32767)
        - count values are 1/200 inch intervals (1/200 in. = 1 mickey)
```

Программы

Инициализация мыши

```
        mov ax, 0
        int 33h
```

Какая кнопка нажата

```
        mov ax, 3h
        int 33h
left:
        test bx, 1
        jz right
        left_button
right:
        test bx, 00000010b
        jz moves
        right_button
```

Движение мыши

```
moves:
        mov ax, 0bh
        int 33h
        cmp cx, 0
        je skip_x
        cmp cl, 0
        jl go_left
        ;Код для движения вправо
        jmp skip_x
go_left:
```

```
      ;Код для движения влево
skip_x:
      mov cx, dx
      cmp cx, 0
      je skip_y
      clear_cursor
      cmp cl, 0
      jl go_up
      ;Код для движения вниз
      jmp skip_y
go_up:
      ;Код для движения вверх
skip_y:
```

#2. Без использования прерывания Int 33h BIOS напишите фрагмент программы, который принимает 5 байт от мыши, которая работает по протоколу 1200. N81 и отключает питание, если нажата клавиша R.

Прîòîêîë 1200. N81 – PC mouse

```
MouseEvents  db 5 dup(?)
Port  equ   3F8h
ds, es – сегмент данных

      mov    dx, Port + 4
      mov    al, 100b
      out    dx, al
; Инициализация PC-мыши

LL1:  call   GetByte
      mov    ah, al
      and    al, 0F8h
      cmp    al, 80h
      jne    LL1                ;
Перейти, если не первый байт (3)
      test   ah, 1              ;
Нажата правая клавиша?
      jnz    PowerOFF

; Приём остальных байт
      lea    di, MouseEvents
      mov    al, ah
      cld
      stosb
      mov    cx, 4
LL2:  call   GetByte
      stosb
      loop   LL2

; Обработка события
      jmp    LL1

PowerOFF:
      mov    dx, Port+4
      xor    al, al
      out    dx, al             ;
Вылючение мыши


GetByte  proc
      mov    dx,Port+5
no:   in     al,dx
      test   al,1
      jz     no
      sub    dx, 5
      in     al, dx
      ret
GetByte  endp
```

Misc

Вывод строки

```
print macro string
       push ax
       push dx
       lea dx, string
       mov ah, 9
       int 21h
       pop dx
       pop ax
endm
```

Читать из файла

```
read_data macro
       lea dx, file
       mov ah, 3dh
       xor al, al
       int 21h
       mov bx, ax
       mov ah, 3fh
       lea dx, data
       mov cx, 64
       int 21h
endm
```

Создание и запись в файл

```
save_to_file proc
       add count,2
       mov ah,3ch
       ;створити файл
       xor cx,cx
       mov dx,offset fname
       int 21h
       mov handle,ax
       mov ah,40h                    ;писати
в файл
       mov cx,count          ;CX = number of
bytes to write
       mov dx,offset buf     ;DS:DX -> data
to write
       mov bx,handle         ;BX = file
handle
       int 21h
       mov ah,3eh
       ;закрити файл
       mov bx,handle
       int 21h
       ret
save_to_file endp
```

Вывод ax

```
print_ax macro
local l1, l2, l3
       push -1
       mov cx, 10
l1: mov dx, 0
       div cx
       push dx
       cmp ax, 0
       jne l1
       mov ah, 2h
l2: pop dx
       cmp dx, -1
       je l3
       add dl, '0'
       int 21h
       jmp l2
l3:
endm
```

#1. Написать программу, разрешающую IRQ8, IRQ9, IRQ13, IRQ14 и запрещающую IRQ7

```
    in      al, 21h
    or      al, 84h
    out     21h, al       ;
```

Разрешение IRQ7 и IRQ2 – второй контроллер прерываний

```
    in     al, 0A1h
    or      al, 01100011b    ;
```

Ðàçðåøåíèå IRQ8, IRQ9, IRQ13, IRQ14

```
    out     0A1h, al

    INT 08H IRQ 0 Timer
    INT 09H IRQ 1 Keyboard
    INT 0aH IRQ 2 cascade
    INT 0bH IRQ 3 COM 2/4
    INT 0cH IRQ 4 COM 1/3
    INT 0dH IRQ 5 LPT 2
    INT 0eH IRQ 6 diskette
    INT 0fH IRQ 7 LPT 1
    INT 70H IRQ 8 RT Clock
    INT 71H IRQ 9 redir IRQ2
    INT 75H IRQ 13 math chip
    INT 76H IRQ 14 hard disk
```

Клавиатура

#60. Напишите фрагмент программы, который помещает в буфер клавиатуры 2 байта нажатия клавиши 'A' (скан-код 1Eh). Можно использовать BIOS.

```
mov     ah, 5
mov     cx, 1E41h
int     16h
```

#24. С использованием BIOS напишите фрагмент программы, который настраивает таймер клавиатуры таким образом, что при нажатии клавиши в теч. 0.4 сек. в буфер клавиатуры заносится 6 байт.

```
Mov     AH, 03h
Mov     AL, 05h
Mov     BH, 00h
Mov     BL, 0Ch
Int     16h
```

47. Напишите фрагмент программы, без использования BIOS, который определят, что буфер клавиатуры полностью заполнен. (+2)

```
MOV     ES, 40h
MOV     SI, 1Ah
MOV     AX, [word ES:SI]
MOV     SI, 1Ch
MOV     BX, [word ES:SI]
CMP     AX, BX
JE      polon
```

Nepolon:

Polon: буфер клавиатуры заполнен

#24. Написать программу очистки буфера клавиатуры до первого символа A, в нём хранящегося.

```
bios_data     segment at 40h
      org 1Ah
Head   dw ?                    ;pointer to keyboard buffer head
Tail   dw ?                    ;pointer to keyboard buffer tail
     org 80h
StartPos  dw ?                 ;starting keyboard AT-buffer address
(usually 1Eh)
EndPos    dw ?                 ;ending keyboard AT-buffer address
(usually 3Dh)
bios_data     ends

KeyKode_A equ  1E41h           ; Scan(A)+ASCII(A)

Start:
       mov    bx, bios_data            ;point DS to BIOS data area
       mov    ds, bx
       assume ds:bios_data
LL23:
       call   GetHeadKey               ;Определяем символ
       jz     EndProc
       cmp    ax, KeyKode_A
       je     EndProc
       call   GetKey                   ; 'Вытаскиваем' его из очереди
       jmp    LL23
EndProc:

;Insert the keycode in AX into the keyboard buffer.
```

```
;ZF=1 if Buffer is Full, ZF=0 otherwise
Insert proc
       cli
       mov    bx, Tail
       mov    dx, bx
       add    dx, 2                ;calculate next buffer position
       cmp    dx, EndPos           ;did we overshoot the end?
       jne    LL1                  ;no, then continue
       mov    dx, StartPos         ;yes, then wrap around
LL1:   cmp    dx, Head             ;is the buffer full?
       je     LL2                  ;yes, then end now
       mov    [bx],ax              ;insert the keycode
       mov    bx,dx                ;advance the tail
       mov    Tail,bx              ;record its new position
LL2:   sti
       ret
Insert endp

;Get Head the keycode in AX from the keyboard buffer.
;ZF=1 if Buffer is Empty, ZF=0 otherwise
GetHeadKey proc
       cli
       mov    ax, Head
       cmp    ax, Tail
       je     LL65
       mov    bx, Head
       mov    ax,[bx]
       or     bx,bx
LL65:  sti
       ret
GetHeadKey endp

;Retreive the keycode in AX from the keyboard buffer and update Head.
;ZF=1 if Buffer is Empty, ZF=0 otherwise
GetKey proc
       cli
       mov    ax, Head
       cmp    ax, Tail
       je     LL45
       add    ax, 2
       cmp    ax, EndPos
       jb     LL46
       mov    ax,StartPos
LL46:
       xchg   Head, ax
       mov    bx, ax
       mov    ax,[bx]
       or     bx,bx
LL45:  sti
       ret
GetKey endp
       end  start
```

#44. Написать фрагмент подпрограммы обработки прерывания 09h, которая игнорирует нажатие клавиши 'A'.

```
Scan_A_Down  equ   1Eh   ; Скан-код нажатия клавиши A
Scan_A_Up    equ   80h or 1Eh

   KeybordIRQHandler proc
       push    ax
       in      al,60h
       cmp     al,Scan_A_Down
       je      @Up_Down
       cmp     al,Scan_A_Up
       je      @Up_Down
       jmp     IntOldRET
@Up_Down:
       mov     al,20h;-end of interrupt
       out     20h,al
       pop     ax
       iret
IntOldRET:
       pop     ax
       jmp     far int09old
KeybordIRQHandler endp
```

#22. Написать программу настройки клавиатуры таким образом, чтобы при нажатии клавиши в теч 1с в буфер клавиатуры заносилось 4 повторения символа.

```
     mov   al, 0F3h
         out   64h, al         ; Set repeat rate
     mov   al, 34h         ; 0.01.10100 => delay 500ms, Repeat rate = 5
cps
     out   64h, al         ; tut mojet luchshe 750 i 16 cps 0.10.00111
```

#23. На клавиатуре неисправен индикатор "CapsLock". Напишите программу, которая показывает на экране текстовый дубль неисправного индикатора.

```
IDEAL
CODESEG
        Mov     AX, 0040h
        Mov     ES, AX
        Test    [ES:0017h], 01000000b
        Jz      @@1
        Lea     DX, [Str1]
        Jmp     @@2
@@1:    Lea     DX, [Str2]
@@2     Mov     AH, 09h
        Int     21h

DATASEG
Str1    DB      'CapsLock активен$'
Str2     DB       'CapsLock не активен$'
```

#24. С использованием BIOS напишите фрагмент программы, который настраивает таймер клавиатуры таким образом, что при нажатии клавиши в теч. 0.4 сек. в буфер клавиатуры заносится 6 байт.

```
        Mov     AH, 03h
        Mov     AL, 05h
        Mov     BH, 00h; -zaderjka na 0,2 sec
        Mov     BL, 0Ch; - a tut nujno navernoe 8h dlja skorosti 15 simvolov za
sec tak kak za ostalnie 0,2 secundi uspeet peredatsja tolko 15/5 simvolov = 6
bytes
        Int     16h
```

Таймер

#13. Напишите фрагмент программы установки будильника на 12:20:45 микросхемы часов реального времени с использованием BIOS.

```
      mov     ah, 7
      int     1Ah
      mov     ah, 6
      mov     ch, 12h         ; Время заносится
      mov     cl, 20h         ; в BCD
      mov     dl, 45h         ;
      int     1Ah
```

#53. Напишите фрагмент программы, который измерят время выполнения процедуры ALMA с точностью 0.001с

```
      bios_data     segment at 40h
        org 6Ch
        TickCount  dd ?
      bios_data     ends
      _data  segment
        Freq     dt  1.19318E+6
        Divider  dw  256
      _data  ends

SetConst macro Const_
      mov     al,00110110b
      out     43h,al
      mov     ax, Const_
      out     40h, al
      nop
      mov     al, ah
      out     40h, al                ; RConst
endm


      cli
      mov     ax, bios_data
      mov     gs, ax
      assume  gs:bios_data, ds:_data
      xor     eax, eax
      xchg    eax, gs:TickCount
      push    eax
      SetConst Divider
      sti
      call    ALMA
      cli
      fild    dword ptr gs:TickCount
      fdiv    tbyte ptr Freq
      fimul   word ptr Divider
      SetConst  0FFFFh
      pop     eax
      mov     gs:TickCount,eax
      sti
; Результат в ST(0)  в секундах
```

#03. Написать программу выдачи с выхода 2-го канала м/с таймера синхроимпульсов скважностью 2 частотой 10КГц, если на вход CLK2 поступают импульсы частотой 1.9МГц.

```
1.9МГц/10КГц = 190 = 0BEh  – константа
mov     al, 10110110b ;
out     43h,al
```

```
mov     al, 0BEh
out     42h, al
xor     al, al
out     42h, al
```

#64. Написать программу, формирующую импульс на выходе 2-го канала микросхемы таймера через 8 мс, если на вход CLK2 поступают импульсы частотой 1,9МГц.

$$Const = f \cdot t = 15200$$

```
Const    dw   15200

mov     al, 10110000b ;  режим 0,
out     43h,al
mov     ax, Const
out     42h, al
mov     al, ah
out     42h, al
```

#37. Ко входу GATA 2-го канала таймера подключён сигнальный выход периферийного устройства. Напишите фрагмент программы настройки канала таким образом, чтобы по приходу каждого сигнала GATA выдавался отрицательный импульс длительностью 0.034с.

```
Const:=f•t, f=1.9МГц

Const    dw   6460

mov     al, 10110010b ;  режим 1,
out     43h,al
mov     ax, Const
out     42h, al
mov     al, ah
out     42h, al
```

```
-таймер-
;--задержка с помощью таймера >=1с
push 40h
pop ds
mov eax,ds:[6Ch]
add eax,182 ;182=18,2*10sec
l: mov ebx,ds:[6Ch]
   cmp eax,ebx
   jne l

;--задержка с помощью часов реального времени
mov al,0 ;регистр нынешнего значения секунд
out 70h,al
in al,71h
xchg bl,al
add bl,10h ;прибавляем 10 секунд задержки в 2-10системе
daa
cmp bl,59h ;проверяем не вылезло ли за 59сек
```

```
jbe l1     ;если меньше равно
sub bl,60h ;если больше 59сек
das
l1: mov al,2 ;регистр нынешнего значения минут
    out 70h,al
    in al,71h
    xchg cl,al
    add cl,0h ;прибавляем 0минут
    daa
    cmp cl,59h
    jbe l2
    sub cl,60h
    das
l2: mov al,4 ;регистр нынешнего значения часов
    out 70h,al
    in al,71h
    xchg dl,al
    add dl,0 ;прибавляем 0 часов
    daa
    cmp dl,59h
    jbe l3
    sub dl,60h
    das

l3: mov al,0  ;начало проверки что заданное
    out 70h,al ;время совпало с нынешним
    in al,71h
    cmp al,bl
    jne l3
    mov al,2
    out 70h,al
    in al,71h
    cmp al,cl
    jne l3
    mov al,4
    out 70h,al
    in al,71h
    cmp al,dl
    jne l3

;--задержка с помощью таймера на время <1с
mov al,00000100b ;переместить значение счетчика в
out 43h,al       ;промежуточный регистр (Рг)
mov al,00110100b ;00-№ канала,11-читать/писать старший, потом младш байт,
out 43h,al       ;010-2й режим,0-двоичные данные

in al,40h ;читаем состояния счетчика на ax
mov ah,al
in al,40h
mov bx,ax
sub bx,100 ;тк Ct работает в декременте (отнимаем 100 –задержка на 100мкс)

l:
mov al,00000100b ;фиксируем нынешнее состояние Ct в Рг
out 43h,al
;читаем состояния счетчика на AX
```

```
mov al,00110100b
out 43h,al

in al,40h
mov ah,al
in al,40h
cmp ax,bx
jne l

;--определение времени работы пр-ры с помощью таймера
push 40h
pop ds
mov eax,ds:[6Ch] ;нынешнее время
push eax
call proc
mov ebx,ds:[6Ch]
pop eax
sub ebx,eax ;столько работала процедура в тиках
;для того чтобы получить в секундах нужно поделить на 18,2

-клавиатура-
;--очистить буфер клавиатуры до первого хранящегося в нем символа 'a'
push 40h
pop ds
mov ax,word ptr ds:[1Ah] ;смотрим на указатель головы ;1A/1B lower byte????
mov si,ax                ;в si текущее смещение головы

cycle:
cmp byte ptr ds:[si],'a' ;сравниваем текущий символ с искомым
je l
add si, 2 ;тк два байта кода
cmp si,3Ch ;сравнение с концом буфера
jbe next
mov si,1Eh ;перевод на начало буфера
next:
cmp si,word ptr ds:[1Ch] ;сравнение с хвостом
jne cycle

l: mov ax,[si]
   mov word ptr ds:[1Ah],ax ;теперь голова указывает на смещение искомого
символа
                            ;если символ не найден, то буфер очищается
;--посчитать кол-во символов, находящихся в буфере клавиатуры
push 40h
pop ds
mov al,ds:[1Bh] ;голова
mov ah,ds:[1Dh] ;хвост
cmp ah,al
jb l
sub ah,al  ;если хвост позже головы
shr ah,1   ;получаем кол-во символов в буфере,тк один символ имеет 2 байта
jmp exit
l: sub al,ah ;если хвост раньше головы
   mov ah,1Eh ;30 байт длинна буфера
   sub ah,al
   shr ah,1   ;делим на 2
```

```
exit:

;при нажатии клавиши в течении 1с в буфер клавы заносить 4 повторения символа
push 40h
pop ds
mov al,0
out 70h,al
in al,71h
add al,1 ;добавляем 1 секунду
mov bl,al
time: mov si,ds:[1Dh] ;хвост
      mov al,ds:[si] ;в al последний символ
      mov cx,4 ;устанавливаем количество дублируемых символов
cycle: add si,2
       cmp si,3Ah ;не вышел ли указатель за конец буфера 60-2
       jbe l
       mov si,1Eh ;установка указателя на начало буфера
       l: mov ds:[si],al ;дублируем символ
       loop cycle

      mov ax,[si]     ;устанавливаем указатель нового хвоста
      mov ds:[1Dh],ax

      mov al,0
      out 70h,al
      in al,71h
      cmp bl,al
      jne time
```