Московский государственный технический университет
им. Н.Э. Баумана

УТВЕРЖДАЮ:

Большаков С.А.
"__"_____2023 г.

Курсовая работа по курсу «Системное программирование»

Исходный текст программного продукта
(вид документа)

писчая бумага
(вид носителя)

21
(количество листов)

ИСПОЛНИТЕЛИ:

студенты группы ИУ5-45б _____
Коновалов И. Н.
Большаков С.А.
"__"_____2023 г.

Москва – 2023

Содержание

1. Файл tsr2.lst

```
Turbo Assembler    Version 3.1     05/19/23 14:03:31          Page 1
tsr2.asm



      1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      2                             ; tsr.asm
      3                             ;
      4                             ; Сборка:
      5                             ;    tasm.exe /l tsr.asm
      6                             ;    tlink /t /x tsr.obj
      7                             ;
      8                             ; Авторы:
      9                             ;  МГТУ им. Н. Э. Баумана, ИУ5-45Б, 2023 г
     10                             ; Коновалов И.Н.
     11                             ;
     12
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     13
     14 0000                        code segment    'code'
     15                                     assume  CS:code, DS:code
     16                                     org     100h
     17 0100                                _start:
     18
     19 0100  E9 0680                               jmp _initTSR ; на начало
программы
     20                                     ;@Если по варианту нужно, чтобы
происходила замена      символов
     21                                     ; данные
     22 0103  A0 A1 A2 A3 A4 A5      F1+        ignoredChars
          DB                                    +
     23       A6 A7 A8 A9 AA AB      AC+
'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'     ; список игнорируемых
символов
     24       AD AE AF E0 E1 E2 E3+
     25       E4 E5 E6 E7 E8 E9   EA+
     26       EB EC ED EE EF
     27 0124  80 81 82 83 84 85  F0+        replaceWith
DB                                  +
     28       86 87 88 89 8A 8B    8C+
'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЬЭЮЯ'     ; список
заменяемых символов
```

```
    29      8D 8E 8F 90 91 92    93+
    30      94 95 96 97 98 99    9A+
    31      9C 9D 9E 9F
    32      =0041                ignoredLength                equ    $-
ignoredChars                     +
    33                           ; длина строки ignoredChars
    34 0144  00                  ignoreEnabled                DB
0                                +
    35                           ; флаг функции игнорирования ввода
    36 0145  4B 56 59 4A 47          translateFrom                DB
        'KVYJG'                      +
    37                           ; символы для замены
    38 014A  8B 8C 8D 8E 8F          translateTo                  DB
        'ЛМНОП'                          +
    39                           ; символы на которые будет      идти замена
    40      =0005                translateLength              equ    $-
translateTo                      +
    41                           ; длина строки trasnlateFrom
    42 014F  00                  translateEnabled
DB      0                        +
    43                           ; флаг функции перевода
    44
    45 0150  00                  signaturePrintingEnabled     DB
        0                            +
    46                           ; флаг функции вывода информации об авторе
    47 0151  00                  cursiveEnabled               DB
0                                +
    48                           ; флаг перевода символа в курсив
    49
    50 0152  3F                  cursiveSymbol
DB    00111111b                  +
    51                           ;@ символ,      составленный из  единичек
(его курсивный    вариант)
    52 0153  21
DB    00100001b
    53 0154  22
DB    00100010b
    54 0155  24
DB    00100100b
    55 0156  78
DB    01111000b
    56 0157  40
DB    01000000b
```

57 0158 40
DB 01000000b

```
   58 0159  70
DB    01110000b
   59 015A  48
DB    01001000b
   60 015B  48
DB    01001000b
   61 015C  44
DB    01000100b
   62 015D  44
DB    01000100b
   63 015E  42
DB    01000010b
   64 015F  82
DB    10000010b
   65 0160  84
DB    10000100b
   66 0161  F8
DB    11111000b
   67
   68 0162  82                    charToCursiveIndex                DB
      'B'                              +
   69                          ;@ символ для замены
   70 0163  10*(FF)                    savedSymbol
DB    16 dup(0FFh)                     +
   71                          ; переменная для хранения старого символа
   72
   73     =00FF                        true                          equ
0FFh                  +
   74                          ; константа истинности
   75 0173  ????                       old_int9hOffset               DW
        ?                             +
   76                          ; адрес старого обработчика int 9h
   77 0175  ????                       old_int9hSegment
DW       ?                             +
   78                          ; сегмент старого обработчика int 9h
   79 0177  ????                       old_int1ChOffset
DW       ?                             +
   80                          ; адрес старого обработчика int 1Ch
```

```
   81 0179  ????                    old_int1ChSegment                DW
          ?                              +
   82                          ; сегмент старого обработчика int 1Ch
   83 017B  ????                    old_int2FhOffset
DW        ?                         +
   84                          ; адрес старого обработчика int 2Fh
   85 017D  ????                    old_int2FhSegment                DW
          ?                              +
   86                          ; сегмент старого обработчика int 2Fh
   87
   88 017F  00                      notLoadTSR
DB        0                         +
   89                          ; 1 - не загружать
   90 0180  0000                    counter                          DW
          0
   91       =0002                   printDelay
equ     2                          +
   92                          ;@ задержка перед выводом "подписи" в
секундах
   93
   94
   95                                   ;@ заменить на собственные
        данные.       формирование таблицы идет по строке большей длины
+
   96                               (1я строка).
   97 0182  B3 8A AE A6 A8 A5       A2+       signatureLine1
DB        179, 'Кожиев Таймураз Казбекович          +
   98       20 92 A0 A9 AC E3  E0+  ',        179
   99       A0 A7 20 8A A0 A7  A1+
  100       A5 AA AE A2 A8 E7        20+
  101       20 20 20 20 20 20     20+
  102       20 20 20 20 20 20     20+
  103       20 20 20 20 20 20     20+
  104       20 20 B3
  105       =0034                   Line1_length                     equ
$-signatureLine1
  10601B6  B3 88 93 35 2D 34        35+       signatureLine2
DB        179, 'ИУ5-45                 +
  107       20 20 20 20 20 20     20+  ',179
  108       20 20 20 20 20 20     20+
  109       20 20 20 20 20 20     20+
  110       20 20 20 20 20 20     20+
  111       20 20 20 20 20 20     20+
  112       20 20 20 20 20 20     20+
```

```
113     20 20 B3
114     =0034                   Line2_length                    equ
$-signatureLine2
```

```
 11501EA  B3 82 A0 E0 A8 A0      AD+     signatureLine3
DB      179, 'Вариант #7                +
 116     E2 20 23 37 20 20      20+  ',179
 117     20 20 20 20 20 20      20+
 118     20 20 20 20 20 20      20+
 119     20 20 20 20 20 20      20+
 120     20 20 20 20 20 20      20+
 121     20 20 20 20 20 20      20+
 122     20 20 B3
 123     =0034                  Line3_length                    equ
$-signatureLine3
 124021E  3E 74 73 72 2E 63 6F+      helpMsg DB      '>tsr.com [/?]', 10,
13
 125     6D 20 5B 2F 3F 5D   0A+
 126     0D
 127022D  20 5B 2F 3F 5D 20      2D+             DB      ' [/?] -
вывод данной справки', 10, 13
 128     20 A2 EB A2 AE A4 20+
 129     A4 A0 AD AD AE A9      20+
 130     E1 AF E0 A0 A2 AA A8+
 131     0A 0D
 132024B  20 20 46 33 20 20  2D+             DB      '  F3  - вывод
ФИО и группы по таймеру внизу экрана', 10, 13
 133     20 A2 EB A2 AE A4 20+
 134     94 88 8E 20 A8 20      A3+
 135     E0 E3 AF AF EB 20      AF+
 136     AE 20 E2 A0 A9 AC A5+
 137     E0 E3 20 A2 AD A8      A7+
 138     E3 20 ED AA E0 A0      AD+
 139     A0 0A 0D
 140027F  20 20 46 34 20 20  2D+             DB      '  F4  -
включение и отключения  курсивного вывода русского символа  +
 141     20 A2 AA AB EE E7 A5+  B', 10, 13
 142     AD A8 A5 20 A8 20      AE+
 143     E2 AA AB EE E7 A5      AD+
 144     A8 EF 20 AA E3 E0      E1+
 145     A8 A2 AD AE A3 AE      20+
 146     A2 EB A2 AE A4 A0      20+
 147     E0 E3 E1 E1 AA AE      A3+
```

```
148      AE 20 E1 A8 AC A2 AE+
149      AB A0 20 82 0A 0D
150 02C4 20 20 46 35 20 20 2D+                    DB      ' F5 -
включение и отключение      частичной русификации клавиатуры   +
151      20 A2 AA AB EE E7 A5+ (KVYJG -> ЛМНОП)',    10, 13
152      AD A8 A5 20 A8 20 AE+
153      E2 AA AB EE E7 A5 AD+
154      A8 A5 20 E7 A0 E1 E2+
155      A8 E7 AD AE A9 20 E0+
156      E3 E1 A8 E4 A8 AA A0+
157      E6 A8 A8 20 AA AB A0+
158      A2 A8 A0 E2 E3 E0 EB+
159      28 4B 56 59 4A 47 20+
160      2D 3E 20 8B 8C 8D 8E+
161      8F 29 0A 0D
162 0315 20 20 46 36 20 20 2D+                    DB      ' F6 -
включение и отключение      режима перевода строчных на        +
163      20 A2 AA AB EE E7 A5+ прописные', 10, 13
164      AD A8 A5 20 A8 20 AE+
165      E2 AA AB EE E7 A5 AD+
166      A8 A5 20 E0 A5 A6 A8+
167      AC A0 20 AF A5 E0 A5+
168      A2 AE A4 A0 20 E1 E2+
169      E0 AE E7 AD EB E5 20+
170      AD A0 20 AF E0 AE AF+
171      A8 E1 AD EB A5 0A 0D
```

```
   172
   173      =013D                   helpMsg_length                  equ
$-helpMsg
   174035B  8E E8 A8 A1 AA A0       20+        errorParamMsg
            DB              'Ошибка параметров коммандной  +
   175      AF A0 E0 A0 AC A5E2+  строки', 10, 13
   176      E0 AE A2 20 AA AEAC+
   177      AC A0 AD A4 AD AE       A9+
   178      20 E1 E2 E0 AE AA  A8+
   179      0A 0D
   180      =0025                   errorParamMsg_length            equ
$-errorParamMsg
   181
   182 0380  DA 32*(C4) BF                 tableTop
            DB              218, Line1_length-2 dup+
   183                              (196), 191
   184      =0034                   tableTop_length
equ     $-tableTop
   185 03B4  C0 32*(C4) D9                 tableBottom
DB          192, Line1_length-2 dup (196), +
   186                              217
   187      =0034                   tableBottom_length              equ
$-tableBottom
   188
   189                              ; сообщения
   190 03E8  90 A5 A7 A8 A4 A5      AD+        installedMsg
            DB          'Резидент загружен!$'
   191      E2 20 A7 A0 A3 E0   E3+
   192      A6 A5 AD 21 24
   193 03FB  90 A5 A7 A8 A4 A5      AD+        alreadyInstalledMsg
DB      'Резидент уже загружен$'
   194      E2 20 E3 A6 A5 20    A7+
   195      A0 A3 E0 E3 A6 A5  AD+
   196      24
   197 0411  8D A5 A4 AE E1 E2      A0+        noMemMsg
                        DB          'Недостаточно памяти$'
   198      E2 AE E7 AD AE 20 AF+
   199      A0 AC EF E2 A8 24
```

```
   200 0425  8D A5 20 E3 A4 A0        AB+       notInstalledMsg
           DB          'Не удалось загрузить резидент$'
   201       AE E1 EC 20 A7 A0  A3+
   202       E0 E3 A7 A8 E2 EC  20+
   203       E0 A5 A7 A8 A4 A5  AD+
   204       E2 24
   205
   206 0443  90 A5 A7 A8 A4 A5        AD+       removedMsg
                   DB          'Резидент выгружен'
   207       E2 20 A2 EB A3 E0  E3+
   208       A6 A5 AD
   209       =0011                   removedMsg_length
equ     $-removedMsg
   210
   211 0454  8D A5 20 E3 A4 A0        AB+       noRemoveMsg
           DB          'Не удалось выгрузить резидент'
   212       AE E1 EC 20 A2 EB  A3+
   213       E0 E3 A7 A8 E2 EC  20+
   214       E0 A5 A7 A8 A4 A5  AD+
   215       E2
   216       =001D                   noRemoveMsg_length
equ     $-noRemoveMsg
   217
   218 0471  46 33                   f1_txt                          DB
           'F3'
   219 0473  46 34                   f2_txt                          DB
           'F4'
   220 0475  46 35                   f3_txt                          DB
           'F5'
   221 0477  46 36                   f4_txt                          DB
           'F6'
   222       =0002                   fx_length                       equ
$-f4_txt
   223
   224 0479                          changeFx proc
   225 0479  50                            push AX
   226 047A  53                            push BX
   227 047B  51                            push CX
   228 047C  52                            push DX
```

```
0047D  55                       push BP
0047E  06                       push ES
0047F  33 DB                            xor BX, BX

0481  B4 03                     mov AH, 03h
0483  CD 10                     int 10h
0485  52                        push DX

0486  0E                        push CS
0487  07                        pop ES

0488                    _checkF1:
0488  BD 0471r                          lea BP, f1_txt
048B  B9 0002                           mov CX, fx_length
048E  B7 00                     mov BH, 0
0490  B6 00                     mov DH, 0
0492  B2 4E                     mov DL, 78
0494  B8 1301                           mov AX, 1301h

0497  80 3E 0150r FF                   cmp
signaturePrintingEnabled, true
049C  74 07                     je  _greenF1

049E                    _redF1:
049E  B3 4F                             mov BL, 01001111b ; red
04A0  CD 10                                     int 10h
04A2  EB 08 90                                  jmp _checkF2

04A5                    _greenF1:
04A5  BD 0471r                                  lea BP, f1_txt
04A8  B3 2F                                     mov BL, 00101111b ;
green
04AA  CD 10                                     int 10h

04AC                    _checkF2:
04AC  BD 0473r                          lea BP, f2_txt
04AF  B9 0002                           mov CX, fx_length
04B2  B7 00                     mov BH, 0
04B4  B6 01                     mov DH, 1
```

```
26604B6  B2 4E                      mov DL, 78
26704B8  B8 1301                    mov AX, 1301h
268
26904BB  80 3E 0151r FF             cmp cursiveEnabled, true
27004C0  74 07                  je  _greenF2
271
27204C2                         _redF2:
27304C2  B3 4F                      mov BL, 01001111b ; red
27404C4  CD 10                          int 10h
27504C6  EB 05 90                       jmp _checkF3
276
27704C9                         _greenF2:
27804C9  B3 2F                      mov BL, 00101111b ; green
27904CB  CD 10                          int 10h
280
28104CD                     _checkF3:
28204CD  BD 0475r                   lea BP, f3_txt
28304D0  B9 0002                    mov CX, fx_length
28404D3  B7 00                  mov BH, 0
28504D5  B6 02                  mov DH, 2
```

```
28604D7  B2 4E                  mov DL, 78
28704D9  B8 1301                mov AX, 1301h
288
28904DC  80 3E 014Fr FF         cmp translateEnabled, true
29004E1  74 07                  je  _greenF3
291
29204E3                         _redF3:
29304E3  B3 4F                  mov BL, 01001111b ; red
29404E5  CD 10                      int 10h
29504E7  EB 05 90                   jmp _checkF4
296
29704EA                         _greenF3:
29804EA  B3 2F                      mov BL, 00101111b ;
green
29904EC  CD 10                      int 10h
300
30104EE                     _checkF4:
30204EE  BD 0477r               lea BP, f4_txt
30304F1  B9 0002                mov CX, fx_length
30404F4  B7 00                  mov BH, 0
30504F6  B6 03                  mov DH, 3
30604F8  B2 4E                  mov DL, 78
30704FA  B8 1301                mov AX, 1301h
308
30904FD  80 3E 0144r FF         cmp ignoreEnabled,  true
3100502  74 07                  je  _greenF4
311
3120504                         _redF4:
3130504  B3 4F                  mov BL, 01001111b ; red
3140506  CD 10                  int 10h
3150508  EB 05 90                   jmp _outFx
316
317050B                         _greenF4:
318050B  B3 2F                  mov BL, 00101111b ; green
319050D  CD 10                      int 10h
320
321050F                     _outFx:
322050F  5A                     pop DX
3230510  B4 02                  mov AH, 02h
```

```
3240512  CD 10                    int 10h
325
3260514  07                      pop ES
3270515  5D                      pop BP
3280516  5A                      pop DX
3290517  59                      pop CX
3300518  5B                      pop BX
3310519  58                      pop AX
332051A  C3                      ret
333051B                       changeFx endp
334
335                           ;новый      обработчик
336051B                       new_int9h proc    far
337                                   ; сохраняем значения всех,
    изменяемых регистров в стэке
338051B  56                      push SI
339051C  50                      push AX
340051D  53                      push BX
341051E  51                      push CX
342051F  52                      push DX
```

```
   3430520  06                              push ES
   3440521  1E                              push DS
   345                                      ; синхронизируем CS и DS
   3460522  0E                              push CS
   3470523  1F                              pop  DS
   348
   3490524  B8 0040                              mov  AX, 40h ; 40h-
сегмент,где хранятся  флаги сост-я клавиатуры, кольц.     +
   350                                  буфер ввода
   3510527  8E C0                           mov  ES, AX
   3520529  E4 60                           in   AL, 60h ; записываем в AL
скан-код  нажатой  клавиши
   353
   354
   355                                      ;@ далее -     код для     всех
вариантов
   356
   357                                      ;проверка F3-F6
   358052B                                  _test_Fx:
   359052B  2C 3A                                sub AL, 58    ; в AL теперь
номер функциональной клавиши
   360052D                                  _F1:
   361052D  3C 03                                cmp AL, 3 ; F1
   362052F  75 0A                                jne _F2
   3630531  F6 16 0150r                          not signaturePrintingEnabled
   3640535  E8 FF41                                    call changeFx
   3650538  EB 2E 90                                   jmp
_translate_or_ignore
   366053B                                  _F2:
   367053B  3C 04                                cmp AL, 4 ; F2
   368053D  75 0D                                      jne _F3
   369053F  F6 16 0151r                          not cursiveEnabled
   3700543  E8 FF33                                    call changeFx
   3710546  E8 01D3                                    call setCursive ;
перевод символа в курсив     и обратно в зависимости     от  +
   372                              флага cursiveEnabled
   3730549  EB 1D 90                                   jmp
_translate_or_ignore
   374054C                                  _F3:
```

```
   375054C  3C 05                          cmp AL, 5 ; F3
   376054E  75 0A                          jne _F4
   3770550  F6 16 014Fr                    not translateEnabled
   3780554  E8 FF22                            call changeFx
   3790557  EB 0F 90                           jmp
_translate_or_ignore
   380055A                             _F4:
   381055A  3C 06                          cmp AL, 6 ; F4
   382055C  75 0A                          jne _translate_or_ignore
   383055E  F6 16 0144r                    not ignoreEnabled
   3840562  E8 FF14                            call changeFx
   3850565  EB 01 90                           jmp
_translate_or_ignore
   386
   387                                 ;игнорирование и перевод
   3880568                             _translate_or_ignore:
   389
   3900568  9C                        pushf
   3910569  2E: FF 1E    0173r              call dword      ptr
CS:[old_int9hOffset] ; вызываем стандартный     обработчик прерывания
   392056E  B8 0040                        mov     AX, 40h         ;
40h-сегмент,где хранятся      флаги сост-я клавы,кольц.   +
   393                          буфер ввода
   3940571  8E C0                     mov     ES, AX
   3950573  26: 8B 1E    001C              mov     BX, ES:[1Ch]    ;
адрес хвоста
   3960578  4B                        dec     BX          ; сместимся
назад к последнему
   3970579  4B                        dec     BX          ; введённому
символу
   398057A  83 FB 1E                       cmp     BX, 1Eh ; не вышли
      ли мы за пределы буфера?
   399057D  73 03                     jae     _go
```

```
   400057F  BB 003C                    mov     BX, 3Ch ; хвост
вышел за пределы буфера, значит последний введённый    +
   401                          символ
   402                                                  ; находится в
конце буфера
   403
   4040582                      _go:
   4050582  26: 8B 17                  mov DX, ES:[BX] ; в DX 0
введённый  символ
   406                                  ;включен ли режим блокировки
ввода?
   4070585  80 3E 0144r FF             cmp ignoreEnabled,    true
   408058A  75 21                      jne _check_translate
   409
   410                                  ; да, включен
   411058C  BE 0000                    mov SI, 0
   412058F  B9 0041                    mov CX, ignoredLength ;кол-
во игнорируемых символов
   413
   414                                  ; проверяем, присутствует ли
текущий символ в списке игнорируемых
   4150592                      _check_ignored:
   4160592  3A 94 0103r                cmp DL,ignoredChars[SI]
   4170596  74 06                      je  _block
   4180598  46                         inc SI
   4190599  E2 F7                  loop _check_ignored
   420059B  EB 10 90                   jmp _check_translate
   421
   422                              ; блокируем
   423059E                          _block:
   424                                  ;@mov ES:[1Ch], BX
;блокировка ввода символа
   425059E  26: 89 07                      mov ES:[BX], AX
   42605A1  33 C0                      xor AX, AX
   42705A3  8A 84 0124r                mov AL, replaceWith[SI]
   42805A7  26: 89 07                  mov ES:[BX], AX         ;
замена символа
   42905AA  EB 23 90                   jmp _quit
   430
```

```
   43105AD                          _check_translate:
   432                                      ; включен ли режим  перевода?
   43305AD  80 3E 014Fr FF                  cmp translateEnabled, true
   43405B2  75 1B                           jne _quit
   435
   436                                      ; да, включен
   43705B4  BE 0000                         mov SI, 0
   43805B7  B9 0005                         mov CX, translateLength ;
кол-во символов для перевода
   439                                      ; проверяем, присутствует ли
текущий символ в списке для перевода
   44005BA                          _check_translate_loop:
   44105BA  3A 94 0145r                             cmp DL,
translateFrom[SI]
   44205BE  74 06                           je  _translate
   44305C0  46                              inc SI
   44405C1  E2 F7                           loop _check_translate_loop
   44505C3  EB 0A 90                        jmp _quit
   446
   447                                      ; переводим
   44805C6                          _translate:
   44905C6  33 C0                           xor AX, AX
   45005C8  8A 84 014Ar                             mov AL,
translateTo[SI]
   45105CC  26: 89 07                               mov ES:[BX], AX ;
замена символа
   452
   45305CF                          _quit:
   454                                      ; восстанавливаем все регистры
   45505CF  1F                              pop     DS
   45605D0  07                              pop     ES
```

```
45705D1  5A                     pop DX
45805D2  59                     pop CX
45905D3  5B                     pop     BX
46005D4  58                     pop     AX
46105D5  5E                     pop SI
46205D6  CF                     iret
46305D7                 new_int9h endp
464
465                     ;=== Обработчик прерывания      int 1Ch
    ===;
466                     ;=== Вызывается каждые 55 мс ===;
46705D7                 new_int1Ch      proc far
46805D7  50                     push AX
46905D8  0E                     push CS
47005D9  1F                     pop DS
471
47205DA  9C                     pushf
47305DB  2E: FF 1E   0177r              call dword      ptr
CS:[old_int1ChOffset]
474
47505E0  80 3E 0150r FF                 cmp signaturePrintingEnabled, true
    ; если нажата управляющая клавиша (в данном случае  +
476                     F1)
47705E5  75 1C                  jne _notToPrint
478
47905E7  83 3E 0180r 25                 cmp counter,
printDelay*1000/55 + 1 ; если  кол-во "тактов"    эквивалентно          +
480                     %printDelay% секундам
48105EC  74 03                  je  _letsPrint
482
48305EE  EB 0E 90                       jmp _dontPrint
484
48505F1                         _letsPrint:
48605F1  F6 16 0150r                    not signaturePrintingEnabled
48705F5  C7 06 0180r 0000                       mov counter, 0
48805FB  E8 0094                        call printSignature
489
49005FE                         _dontPrint:
49105FE  83 06 0180r 01                 add counter, 1
```

```
492
0603                    _notToPrint:
494
0603  58                pop AX
496
0604  CF                iret
0605                new_int1Ch      endp
499
500                     ;=== Обработчик прерывания      int 2Fh
    ===;
501                     ;=== Служит для:
502                     ;===  1) проверки факта присутствия TSR в
памяти (при AH=0FFh, AL=0)
503                     ;===     будет возвращён AH='i' в случае, если
TSR   уже загружен
504                     ;===  2) выгрузки TSR из памяти (при
AH=0FFh, AL=1)
505                     ;===
0605                new_int2Fh      proc
0605  80 FC FF                cmp     AH, 0FFh            ;наша
функция?
0608  75 0B                 jne     _2Fh_std      ;нет - на старый
обработчик
060A  3C 00                 cmp     AL, 0   ;подфункция проверки,
загружен ли резидент    в память?
060C  74 0C                 je      _already_installed
060E  3C 01                 cmp     AL, 1   ;подфункция выгрузки из
памяти?
0610  74 0B                 je      _uninstall
0612  EB 01 90                  jmp     _2Fh_std            ;нет - на
старый обработчик
```

```
514
5150615                 _2Fh_std:
5160615 2E: FF 2E 017Br         jmp     dword ptr
CS:[old_int2FhOffset] ;вызов старого обработчика
517
518061A                 _already_installed:
519061A B4 69                   mov     AH, 'i' ;вернём 'i', если
резидент     загружен     в память
520061C CF                      iret
521
522061D                 _uninstall:
523061D 1E                      push    DS
524061E 06                      push    ES
525061F 52                      push    DX
5260620 53                      push    BX
527
5280621 33 DB                   xor BX, BX
529
530                             ; CS = ES,        для доступа к
переменным
5310623 0E                      push CS
5320624 07                      pop ES
533
5340625 B8 2509                 mov     AX, 2509h
5350628 26: 8B 16    0173r      mov DX, ES:old_int9hOffset
        ; возвращаем вектор прерывания
536062D 26: 8E 1E    0175r      mov DS, ES:old_int9hSegment    ; на
место
5370632 CD 21                   int     21h
538
5390634 B8 251C                 mov     AX, 251Ch
5400637 26: 8B 16    0177r      mov DX, ES:old_int1ChOffset       ;
возвращаем вектор прерывания
541063C 26: 8E 1E    0179r      mov DS, ES:old_int1ChSegment
; на место
5420641 CD 21                   int     21h
543
5440643 B8 252F                 mov     AX, 252Fh
```

```
   5450646  26: 8B 16     017Br            mov DX, ES:old_int2FhOffset       ;
возвращаем вектор прерывания
   546064B  26: 8E 1E    017Dr          mov DS, ES:old_int2FhSegment
; на место
   5470650  CD 21                  int     21h
   548
   5490652  2E: 8E 06     002C             mov     ES, CS:2Ch         ; загрузим
       в ES адрес окружения
   5500657  B4 49                  mov     AH, 49h             ; выгрузим
       из памяти окружение
   5510659  CD 21                  int     21h
   552065B  72 0B                  jc  _notRemove
   553
   554065D  0E                     push    CS
   555065E  07                     pop     ES              ;в  ES - адрес
резидентной программы
   556065F  B4 49                  mov     AH, 49h  ;выгрузим из памяти
резидент
   5570661  CD 21                  int     21h
   5580663  72 03                  jc  _notRemove
   5590665  EB 15 90                       jmp _unloaded
   560
   5610668                     _notRemove: ; не удалось выполнить
       выгрузку
   562                             ; вывод сообщения о неудачной выгрузке
   5630668  B4 03                    mov AH, 03h                       ;
получаем    позицию     курсора
   564066A  CD 10                          int 10h
   565066C  BD 0454r                       lea BP, noRemoveMsg
   566066F  B9 001D                        mov CX, noRemoveMsg_length
   5670672  B3 07                  mov BL, 0111b
   5680674  B8 1301                        mov AX, 1301h
   5690677  CD 10                  int 10h
   5700679  EB 12 90                       jmp _2Fh_exit
```

```
571
572 067C                    _unloaded:      ; выгрузка прошла успешно
573                                        ; вывод сообщения об удачной выгрузке
574 067C B4 03                     mov AH, 03h                     ;
получаем позицию курсора
575 067E CD 10                         int 10h
576 0680 BD 0443r                      lea BP, removedMsg
577 0683 B9 0011                       mov CX, removedMsg_length
578 0686 B3 07                     mov BL, 0111b
579 0688 B8 1301                       mov AX, 1301h
580 068B CD 10                         int 10h
581
582 068D                    _2Fh_exit:
583 068D 5B                        pop BX
584 068E 5A                        pop     DX
585 068F 07                        pop     ES
586 0690 1F                        pop     DS
587 0691 CF                        iret
588 0692                    new_int2Fh      endp
589
590                         ;=== Процедура вывода подписи (ФИО,
группа)
591                         ;=== Настраивается  значениями переменных
в       начале исходника
592                         ;===
593 0692                    printSignature proc
594 0692 50                        push AX
595 0693 52                        push DX
596 0694 51                        push CX
597 0695 53                        push BX
598 0696 06                        push ES
599 0697 54                        push SP
600 0698 55                        push BP
601 0699 56                        push SI
602 069A 57                        push DI
603
604 069B 33 C0                     xor AX, AX
605 069D 33 DB                         xor BX, BX
606 069F 33 D2                     xor DX, DX
```

```
607
60806A1  B4 03                mov AH, 03h
;чтение текущей позиции курсора
60906A3  CD 10                    int 10h
61006A5  52                   push DX
;помещаем информацию о        +
611                           положении курсора в стек
612
613
614                                ;по варианту вывод по низу
61506A6                                _printBottom:
61606A6  B6 13                      mov DH, 19
61706A8  B2 0F                           mov DL, 15
61806AA  EB 01 90                        jmp _actualPrint
619
62006AD                       _actualPrint:
62106AD  B4 0F                                mov AH, 0Fh
        ;чтение текущего видеорежима. в+
622                           BH        - текущая страница
62306AF  CD 10                           int 10h
624
62506B1  0E                        push CS
62606B2  07                        pop ES
;указываем ES на CS
627
```

```
   628                                  ;вывод 'верхушки' таблицы
   62906B3  52                          push DX
   63006B4  BD 0380r                            lea BP, tableTop
                    ;помещаем в BP указатель на    +
   631                              выводимую строку
   63206B7  B9 0034                             mov CX, tableTop_length
;в      CX - длина строки
   63306BA  B3 07                               mov BL, 0111b
;цвет выводимого текста ref:      +
   634
http://en.wikipedia.org/wiki/BIOS_color_attributes
   63506BC  B8 1301                             mov AX, 1301h
                    ;AH=13h - номер ф-ии, AL=01h - +
   636                              курсор перемещается при выводе каждого из
символов    строки
   63706BF  CD 10                               int 10h
   63806C1  5A                          pop DX
   63906C2  FE C6                               inc DH
   640
   641
   642                                  ;вывод первой линии
   64306C4  52                          push DX
   64406C5  BD 0182r                            lea BP, signatureLine1
   64506C8  B9 0034                             mov CX, Line1_length
   64606CB  B3 07                               mov BL, 0111b
   64706CD  B8 1301                             mov AX, 1301h
   64806D0  CD 10                               int 10h
   64906D2  5A                          pop DX
   65006D3  FE C6                               inc DH
   651
   652                                  ;вывод второй линии
   65306D5  52                          push DX
   65406D6  BD 01B6r                            lea BP, signatureLine2
   65506D9  B9 0034                             mov CX, Line2_length
   65606DC  B3 07                               mov BL, 0111b
   65706DE  B8 1301                             mov AX, 1301h
   65806E1  CD 10                               int 10h
   65906E3  5A                          pop DX
   66006E4  FE C6                               inc DH
```

```
661
662                                     ;вывод третьей линии
66306E6  52                             push DX
66406E7  BD 01EAr                               lea BP, signatureLine3
66506EA  B9 0034                                mov CX, Line3_length
66606ED  B3 07                                  mov BL, 0111b
66706EF  B8 1301                                mov AX, 1301h
66806F2  CD 10                                  int 10h
66906F4  5A                             pop DX
67006F5  FE C6                          inc DH
671
672                                     ;вывод 'низа' таблицы
67306F7  52                             push DX
67406F8  BD 03B4r                               lea BP, tableBottom
67506FB  B9 0034                                mov CX, tableBottom_length
67606FE  B3 07                          mov BL, 0111b
6770700  B8 1301                                mov AX, 1301h
6780703  CD 10                          int 10h
6790705  5A                             pop DX
6800706  FE C6                          inc DH
681
6820708  33 DB                                 xor BX, BX
683070A  5A                                     pop DX
;восстанавливаем из стека       +
684                             прежнее положение курсора
```

```
   685070B  B4 02                    mov AH, 02h
;меняем положение курсора на   +
   686                       первоначальное
   687070D  CD 10                          int 10h
   688070F  E8 FD67                        call changeFx
   689
   6900712  5F                   pop DI
   6910713  5E                   pop SI
   6920714  5D                   pop BP
   6930715  5C                   pop SP
   6940716  07                   pop ES
   6950717  5B                   pop BX
   6960718  59                   pop CX
   6970719  5A                   pop DX
   698071A  58                   pop AX
   699
   700071B  C3                   ret
   701071C                   printSignature endp
   702
   703                       ;=== Функция, которая в зависимости от
флага cursiveEnabled меняет   начертание символа с курсива+
   704                       на обычное      и наоброт
   705                       ;=== Сама смена происходит      в процедуре
changeFont, а здесь       подготавливаются данные
   706071C                   setCursive      proc
   707071C  06                   push ES ; сохраняем регистры
   708071D  50                   push AX
   709071E  0E                   push CS
   710071F  07                   pop ES
   711
   7120720  80 3E 0151r FF               cmp cursiveEnabled, true
   7130725  75 30                jne _restoreSymbol
   714                           ; если флаг равен true, выполняем
замену символа на курсивный вариант,
   715                           ; предварительно сохраняя старый
символ в savedSymbol
   716
   7170727  E8 004C                      call saveFont
   718072A  8A 0E 0162r                  mov CL, charToCursiveIndex
```

```
719072E                 _shifTtable:
720                             ; мы получаем в BP  таблицу     всех
символов. адрес указывает на символ 0
721                             ; поэтому нуэно совершить сдвиг 16*X -
где    X - код      символа
722072E  83 C5 10               add BP, 16
7230731  E2 FB                  loop _shiftTable
724
725                             ; при savefont смещается регистр ES
726                             ; поэтому приходится делать такие
махинации, чтобы
727                             ; записать     полученный элемент в
savedSymbol
728                             ; swap(ES,     DS) и сохранение
старого значения DS
7290733  1E                     push DS
7300734  58                     pop AX
7310735  06                     push ES
7320736  1F                     pop DS
7330737  50                     push AX
7340738  07                     pop ES
7350739  50                     push AX
736
737073A  8B F5                  mov SI, BP
738073C  BF 0163r               lea DI, savedSymbol
739                             ; сохраняем в переменную savedSymbol
740                             ; таблицу нужного символа
741073F  B9 0010                mov CX, 16
```

```
                                ; movsb из      DS:SI в      ES:DI
0742 F3> A4                             rep movsb
                                ; исходные      позиции      сегментов возвращены
0744 1F                         pop DS ; восстановление DS

                                ; заменим написание символа на курсив
0745 B9 0001                            mov CX, 1
0748 B6 00                      mov DH, 0
074A 8A 16 0162r                        mov DL, charToCursiveIndex
074E BD 0152r                           lea BP, cursiveSymbol
0751 E8 0015                            call changeFont
0754 EB 10 90                           jmp _exitSetCursive

0757                            _restoreSymbol:
                                ; если флаг равен 0, выполняем замену курсивного символа на старый      вариант

0757 B9 0001                            mov CX, 1
075A B6 00                      mov DH, 0
075C 8A 16 0162r                        mov DL, charToCursiveIndex
0760 BD 0163r                           lea bp, savedSymbol
0763 E8 0003                            call changeFont

0766                            _exitSetCursive:
0766 58                                 pop AX
0767 07                                 pop ES
0768 C3                                 ret
0769                            setCursive      endp

                                ;=== Функция смены      начертания символа (курсив/нормальное)
                                ;===
                                ; *** входные данные
                                ; DL = номер символа для замены
                                ; CX = Кол-во символов заменяемых изображений символов
                                ; (начиная      с символа указанного в DX)
                                ; ES:bp = адрес таблицы
```

```
   777                          ;
   778                          ; *** описание работы процедуры
   779                          ; Происходит вызов  int 10h
       (видеосервис)
   780                          ; с функцией AH = 11h (функции
знакогенератора)
   781                          ; Параметр       AL = 0 сообщает, что будет
заменено изображение
   782                          ; символа для текущего шрифта
   783                          ; В случаях, когда      AL = 1 или 2, будет
заменено изображение
   784                          ; только для опредленного шрифта (8x14 и
8x8 соответственно)
   785                          ; Параметр       BH = 0Eh сообщает, что на
опредление каждого изображения символа
   786                          ; расходуется по 14 байт (режим 8x14 бит как
раз 14 байт)
   787                          ; Параметр       BL = 0 - блок шрифта для
загрузки (от 0       до 4)
   788                          ;
   789                          ; *** результат
   790                          ; изображение указанного(ых) символа(ов)
будет заменено
   791                          ; на предложенное пользователем.
   792                          ; Изменению подвергнутся все символы,
находящиеся на экране,
   793                          ; то есть если изображение заменено, старый
вариант нигде уже не проявится
   794
   7950769                      changeFont      proc
   7960769  50                          push AX
   797076A  53                          push BX
   798076B  B8 1100                             mov AX, 1100h
```

```
799076E  BB 1000                mov BX, 1000h
8000771  CD 10                  int 10h
8010773  58                     pop AX
8020774  5B                     pop BX
8030775  C3                     ret
8040776                 changeFont      endp
805
806                     ;=== Функция сохранения нормального
начертания символа
807                     ;===
808                     ; *** входные данные
809                     ; BH - тип      возвращаемой символьной
    таблицы
810                     ;    0 - таблица из     int 1fh
811                     ;    1 - таблица из     int 44h
812                     ;    2-5 - таблица из 8x14,   8x8, 8x8 (top),
    9x14
813                     ;    6 - 8x16
814                     ;
815                     ; *** описание работы процедуры
816                     ; Происходит вызов  int 10h
    (видеосервис)
817                     ; с функцией AH = 11h (функции
знакогенератора)
818                     ; Параметр      AL = 30      - подфункция
получения информации о EGA
819                     ;
820                     ; *** результат
821                     ; в ES:BP находится таблица символов
(полная)
822                     ; в CX находится байт на символ
823                     ; в DL количество экранных      строк
824                     ; ВАЖНО! Происходит сдвиг регистра ES
825                     ; ( ES становится равным C000h )
826
8270776                 saveFont proc
8280776  50                     push AX
8290777  53                     push BX
8300778  B8 1130                mov AX, 1130h
```

```
   831077B  BB 0600                    mov BX, 0600h
   832077E  CD 10                      int 10h
   8330780  58                    pop AX
   8340781  5B                    pop BX
   8350782  C3                    ret
   8360783                   saveFont endp
   837
   838
   839                       ;=== Отсюда начинается выполнение
основной    части программы ===;
   840                       ;===
   8410783                   _initTSR:                          ; старт
резидента
   8420783  B4 03                 mov AH, 03h
   8430785  CD 10                 int 10h
   8440787  52                   push DX
   8450788  B4 00                mov AH,00h
; установка видеорежима (83h  текст   +
   846                       80x25  16/8  CGA,EGA  b800
Comp,RGB,Enhanced), без очистки экрана
   847078A  B0 83                 mov AL,83h
   848078C  CD 10                      int 10h
   849078E  5A                    pop DX
   850078F  B4 02                 mov AH, 02h
   8510791  CD 10                 int 10h
   852
   853
   8540793  E8 008C                  call commandParamsParser
   8550796  B8 3509                    mov AX,3509h            ;
получить    в ES:BX      вектор 09
```

```
   8560799  CD 21                int 21h                         ; прерывания
   857
   858
   859079B  80 3E 017Fr FF                cmp notLoadTSR, true
; если была выведена справка
   86007A0  74 0E                    je  _exit_tmp
; просто выходим
   861
   862                               ;@ Если по       варианту необходимо
выгружать резидент по повторному запуску, то     +
   863                                    комментируем 5 строк ниже
   864                               ;@ если необходимо  выгружать по
параметру коммандной строки, то оставляем их
   86507A2  B4 FF                         mov AH, 0FFh
   86607A4  B0 00                    mov AL, 0
   86707A6  CD 2F                         int 2Fh
   86807A8  80 FC 69                      cmp AH, 'i'  ; проверка того,
загружена ли        уже программа
   86907AB  74 62                    je  _alreadyInstalled
   870
   87107AD  EB 04 90                      jmp _tmp
   872
   87307B0                           _exit_tmp:
   87407B0  EB 6E 90                           jmp _exit
   875
   87607B3                           _tmp:
   87707B3  06                       push ES
   87807B4  A1 002C                       mov AX, DS:[2Ch]                ; psp
   87907B7  8E C0                         mov ES, AX
   88007B9  B4 49                    mov AH, 49h                     ; хватит
памяти чтоб остаться
   88107BB  CD 21                         int 21h                         ;
резидентом?
   88207BD  07                       pop ES
   88307BE  72 59                    jc _notMem                 ; не хватило -
        выходим
   884
   885                                ;== int 09h ==;
   886
```

```
   88707C0  2E: 89 1E    0173r            mov     word ptr
CS:old_int9hOffset, BX
   88807C5  2E: 8C 06    0175r            mov     word ptr
CS:old_int9hSegment, ES
   88907CA  B8 2509                  mov AX, 2509h                  ;
установим вектор на 09
   89007CD  BA 051Br                 mov DX, offset     new_int9h       ;
прерывание
   89107D0  CD 21                    int 21h
   892
   893                            ;== int 1Ch ==;
   89407D2  B8 351C                  mov AX,351Ch             ;
получить     в ES:BX      вектор 1C
   89507D5  CD 21                    int 21h                  ;
прерывания
   89607D7  2E: 89 1E    0177r            mov     word ptr
CS:old_int1ChOffset, BX
   89707DC  2E: 8C 06    0179r            mov     word ptr
CS:old_int1ChSegment, ES
   89807E1  B8 251C                  mov AX, 251Ch                  ;
установим вектор на 1C
   89907E4  BA 05D7r                 mov DX, offset new_int1Ch
; прерывание
   90007E7  CD 21                    int 21h
   901
   902                            ;== int 2Fh ==;
   90307E9  B8 352F                  mov AX,352Fh             ;
получить     в ES:BX      вектор 1C
   90407EC  CD 21                    int 21h                  ;
прерывания
   90507EE  2E: 89 1E    017Br            mov     word ptr
CS:old_int2FhOffset, BX
   90607F3  2E: 8C 06    017Dr            mov     word ptr
CS:old_int2FhSegment, ES
   90707F8  B8 252F                  mov AX, 252Fh                  ;
установим вектор на 2F
   90807FB  BA 0605r                 mov DX, offset new_int2Fh
; прерывание
   90907FE  CD 21                    int 21h
   910
   9110800  E8 FC76                  call changeFx
   9120803  BA 03E8r                 mov DX, offset     installedMsg
; выводим что все ок
```

```
0130806  B4 09                mov AH, 9
0140808  CD 21                int 21h
015080A  BA 0783r                   mov DX, offset _initTSR      ;
остаемся в памяти резидентом
016080D  CD 27                      int 27h                      ; и выходим
017                           ; конец основной программы
018080F                     _alreadyInstalled:
019080F  B4 09                  mov AH, 09h
0200811  BA 03FBr                     lea DX, alreadyInstalledMsg
0210814  CD 21                  int 21h
0220816  EB 08 90                     jmp _exit
0230819                     _notMem:                             ; не хватает
памяти, чтобы остаться резидентом
0240819  BA 0411r                   mov DX, offset noMemMsg
025081C  B4 09                  mov AH, 9
026081E  CD 21                      int 21h
0270820                     _exit:                               ; выход
0280820  CD 20                  int 20h
029
030                         ;=== Процедура проверки параметров  ком.
строки ===;
031                         ;===
0320822                     commandParamsParser proc
0330822  0E                       push CS
0340823  07                       pop ES
0350824  C6 06 017Fr 00                mov notLoadTSR, 0
036
0370829  BE 0080                      mov SI, 80h
;SI=смещение командной строки.
038082C  AC                       lodsb
;Получим кол-во символов.
039082D  0A C0                        or  AL, AL
;Если 0 символов введено,
040082F  74 3A                    jz  _exitHelp                   ;то все в
порядке.
041
0420831                           _nextChar:
043
```

```
   9440831  46                      inc SI
;Теперь SI   указывает на первый символ  +
   945                         строки.
   946
   9470832  80 3C 0D                   cmp [SI], BYTE ptr    13
   9480835  74 34                  je  _exitHelp
   949
   950
   9510837  AD                        lodsw
;Получаем два символа
   9520838  3D 3F2F                        cmp AX, '?/'
;Это '/?' (данные расположены в          +
   953                         обратном порядк, т.е. AL:AH вместо
       AH:AL)
   954083B  74 03                     je  _question
   955
   956                                ;cmp AH, '/'
   957                                ;je _errorParam
   958
   959083D  EB 2C 90                        jmp _exitHelp
   960
   9610840                         _question:
   962                                ; вывод строки помощи
   9630840  B4 03                           mov AH,03
   9640842  CD 10                           int 10h
   9650844  BD 021Er                              lea BP, helpMsg
   9660847  B9 013D                           mov CX,
helpMsg_length
   967084A  B3 07                     mov BL, 0111b
   968084C  B8 1301                         mov AX, 1301h
   969084F  CD 10                           int 10h
```

```
970                                     ; конец вывода строки помощи
9710851  F6 16 017Fr                    not notLoadTSR          ;флаг
того, что необходимо    не загружать резидент
9720855  EB DA                              jmp _nextChar
973
9740857  EB 12 90                       jmp _exitHelp
975
976085A                             _errorParam:
977                                     ;вывод строки
978085A  B4 03                              mov AH,03
979085C  CD 10                                  int 10h
980085E  BD 035Br                               lea BP,
CS:errorParamMsg
9810861  B9 0025                                mov CX,
errorParamMsg_length
9820864  B3 07                              mov BL, 0111b
9830866  B8 1301                                mov AX, 1301h
9840869  CD 10                              int 10h
985                                     ;конец вывода строки
986086B                             _exitHelp:
987086B  C3                         ret
988086C                         commandParamsParser endp
989
990086C                         code ends
991                             end _start
```

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 19
Symbol Table

Symbol Name	Type	Value
??DATE	Text	"04/29/21"
??FILENAME	Text	"tsr2 "
??TIME	Text	"22:19:48"
??VERSION	Number	030A
@CPU	Text	0101H
@CURSEG	Text	CODE
@FILENAME	Text	TSR2
@WORDSIZE	Text	2
ALREADYINSTALLEDMSG	Byte	CODE:03FB
CHANGEFONT	Near	CODE:0769
CHANGEFX	Near	CODE:0479
CHARTOCURSIVEINDEX	Byte	CODE:0162
COMMANDPARAMSPARSER	Near	CODE:0822
COUNTER	Word	CODE:0180
CURSIVEENABLED	Byte	CODE:0151
CURSIVESYMBOL	Byte	CODE:0152
ERRORPARAMMSG	Byte	CODE:035B
ERRORPARAMMSG_LENGTH	Number	0025
F1_TXT	Byte	CODE:0471
F2_TXT	Byte	CODE:0473
F3_TXT	Byte	CODE:0475
F4_TXT	Byte	CODE:0477
FX_LENGTH	Number	0002
HELPMSG	Byte	CODE:021E
HELPMSG_LENGTH	Number	013D
IGNOREDCHARS	Byte	CODE:0103
IGNOREDLENGTH	Number	0041
IGNOREENABLED	Byte	CODE:0144
INSTALLEDMSG	Byte	CODE:03E8
LINE1_LENGTH	Number	0034
LINE2_LENGTH	Number	0034
LINE3_LENGTH	Number	0034
NEW_INT1CH	Far	CODE:05D7
NEW_INT2FH	Near	CODE:0605
NEW_INT9H	Far	CODE:051B
NOMEMMSG	Byte	CODE:0411

NOREMOVEMSG Byte CODE:0454
NOREMOVEMSG_LENGTH Number 001D
NOTINSTALLEDMSG Byte CODE:0425
NOTLOADTSR Byte CODE:017F
OLD_INT1CHOFFSET Word CODE:0177
OLD_INT1CHSEGMENT Word CODE:0179
OLD_INT2FHOFFSET Word CODE:017B
OLD_INT2FHSEGMENT Word CODE:017D
OLD_INT9HOFFSET Word CODE:0173
OLD_INT9HSEGMENT Word CODE:0175
PRINTDELAY Number 0002
PRINTSIGNATURE Near CODE:0692
REMOVEDMSG Byte CODE:0443
REMOVEDMSG_LENGTH Number 0011
REPLACEWITH Byte CODE:0124
SAVEDSYMBOL Byte CODE:0163
SAVEFONT Near CODE:0776
SETCURSIVE Near CODE:071C

Symbol Table

SIGNATURELINE1 Byte CODE:0182
SIGNATURELINE2 Byte CODE:01B6
SIGNATURELINE3 Byte CODE:01EA
SIGNATUREPRINTINGENABLED Byte CODE:0150
TABLEBOTTOM Byte CODE:03B4
TABLEBOTTOM_LENGTH Number 0034
TABLETOP Byte CODE:0380
TABLETOP_LENGTH Number 0034
TRANSLATEENABLED Byte CODE:014F
TRANSLATEFROM Byte CODE:0145
TRANSLATELENGTH Number 0005
TRANSLATETO Byte CODE:014A
TRUE Number 00FF
_2FH_EXIT Near CODE:068D
_2FH_STD Near CODE:0615
_ACTUALPRINT Near CODE:06AD
_ALREADYINSTALLED Near CODE:080F
_ALREADY_INSTALLED Near CODE:061A
_BLOCK Near CODE:059E
_CHECKF1 Near CODE:0488
_CHECKF2 Near CODE:04AC
_CHECKF3 Near CODE:04CD
_CHECKF4 Near CODE:04EE
_CHECK_IGNORED Near CODE:0592
_CHECK_TRANSLATE Near CODE:05AD
_CHECK_TRANSLATE_LOOP Near CODE:05BA
_DONTPRINT Near CODE:05FE
_ERRORPARAM Near CODE:085A
_EXIT Near CODE:0820
_EXITHELP Near CODE:086B
_EXITSETCURSIVE Near CODE:0766
_EXIT_TMP Near CODE:07B0
_F1 Near CODE:052D
_F2 Near CODE:053B
_F3 Near CODE:054C
_F4 Near CODE:055A
_GO Near CODE:0582
_GREENF1 Near CODE:04A5
_GREENF2 Near CODE:04C9

```
_GREENF3                Near CODE:04EA
_GREENF4                Near CODE:050B
_INITTSR                Near CODE:0783
_LETSPRINT                  Near CODE:05F1
_NEXTCHAR                   Near CODE:0831
_NOTMEM                     Near CODE:0819
_NOTREMOVE                  Near CODE:0668
_NOTTOPRINT                 Near CODE:0603
_OUTFX                      Near CODE:050F
_PRINTBOTTOM                Near CODE:06A6
_QUESTION                   Near CODE:0840
_QUIT                       Near CODE:05CF
_REDF1                      Near CODE:049E
_REDF2                      Near CODE:04C2
_REDF3                      Near CODE:04E3
_REDF4                      Near CODE:0504
_RESTORESYMBOL                  Near CODE:0757
_SHIFTTABLE                 Near CODE:072E
```

Symbol Table

_START	Near	CODE:0100
_TEST_FX	Near	CODE:052B
_TMP	Near	CODE:07B3
_TRANSLATE	Near	CODE:05C6
_TRANSLATE_OR_IGNORE	Near	CODE:0568
_UNINSTALL	Near	CODE:061D
_UNLOADED	Near	CODE:067C

Groups & Segments	Bit	Size	Align	Combine	Class
CODE	16	086C	Para	none	CODE

2. Файл unloader.lst

```
Turbo Assembler    Version 3.1     05/19/23 14:32:48          Page 1
unldtsr.ASM



    1
    2
    3  0000                    code segment    'code'
    4                                  assume  CS:code, DS:code
    5                                  org     100h
    6  0100                            _start:
    7
    8  0100  B4 FF                     mov AH, 0FFh
    9  0102  B0 01                     mov AL, 1
   10  0104  CD 2F                           int 2Fh
   11  0106  CD 20                     int 20h
   12
   13  0108                    code ends
   14                          end _start
Turbo Assembler    Version 3.1     05/19/23 14:32:48          Page 2
Symbol Table



Symbol Name                     Type Value

??DATE                          Text  "05/19/23"
??FILENAME                      Text  "unldtsr "
??TIME                          Text  "14:28:23"
??VERSION                       Number 030A
@CPU                            Text  0101H
@CURSEG                         Text  CODE
@FILENAME                       Text  UNLDTSR
@WORDSIZE                       Text  2
_START                          Near  CODE:0100

Groups & Segments               Bit Size Align  Combine Class

CODE                            16  0108 Para       none   CODE
```