Московский государственный технический университет
им. Н.Э. Баумана

УТВЕРЖДАЮ:

Большаков С.А.
"__"_____2024 г.

Курсовая работа по курсу «Системное программирование»

Исходный текст программного продукта
(вид документа)

писчая бумага
(вид носителя)

18
(количество листов)

ИСПОЛНИТЕЛИ:

студенты группы ИУ5-41б _____
Ларкин Б. В.
"__"_____2024 г.

Москва – 2024

Содержание

1. Файл tsr.lst

```
Turbo Assembler  Version 3.1         05/07/24 18:38:25
tsr.asm

      1         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      2         ; tsr.asm - заголовок
      3         ;
      4         ; Сборка:
      5         ; tasm.exe /l tsr.asm
      6         ; tlink /t /x tsr.obj
      7         ;
      8         ; Примечания:
      9         ; 1) комментарии, начинающиеся с символа @ - места, где код зависит от варианта
     10         ;
     11         ; Авторы:
     12         ; МГТУ им. Н.Э. Баумана, ИУ5-44, 2013 г.
     13         ;  Леонтьев А.В.
     14         ;  Латкин И.И.
     15         ;  Назаров К.В.
     16         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     17         ; Резидентная часть
     18 0000        code segment    'code'
     19          assume  CS:code, DS:code
     20          org     100h
     21 0100        _start:
     22
     23 0100  E9 0606        jmp _initTSR ; на начало программы
     24
     25          ; данные резидента
     26          ; Игнорирование
     27
     28 0103  41 42 43 44 45 46 47+      ignoredChars         DB         +
     29       48 49 4A 4B 4C 4D 4E+  'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'  ;список
игнорируемых символов
     30       4F 50 51 52 53 54 55+
     31       56 57 58 59 5A 61 62+
     32       63 64 65 66 67 68 69+
     33       6A 6B 6C 6D 6E 6F 70+
     34       71 72 73 74 75 76 77+
     35       78 79 7A
     36       =0034        ignoredLength        equ     $-ignoredChars       +
     37         ; длина строки ignoredChars
     38 0137  00        ignoreEnabled        DB      0         +
     39         ; флаг функции игнорирования ввода
     40 0138  4B 56 59 4A 47       translateFrom        DB      'KVYJG'        +
     41         ;@ символы для замены (ЛМНОП на англ. раскладке)
     42 013D  8B 8C 8D 8E 8F       translateTo        DB      'ЛМНОП'        +
     43         ;@ символы на которые будет идти замена
     44       =0005        translateLength        equ     $-translateTo       +
     45         ; длина строки trasnlateFrom
     46 0142  00        translateEnabled        DB      0        +
     47         ; флаг функции перевода
     48
     49 0143  00        signaturePrintingEnabled       DB      0         +
     50         ; флаг функции вывода информации об авторе
     51 0144  00        cursiveEnabled        DB      0         +
     52         ; флаг перевода символа в курсив
     53 0145  00        cursiveSymbol        DB 00000000b    ;@ символ И, составленный+
     54         из единичек (его курсивный вариант)
     55 0146  00              DB 00000000b
     56 0147  63              DB 01100011b
     57 0148  67              DB 01100111b
     58 0149  67              DB 01100111b
     59 014A  6B              DB 01101011b
     60 014B  6B              DB 01101011b
     61 014C  CE              DB 11001110b
     62 014D  D6              DB 11010110b
     63 014E  D6              DB 11010110b
     64 014F  E6              DB 11100110b
     65 0150  E6              DB 11100110b
     66 0151  C6              DB 11000110b
     67 0152  00              DB 00000000b
     68 0153  00              DB 00000000b
     69 0154  00              DB 00000000b
     70
     71 0155  88        charToCursiveIndex        DB 'И'         +
```

```
     72          ;@ символ для замены
     73 0156  10*(FF)        savedSymbol         DB 16 dup(0FFh)
     74          ; переменная для хранения старого символа
     75
     76       =00FF        true          equ     0FFh       +
     77          ; константа истинности
     78 0166  ????        old_int9hOffset         DW      ?        +
     79          ; адрес старого обработчика int 9h
     80 0168  ????        old_int9hSegment         DW      ?        +
     81          ; сегмент старого обработчика int 9h
     82 016A  ????        old_int1ChOffset         DW      ?        +
     83          ; адрес старого обработчика int 1Ch
     84 016C  ????        old_int1ChSegment        DW      ?         +
     85          ; сегмент старого обработчика int 1Ch
     86 016E  ????        old_int2FhOffset         DW      ?        +
     87          ; адрес старого обработчика int 2Fh
     88 0170  ????        old_int2FhSegment        DW      ?         +
     89          ; сегмент старого обработчика int 2Fh
     90
     91 0172  00        unloadTSR          DB      0        +
     92          ; 1 - выгрузить резидент
     93 0173  00        notLoadTSR          DB      0        +
     94          ; 1 - не загружать
     95 0174  0000        counter          DW      0
     96       =0007        printDelay          equ     7        +
     97          ;@ задержка перед выводом "подписи" в секундах
     98 0176  0001        printPos           DW      1       +
     99          ;@ положение подписи на экране. 0 - верх, 1 - центр, 2 - низ
    100
    101          ;@ заменить на собственные данные. формирование таблицы идет по строке большей длины   +
    102          (1я строка).
    103 0178  B3 8B A0 E0 AA A8 AD+      signatureLine1        DB      179, 'Ларкин Борис', 179
    104       20 81 AE E0 A8 E1 B3
    105
    106       =000E        Line1_length         equ     $-signatureLine1
    107 0186  B3 88 93 35 2D 34 31+      signatureLine2        DB      179, 'ИУ5-41      ', 179
    108       20 20 20 20 20 20 B3
    109       =000E        Line2_length         equ     $-signatureLine2
    110 0194  B3 82 A0 E0 A8 A0 AD+      signatureLine3        DB      179, 'Вариант #11 ', 179
    111       E2 20 23 31 31 20 B3
    112       =000E        Line3_length         equ     $-signatureLine3
    113          ; Справка
    114 01A2  3E 74 73 72 2E 63  6F+     helpMsg DB      '>tsr.com [/?]', 10, 13
    115       6D 20 5B 2F 3F 5D 20+
    116       5B 2F 75 5D 0A 0D
    117 01B6  20 5B 2F 3F 5D 20 2D+        DB ' [/?] - вывод данной справки', 10, 13
    118       20 A2 EB A2 AE A4 20+
    119       A4 A0 AD AD AE A9 20+
    120       E1 AF E0 A0 A2 AA A8+
    121       0A 0D
    122
    123 01D4  20 20 46 31 20 20 2D+        DB '  F1  - включение и отключения курсивного вывода русского
символа  +
    124       20 A2 AA AB EE E7 A5+  И', 10, 13
    125       AD A8 A5 20 A8 20 AE+
    126       E2 AA AB EE E7 A5 AD+
    127       A8 EF 20 AA E3 E0 E1+
    128       A8 A2 AD AE A3 AE 20+
    129       A2 EB A2 AE A4 A0 20+
    130       E0 E3 E1 E1 AA AE A3+
    131       AE 20 E1 A8 AC A2 AE+
    132       AB A0 20 88 0A 0D
    133 0219  20 20 46 32 20 20 2D+        DB '  F2  - включение и отключение частичной русификации
клавиатуры    +
    134       20 A2 AA AB EE E7 A5+  (KVYJG -> ЛМНОП)', 10, 13
    135       AD A8 A5 20 A8 20 AE+
    136       E2 AA AB EE E7 A5 AD+
    137       A8 A5 20 E7 A0 E1 E2+
    138       A8 E7 AD AE A9 20 E0+
    139       E3 E1 A8 E4 A8 AA A0+
    140       E6 A8 A8 20 AA AB A0+
    141       A2 A8 A0 E2 E3 E0 EB+
    142       28 4B 56 59 4A 47 20+
    143       2D 3E 20 8B 8C 8D 8E+
    144       8F 29 0A 0D
    145 026A  20 20 46 33 20 20 2D+ DB '  F3  - включение и отключение режима блокировки ввода латинских   +
    146       20 A2 AA AB EE E7 A5+  букв', 10, 13
```

```
147       AD A8 A5 20 A8 20 AE+
148       E2 AA AB EE E7 A5 AD+
149       A8 A5 20 E0 A5 A6 A8+
150       AC A0 20 A1 AB AE AA+
151       A8 E0 AE A2 AA A8 20+
152       A2 A2 AE A4 A0 20 AB+
153       A0 E2 A8 AD E1 AA A8+
154       E5 20 A1 E3 AA A2 0A+
155       0D
156 02B1  20 20 46 39 20 20 2D+        DB ' F9  - вывод ФИО и группы по таймеру в центре экрана', 10, 13
157       20 A2 EB A2 AE A4 20+
158       94 88 8E 20 A8 20 A3+
159       E0 E3 AF AF EB 20 AF+
160       AE 20 E2 A0 A9 AC A5+
161       E0 E3 20 A2 20 E6 A5+
162       AD E2 E0 A5 20 ED AA+
163       E0 A0 AD A0 0A 0D
164
165       =0146        helpMsg_length        equ  $-helpMsg
166 02E8  8E E8 A8 A1 AA A0 20+      errorParamMsg         DB      'Ошибка параметров коммандной  +
167       AF A0 E0 A0 AC A5 E2+  строки', 10, 13
168       E0 AE A2 20 AA AE AC+
169       AC A0 AD A4 AD AE A9+
170       20 E1 E2 E0 AE AA A8+
171       0A 0D
172       =0025        errorParamMsg_length       equ     $-errorParamMsg
173
174 030D  DA 0C*(C4) BF       tableTop           DB      218, Line1_length-2 dup+
175         (196), 191
176       =000E        tableTop_length         equ     $-tableTop
177 031B  C0 0C*(C4) D9       tableBottom         DB      192, Line1_length-2 dup (196), +
178         217
179       =000E        tableBottom_length        equ  $-tableBottom
180
181          ; сообщения
182 0329  90 A5 A7 A8 A4 A5 AD+      installedMsg         DB  'Резидент загружен!$'
183       E2 20 A7 A0 A3 E0 E3+
184       A6 A5 AD 21 24
185 033C  90 A5 A7 A8 A4 A5 AD+      alreadyInstalledMsg       DB  'Резидент уже загружен$'
186       E2 20 E3 A6 A5 20 A7+
187       A0 A3 E0 E3 A6 A5 AD+
188       24
189 0352  8D A5 A4 AE E1 E2 A0+      noMemMsg           DB  'Недостаточно памяти$'
190       E2 AE E7 AD AE 20 AF+
191       A0 AC EF E2 A8 24
192 0366  8D A5 20 E3 A4 A0 AB+      notInstalledMsg        DB  'Не удалось загрузить резидент$'
193       AE E1 EC 20 A7 A0 A3+
194       E0 E3 A7 A8 E2 EC 20+
195       E0 A5 A7 A8 A4 A5 AD+
196       E2 24
197
198 0384  90 A5 A7 A8 A4 A5 AD+      removedMsg          DB  'Резидент выгружен'
199       E2 20 A2 EB A3 E0 E3+
200       A6 A5 AD
201       =0011        removedMsg_length         equ     $-removedMsg
202
203 0395  8D A5 20 E3 A4 A0 AB+      noRemoveMsg         DB  'Не удалось выгрузить резидент'
204       AE E1 EC 20 A2 EB A3+
205       E0 E3 A7 A8 E2 EC 20+
206       E0 A5 A7 A8 A4 A5 AD+
207       E2
208       =001D        noRemoveMsg_length        equ     $-noRemoveMsg
209
210 03B2  46 31        f1_txt          DB      'F1'
211 03B4  46 32        f2_txt          DB      'F2'
212 03B6  46 33        f3_txt          DB      'F3'
213 03B8  46 39        f9_txt          DB      'F9'
214       =0002        fx_length          equ     $-f9_txt
215          ; Проверка клавиш
216 03BA         changeFx proc
217 03BA  50         push AX
218 03BB  53         push BX
219 03BC  51         push CX
220 03BD  52         push DX
221 03BE  55         push BP
222 03BF  06         push ES
223 03C0  33 DB         xor BX, BX
```

```
224
225 03C2  B4 03         mov AH, 03h
226 03C4  CD 10         int 10h
227 03C6  52         push DX
228
229 03C7  0E         push CS
230 03C8  07         pop ES
231
232 03C9         _checkF1:
233 03C9  BD 03B2r         lea BP, f1_txt
234 03CC  B9 0002         mov CX, fx_length
235 03CF  B7 00         mov BH, 0
236 03D1  B6 00         mov DH, 0
237 03D3  B2 4E         mov DL, 78
238 03D5  B8 1301         mov AX, 1301h
239
240 03D8  80 3E 0144r FF        cmp cursiveEnabled, true
241 03DD  74 07         je _greenF1
242
243 03DF          _redF1:
244 03DF  B3 4F          mov BL, 01001111b ; red
245 03E1  CD 10          int 10h
246 03E3  EB 08 90          jmp _checkF2
247
248 03E6          _greenF1:
249 03E6  BD 03B2r          lea BP, f1_txt
250 03E9  B3 2F          mov BL, 00101111b ; green
251 03EB  CD 10          int 10h
252
253 03ED         _checkF2:
254 03ED  BD 03B4r         lea BP, f2_txt
255 03F0  B9 0002         mov CX, fx_length
256 03F3  B7 00         mov BH, 0
257 03F5  B6 01         mov DH, 1
258 03F7  B2 4E         mov DL, 78
259 03F9  B8 1301         mov AX, 1301h
260
261 03FC  80 3E 0142r FF        cmp translateEnabled, true
262 0401  74 07         je _greenF2
263
264 0403          _redF2:
265 0403  B3 4F          mov BL, 01001111b ; red
266 0405  CD 10          int 10h
267 0407  EB 05 90          jmp _checkF3
268
269 040A          _greenF2:
270 040A  B3 2F          mov BL, 00101111b ; green
271 040C  CD 10          int 10h
272
273 040E         _checkF3:
274 040E  BD 03B6r         lea BP, f3_txt
275 0411  B9 0002         mov CX, fx_length
276 0414  B7 00         mov BH, 0
277 0416  B6 02         mov DH, 2
278 0418  B2 4E         mov DL, 78
279 041A  B8 1301         mov AX, 1301h
280
281 041D  80 3E 0137r FF        cmp ignoreEnabled, true
282 0422  74 07         je _greenF3
283
284 0424          _redF3:
285 0424  B3 4F          mov BL, 01001111b ; red
286 0426  CD 10          int 10h
287 0428  EB 05 90          jmp _checkf9
288
289 042B          _greenF3:
290 042B  B3 2F          mov BL, 00101111b ; green
291 042D  CD 10          int 10h
292
293 042F         _checkf9:
294 042F  BD 03B8r         lea BP, f9_txt
295 0432  B9 0002         mov CX, fx_length
296 0435  B7 00         mov BH, 0
297 0437  B6 03         mov DH, 3
298 0439  B2 4E         mov DL, 78
299 043B  B8 1301         mov AX, 1301h
300
```

```
301 043E  80 3E 0143r FF        cmp signaturePrintingEnabled, true
302 0443  74 07         je _greenf9
303
304 0445          _redf9:
305 0445  B3 4F          mov BL, 01001111b ; red
306 0447  CD 10          int 10h
307 0449  EB 05 90          jmp _outFx
308
309 044C          _greenf9:
310 044C  B3 2F          mov BL, 00101111b ; green
311 044E  CD 10          int 10h
312
313 0450         _outFx:
314 0450  5A          pop DX
315 0451  B4 02         mov AH, 02h
316 0453  CD 10         int 10h
317
318 0455  07         pop ES
319 0456  5D         pop BP
320 0457  5A         pop DX
321 0458  59         pop CX
322 0459  5B         pop BX
323 045A  58         pop AX
324 045B  C3         ret
325 045C         changeFx endp
326           ; новый обработчик  new_int9h
327
328      ;новый обработчик
329 045C     new_int9h proc far
330           ; сохраняем значения всех, изменяемых регистров в стэке
331 045C  56         push SI
332 045D  50         push AX
333 045E  53         push BX
334 045F  51         push CX
335 0460  52         push DX
336 0461  06         push ES
337 0462  1E         push DS
338           ; синхронизируем CS и DS
339 0463  0E         push CS
340 0464  1F         pop     DS
341
342 0465  B8 0040         mov     AX, 40h;40h-сегмент,где хранятся флаги сост-я клавиатуры, кольц.     +
343         буфер ввода
344 0468  8E C0         mov     ES, AX
345 046A  E4 60         in      AL, 60h ; записываем в AL скан-код нажатой клавиши
346
347           ;@ проверка на Ctrl+U, только для ИУ5-41
348 046C  3C 16         cmp     AL, 22  ; была нажата клавиша U?
349 046E  75 24         jne     _test_Fx
350 0470  26: 8A 26 0017        mov     AH, ES:[17h]     ; флаги клавиатуры
351 0475  80 E4 0F         and     AH, 00001111b
352 0478  80 FC 04         cmp     AH, 00000100b   ; был ли нажат ctrl?
353 047B  75 17         jne     _test_Fx
354           ; выгрузка
355 047D  B4 FF          mov AH, 0FFh
356 047F  B0 01          mov AL, 01h
357 0481  CD 2F          int 2Fh
358            ; завершаем обработку нажатия
359            ; Работа с портом в/в
360
361 0483  E4 61          in      AL, 61h ;контроллер состояния клавиатуры
362 0485  0C 80          or      AL, 10000000b   ;пометим, что клавишу нажали
363 0487  E6 61          out     61h, AL
364 0489  24 7F          and     AL, 01111111b   ;пометим, что клавишу отпустили
365 048B  E6 61          out     61h, AL
366 048D  B0 20          mov     AL, 20h
367 048F  E6 20          out     20h, AL ;отправим в контроллер прерываний признак конца     +
368         прерывания
369
370            ; выходим
371 0491  E9 009D          jmp _quit
372
373
374           ;проверка F1-f9
375 0494          _test_Fx:
376 0494  2C 3A         sub AL, 58 ; в AL теперь номер функциональной клавиши
377 0496          _F9: ;signaturePrint
```

```
    378 0496  3C 09          cmp AL, 9 ; F9
    379 0498  75 0A          jne _F1
    380 049A  F6 16 0143r         not signaturePrintingEnabled
    381 049E  E8 FF19          call changeFx
    382 04A1  EB 2E 90          jmp _translate_or_ignore
    383 04A4          _F1: ;Cursive
    384 04A4  3C 01          cmp AL, 1 ; F1
    385 04A6  75 0D          jne _F2
    386 04A8  F6 16 0144r         not cursiveEnabled
    387 04AC  E8 FF0B          call changeFx
    388 04AF  E8 01F0          call setCursive ; перевод символа в курсив и обратно в зависимости от  +
    389         флага cursiveEnabled
    390 04B2  EB 1D 90          jmp _translate_or_ignore
    391 04B5          _F2: ;translate
    392 04B5  3C 02          cmp AL, 2 ; F2
    393 04B7  75 0A          jne _F3
    394 04B9  F6 16 0142r         not translateEnabled
    395 04BD  E8 FEFA          call changeFx
    396 04C0  EB 0F 90          jmp _translate_or_ignore
    397 04C3          _F3: ;ignore
    398 04C3  3C 03          cmp AL, 3 ; F3
    399 04C5  75 0A          jne _translate_or_ignore
    400 04C7  F6 16 0137r         not ignoreEnabled
    401 04CB  E8 FEEC          call changeFx
    402 04CE  EB 01 90          jmp _translate_or_ignore
    403
    404           ;игнорирование и перевод
    405 04D1          _translate_or_ignore:
    406
    407         ; Вызов старого обработчика  old_int9hOffset
    408
    409 04D1  9C          Pushf
    410
    411 04D2  2E: FF 1E 0166r        call dword ptr CS:[old_int9hOffset] ; вызываем стандартный обработчик
прерывания
    412 04D7  B8 0040         mov     AX, 40h      ; 40h-сегмент,где хранятся флаги сост-я клавы,кольц.   +
    413         буфер ввода
    414         ; Работа с клавиатурой
    415
    416 04DA  8E C0          mov     ES, AX
    417 04DC  26: 8B 1E 001C        mov     BX, ES:[1Ch]    ; адрес хвоста
    418 04E1  4B          dec     BX      ; сместимся назад к последнему
    419 04E2  4B          dec     BX      ; введённому символу
    420 04E3  83 FB 1E          cmp     BX, 1Eh ; не вышли ли мы за пределы буфера?
    421 04E6  73 03          jae     _go
    422 04E8  BB 003C          mov     BX, 3Ch;хвост вышел за пределы буфера, значит последний введённый    +
    423         символ
    424          ; находится в конце буфера
    425
    426 04EB          _go:
    427 04EB  26: 8B 17          mov DX, ES:[BX] ; в DX 0 введённый символ
    428           ;включен ли режим блокировки ввода?
    429 04EE  80 3E 0137r FF        cmp ignoreEnabled, true
    430 04F3  75 1A          jne _check_translate
    431           ; Блокировка ввода символов
    432
    433           ; да, включен
    434 04F5  BE 0000          mov SI, 0
    435 04F8  B9 0034          mov CX, ignoredLength ;кол-во игнорируемых символов
    436
    437           ; проверяем, присутствует ли текущий символ в списке игнорируемых
    438 04FB          _check_ignored:
    439 04FB  3A 94 0103r        cmp DL,ignoredChars[SI]
    440 04FF  74 06          je _block
    441 0501  46          inc SI
    442 0502  E2 F7        loop _check_ignored
    443 0504  EB 09 90          jmp _check_translate
    444
    445          ; блокируем
    446 0507          _block:
    447 0507  26: 89 1E 001C        mov ES:[1Ch], BX ;блокировка ввода символа
    448 050C  EB 23 90          jmp _quit
    449            ; Замена символов
    450
    451 050F          _check_translate:
    452           ; включен ли режим перевода?
    453 050F  80 3E 0142r FF        cmp translateEnabled, true
```

```
    454 0514  75 1B         jne _quit
    455
    456           ; да, включен
    457 0516  BE 0000         mov SI, 0
    458 0519  B9 0005         mov CX, translateLength ; кол-во символов для перевода
    459           ; проверяем, присутствует ли текущий символ в списке для перевода
    460 051C          _check_translate_loop:
    461 051C  3A 94 0138r         cmp DL, translateFrom[SI]
    462 0520  74 06          je _translate
    463 0522  46          inc SI
    464 0523  E2 F7         loop _check_translate_loop
    465 0525  EB 0A 90         jmp _quit
    466
    467           ; переводим
    468 0528          _translate:
    469 0528  33 C0          xor AX, AX
    470 052A  8A 84 013Dr         mov AL, translateTo[SI]
    471 052E  26: 89 07          mov ES:[BX], AX ; замена символа
    472
    473 0531         _quit:
    474           ; восстанавливаем все регистры
    475 0531  1F         pop     DS
    476 0532  07         pop     ES
    477 0533  5A         pop DX
    478 0534  59         pop CX
    479 0535  5B         pop     BX
    480 0536  58         pop     AX
    481 0537  5E         pop SI
    482 0538  CF         iret
    483 0539        new_int9h endp
    484
    485         ;=== Обработчик прерывания int 1Ch ===;
    486         ;=== Вызывается каждые 55 мс ===;
    487         ; Новый обработчик  new_int1Ch
    488
    489 0539        new_int1Ch proc far
    490 0539  50        push AX
    491 053A  0E        push CS
    492 053B  1F        pop DS
    493         ; Вызов старого обработчика  old_int1ChOffset
    494
    495 053C  9C        pushf
    496 053D  2E: FF 1E 016Ar       call dword ptr CS:[old_int1ChOffset]
    497
    498 0542  80 3E 0143r FF       cmp signaturePrintingEnabled, true ; если нажата управляющая клавиша (в
данном случае  +
    499         F1)
    500 0547  75 1D        jne _notToPrint
    501
    502         ; Контроль счетчика циклов
    503
    504 0549  81 3E 0174r 0080        cmp counter, printDelay*1000/55 + 1 ; если кол-во "тактов"
эквивалентно     +
    505         %printDelay% секундам
    506 054F  74 03         je _letsPrint
    507
    508 0551  EB 0E 90         jmp _dontPrint
    509
    510 0554          _letsPrint:
    511 0554  F6 16 0143r         not signaturePrintingEnabled
    512 0558  C7 06 0174r 0000         mov counter, 0
    513 055E  E8 0094          call printSignature
    514
    515 0561          _dontPrint:
    516 0561  83 06 0174r 01         add counter, 1
    517
    518 0566         _notToPrint:
    519
    520 0566  58        pop AX
    521
    522 0567  CF        iret
    523 0568        new_int1Ch endp
    524
    525         ;=== Обработчик прерывания int 2Fh ===;
    526         ;=== Служит для:
    527         ;===  1) проверки факта присутствия TSR в памяти (при AH=0FFh, AL=0)
    528         ;===     будет возвращён AH='i' в случае, если TSR уже загружен
```

```
529         ;===  2) выгрузки TSR из памяти (при AH=0FFh, AL=1)
530         ;===
531          ; Новый обработчик  new_int2Fh
532
533 0568        new_int2Fh proc
534 0568  80 FC FF         cmp     AH, 0FFh      ;наша функция?
535 056B  75 0B         jne     _2Fh_std      ;нет - на старый обработчик
536 056D  3C 00         cmp     AL, 0   ;подфункция проверки, загружен ли резидент в память?
537 056F  74 0C         je      _already_installed
538 0571  3C 01         cmp     AL, 1   ;подфункция выгрузки из памяти?
539 0573  74 0B         je      _uninstall
540 0575  EB 01 90         jmp     _2Fh_std      ;нет - на старый обработчик
541
542 0578        _2Fh_std:
543         ; Вызов старого обработчика  old_int2FhOffset
544
545 0578  2E: FF 2E 016Er       jmp     dword ptr CS:[old_int2FhOffset] ;вызов старого обработчика
546
547 057D        _already_installed:
548 057D  B4 69          mov     AH, 'i' ;вернём 'i', если резидент загружен     в память
549 057F  CF          iret
550
551 0580        _uninstall:
552 0580  1E         push    DS
553 0581  06         push    ES
554 0582  52         push    DX
555 0583  53         push    BX
556
557 0584  33 DB         xor BX, BX
558
559          ; CS = ES, для доступа к переменным
560 0586  0E         push CS
561 0587  07         pop ES
562         ; выгрузка резидента
563
564 0588  B8 2509         mov     AX, 2509h
565 058B  26: 8B 16 0166r       mov DX, ES:old_int9hOffset  ; возвращаем вектор прерывания
566 0590  26: 8E 1E 0168r   mov DS, ES:old_int9hSegment     ; на место
567 0595  CD 21         int     21h
568
569 0597  B8 251C         mov     AX, 251Ch
570 059A  26: 8B 16 016Ar       mov DX, ES:old_int1ChOffset  ; возвращаем вектор прерывания
571 059F  26: 8E 1E 016Cr   mov DS, ES:old_int1ChSegment      ; на место
572 05A4  CD 21         int     21h
573
574 05A6  B8 252F         mov     AX, 252Fh
575 05A9  26: 8B 16 016Er       mov DX, ES:old_int2FhOffset  ; возвращаем вектор прерывания
576 05AE  26: 8E 1E 0170r   mov DS, ES:old_int2FhSegment      ; на место
577 05B3  CD 21         int     21h
578
579 05B5  2E: 8E 06 002C        mov     ES, CS:2Ch      ; загрузим в ES адрес окружения
580 05BA  B4 49         mov     AH, 49h      ; выгрузим из памяти окружение
581 05BC  CD 21         int     21h
582 05BE  72 0B         jc _notRemove
583
584 05C0  0E         push    CS
585 05C1  07         pop     ES      ;в ES - адрес резидентной программы
586 05C2  B4 49         mov     AH, 49h  ;выгрузим из памяти резидент
587 05C4  CD 21         int     21h
588 05C6  72 03         jc _notRemove
589 05C8  EB 15 90         jmp _unloaded
590
591 05CB        _notRemove: ; не удалось выполнить выгрузку
592       ; вывод сообщения о неудачной выгрузке
593 05CB  B4 03         mov AH, 03h          ; получаем позицию курсора
594 05CD  CD 10         int 10h
595 05CF  BD 0395r         lea BP, noRemoveMsg
596 05D2  B9 001D         mov CX, noRemoveMsg_length
597 05D5  B3 07         mov BL, 0111b
598 05D7  B8 1301         mov AX, 1301h
599 05DA  CD 10         int 10h
600 05DC  EB 12 90         jmp _2Fh_exit
601
602 05DF        _unloaded: ; выгрузка прошла успешно
603       ; вывод сообщения об удачной выгрузке
604 05DF  B4 03         mov AH, 03h          ; получаем позицию курсора
605 05E1  CD 10         int 10h
```

```
606 05E3  BD 0384r        lea BP, removedMsg
607 05E6  B9 0011        mov CX, removedMsg_length
608 05E9  B3 07        mov BL, 0111b
609 05EB  B8 1301        mov AX, 1301h
610 05EE  CD 10        int 10h
611
612 05F0        _2Fh_exit:
613 05F0  5B        pop BX
614 05F1  5A        pop     DX
615 05F2  07        pop     ES
616 05F3  1F        pop     DS
617 05F4  CF        iret
618 05F5        new_int2Fh endp
619
620         ;=== Процедура вывода подписи (ФИО, группа)
621         ;=== Настраивается значениями переменных в начале исходника
622         ;===
623         ; Вывод подписи
624
625 05F5        printSignature proc
626 05F5  50        push AX
627 05F6  52        push DX
628 05F7  51        push CX
629 05F8  53        push BX
630 05F9  06        push ES
631 05FA  54        push SP
632 05FB  55        push BP
633 05FC  56        push SI
634 05FD  57        push DI
635
636 05FE  33 C0        xor AX, AX
637 0600  33 DB        xor BX, BX
638 0602  33 D2        xor DX, DX
639
640 0604  B4 03        mov AH, 03h          ;чтение текущей позиции курсора
641 0606  CD 10        int 10h
642 0608  52        push DX           ;помещаем информацию о     +
643         положении курсора в стек
644
645 0609  83 3E 0176r 00       cmp printPos, 0
646 060E  74 0E        je _printTop
647
648 0610  83 3E 0176r 01       cmp printPos, 1
649 0615  74 0E        je _printCenter
650
651 0617  83 3E 0176r 02       cmp printPos, 2
652 061C  74 0E        je _printBottom
653
654          ;все числа подобраны на глаз...
655 061E         _printTop:
656 061E  B6 00         mov DH, 0
657 0620  B2 0F         mov DL, 15
658 0622  EB 0F 90         jmp _actualPrint
659
660 0625         _printCenter:
661 0625  B6 09         mov DH, 9
662 0627  B2 1E         mov DL, 30
663 0629  EB 08 90         jmp _actualPrint
664
665 062C         _printBottom:
666 062C  B6 13         mov DH, 19
667 062E  B2 0F         mov DL, 15
668 0630  EB 01 90         jmp _actualPrint
669
670 0633         _actualPrint:
671 0633  B4 0F         mov AH, 0Fh         ;чтение текущего видеорежима. в+
672         BH - текущая страница
673 0635  CD 10         int 10h
674
675 0637  0E         push CS
676 0638  07         pop ES          ;указываем ES на CS
677
678           ;вывод 'верхушки' таблицы
679 0639  52         push DX
680 063A  BD 030Dr         lea BP, tableTop         ;помещаем в BP указатель на    +
681         выводимую строку
682 063D  B9 000E         mov CX, tableTop_length      ;в CX - длина строки
```

```
683 0640  B3 07         mov BL, 0111b        ;цвет выводимого текста ref:     +
684         http://en.wikipedia.org/wiki/BIOS_color_attributes
685 0642  B8 1301         mov AX, 1301h         ;AH=13h - номер ф-ии, AL=01h - +
686         курсор перемещается при выводе каждого из символов строки
687 0645  CD 10         int 10h
688 0647  5A         pop DX
689 0648  FE C6         inc DH
690
691
692           ;вывод первой линии
693 064A  52         push DX
694 064B  BD 0178r         lea BP, signatureLine1
695 064E  B9 000E         mov CX, Line1_length
696 0651  B3 07         mov BL, 0111b
697 0653  B8 1301         mov AX, 1301h
698 0656  CD 10         int 10h
699 0658  5A         pop DX
700 0659  FE C6         inc DH
701
702           ;вывод второй линии
703 065B  52         push DX
704 065C  BD 0186r         lea BP, signatureLine2
705 065F  B9 000E         mov CX, Line2_length
706 0662  B3 07         mov BL, 0111b
707 0664  B8 1301         mov AX, 1301h
708 0667  CD 10         int 10h
709 0669  5A         pop DX
710 066A  FE C6         inc DH
711
712           ;вывод третьей линии
713 066C  52         push DX
714 066D  BD 0194r         lea BP, signatureLine3
715 0670  B9 000E         mov CX, Line3_length
716 0673  B3 07         mov BL, 0111b
717 0675  B8 1301         mov AX, 1301h
718 0678  CD 10         int 10h
719 067A  5A         pop DX
720 067B  FE C6         inc DH
721
722           ;вывод 'низа' таблицы
723 067D  52         push DX
724 067E  BD 031Br         lea BP, tableBottom
725 0681  B9 000E         mov CX, tableBottom_length
726 0684  B3 07         mov BL, 0111b
727 0686  B8 1301         mov AX, 1301h
728 0689  CD 10         int 10h
729 068B  5A         pop DX
730 068C  FE C6         inc DH
731
732 068E  33 DB         xor BX, BX
733 0690  5A         pop DX          ;восстанавливаем из стека     +
734         прежнее положение курсора
735 0691  B4 02         mov AH, 02h         ;меняем положение курсора на   +
736         первоначальное
737 0693  CD 10         int 10h
738 0695  E8 FD22         call changeFx
739
740 0698  5F        pop DI
741 0699  5E        pop SI
742 069A  5D        pop BP
743 069B  5C        pop SP
744 069C  07        pop ES
745 069D  5B        pop BX
746 069E  59        pop CX
747 069F  5A        pop DX
748 06A0  58        pop AX
749
750 06A1  C3        ret
751 06A2       printSignature endp
752         ; Смена  шрифта
753
754         ;=== Функция, которая в зависимости от флага cursiveEnabled меняет начертание символа с
курсива+
755         на обычное и наоброт
756         ;=== Сама смена происходит в процедуре changeFont, а здесь подготавливаются данные
757 06A2       setCursive proc
758 06A2  06        push ES ; сохраняем регистры
```

```
759 06A3  50        push AX
760 06A4  0E        push CS
761 06A5  07        pop ES
762
763 06A6  80 3E 0144r FF       cmp cursiveEnabled, true
764 06AB  75 30        jne _restoreSymbol
765          ; если флаг равен true, выполняем замену символа на курсивный вариант,
766          ; предварительно сохраняя старый символ в savedSymbol
767
768 06AD  E8 004C        call saveFont
769 06B0  8A 0E 0155r       mov CL, charToCursiveIndex
770 06B4        _shifTtable:
771          ; мы получаем в BP таблицу всех символов. адрес указывает на символ 0
772          ; поэтому нуэно совершить сдвиг 16*X - где X - код символа
773 06B4  83 C5 10        add BP, 16
774 06B7  E2 FB        loop _shiftTable
775
776          ; при savefont смещается регистр ES
777          ; поэтому приходится делать такие махинации, чтобы
778          ; записать полученный элемент в savedSymbol
779          ; swap(ES, DS) и сохранение старого значения DS
780 06B9  1E        push DS
781 06BA  58        pop AX
782 06BB  06        push ES
783 06BC  1F        pop DS
784 06BD  50        push AX
785 06BE  07        pop ES
786 06BF  50        push AX
787
788 06C0  8B F5        mov SI, BP
789 06C2  BF 0156r        lea DI, savedSymbol
790          ; сохраняем в переменную savedSymbol
791          ; таблицу нужного символа
792 06C5  B9 0010        mov CX, 16
793          ; movsb из DS:SI в ES:DI
794 06C8  F3> A4        rep movsb
795          ; исходные позиции сегментов возвращены
796 06CA  1F        pop DS ; восстановление DS
797
798          ; заменим написание символа на курсив
799 06CB  B9 0001        mov CX, 1
800 06CE  B6 00        mov DH, 0
801 06D0  8A 16 0155r       mov DL, charToCursiveIndex
802 06D4  BD 0145r        lea BP, cursiveSymbol
803 06D7  E8 0015        call changeFont
804 06DA  EB 10 90        jmp _exitSetCursive
805         ; Восстановление  шрифта
806
807 06DD        _restoreSymbol:
808          ; если флаг равен 0, выполняем замену курсивного символа на старый вариант
809
810 06DD  B9 0001        mov CX, 1
811 06E0  B6 00        mov DH, 0
812 06E2  8A 16 0155r       mov DL, charToCursiveIndex
813 06E6  BD 0156r        lea bp, savedSymbol
814 06E9  E8 0003        call changeFont
815
816 06EC        _exitSetCursive:
817 06EC  58        pop AX
818 06ED  07        pop ES
819 06EE  C3        ret
820 06EF        setCursive endp
821
822         ;=== Функция смены начертания символа (курсив/нормальное)
823         ;===
824         ; *** входные данные
825         ; DL = номер символа для замены
826         ; CX = Кол-во символов заменяемых изображений символов
827         ; (начиная с символа указанного в DX)
828         ; ES:bp = адрес таблицы
829         ;
830         ; *** описание работы процедуры
831         ; Происходит вызов int 10h (видеосервис)
832         ; с функцией AH = 11h (функции знакогенератора)
833         ; Параметр AL = 0 сообщает, что будет заменено изображение
834         ; символа для текущего шрифта
835         ; В случаях, когда AL = 1 или 2, будет заменено изображение
```

```
836         ; только для опредленного шрифта (8x14 и 8x8 соответственно)
837         ; Параметр BH = 0Eh сообщает, что на опредление каждого изображения символа
838         ; расходуется по 14 байт (режим 8x14 бит как раз 14 байт)
839         ; Параметр BL = 0 - блок шрифта для загрузки (от 0 до 4)
840         ;
841         ; *** результат
842         ; изображение указанного(ых) символа(ов) будет заменено
843         ; на предложенное пользователем.
844         ; Изменению подвергнутся все символы, находящиеся на экране,
845         ; то есть если изображение заменено, старый вариант нигде уже не проявится
846
847 06EF        changeFont proc
848 06EF  50          push AX
849 06F0  53          push BX
850 06F1  B8 1100         mov AX, 1100h
851 06F4  BB 1000         mov BX, 1000h
852 06F7  CD 10         int 10h
853 06F9  58          pop AX
854 06FA  5B          pop BX
855 06FB  C3          ret
856 06FC        changeFont endp
857
858         ;=== Функция сохранения нормального начертания символа
859         ;===
860         ; *** входные данные
861         ; BH - тип возвращаемой символьной таблицы
862         ;  0 - таблица из int 1fh
863         ;  1 - таблица из int 44h
864         ;  2-5 - таблица из 8x14, 8x8, 8x8 (top), 9x14
865         ;  6 - 8x16
866         ;
867         ; *** описание работы процедуры
868         ; Происходит вызов int 10h (видеосервис)
869         ; с функцией AH = 11h (функции знакогенератора)
870         ; Параметр AL = 30 - подфункция получения информации о EGA
871         ;
872         ; *** результат
873         ; в ES:BP находится таблица символов (полная)
874         ; в CX находится байт на символ
875         ; в DL количество экранных строк
876         ; ВАЖНО! Происходит сдвиг регистра ES
877         ; ( ES становится равным C000h )
878
879 06FC        saveFont proc
880 06FC  50          push AX
881 06FD  53          push BX
882 06FE  B8 1130         mov AX, 1130h
883 0701  BB 0600         mov BX, 0600h
884 0704  CD 10         int 10h
885 0706  58          pop AX
886 0707  5B          pop BX
887 0708  C3          ret
888 0709        saveFont endp
889
890
891         ;=== Отсюда начинается выполнение основной части программы ===;
892         ;===
893         ; Часть Инициализации
894
895 0709         _initTSR:         ; старт резидента
896 0709  B4 03         mov AH, 03h
897 070B  CD 10         int 10h
898 070D  52          push DX
899 070E  B4 00         mov AH,00h          ; установка видеорежима (83h  текст    +
900           80x25  16/8  CGA,EGA  b800  Comp,RGB,Enhanced), без очистки экрана
901 0710  B0 83         mov AL,83h
902 0712  CD 10         int 10h
903 0714  5A          pop DX
904 0715  B4 02         mov AH, 02h
905 0717  CD 10         int 10h
906
907         ; Новые вектора Инициализации
908
909 0719  E8 00B3    call commandParamsParser
910 071C  B8 3509         mov AX,3509h       ; получить в ES:BX вектор 09
911 071F  CD 21     int 21h     ; прерывания
912
```

```
913          ;@ === Удаление резидента из памяти ===
914 0721  80 3E 0172r FF       cmp unloadTSR, true
915 0726  74 03        je _removingOnParameter
916 0728  EB 15 90        jmp _notRemovingNow
917         ; Проверка загрузки
918
919 072B         _removingOnParameter:
920 072B  B4 FF         mov AH, 0FFh
921 072D  B0 00         mov AL, 0
922 072F  CD 2F         int 2Fh
923 0731  80 FC 69         cmp AH, 'i'  ; проверка того, загружена ли уже программа
924 0734  74 7D         je _remove
925 0736  B4 09         mov AH, 09h
926 0738  BA 0366r         lea DX, notInstalledMsg
927 073B  CD 21         int 21h
928 073D  CD 20         int 20h
929
930 073F         _notRemovingNow:
931
932 073F  80 3E 0173r FF       cmp notLoadTSR, true       ; если была выведена справка
933 0744  74 0E        je _exit_tmp          ; просто выходим
934
935
936          ;@ если необходимо выгружать по параметру коммандной строки, то оставляем их
937 0746  B4 FF        mov AH, 0FFh
938 0748  B0 00        mov AL, 0
939 074A  CD 2F        int 2Fh
940 074C  80 FC 69        cmp AH, 'i'  ; проверка того, загружена ли уже программа
941 074F  74 6B        je _alreadyInstalled
942
943 0751  EB 04 90        jmp _tmp
944
945 0754         _exit_tmp:
946 0754  EB 77 90         jmp _exit
947
948 0757         _tmp:
949 0757  06        push ES
950         ; Проверка наличия памяти
951
952 0758  A1 002C    mov AX, DS:[2Ch]   ; psp
953 075B  8E C0    mov ES, AX
954 075D  B4 49    mov AH, 49h    ; хватит памяти чтоб остаться
955 075F  CD 21    int 21h    ; резидентом?
956 0761  07    pop ES
957 0762  72 62    jc _notMem    ; не хватило ? выходим
958         ; Сохранение старых векторов и установка новых
959
960
961          ;== int 09h ==;
962
963 0764  2E: 89 1E 0166r       mov     word ptr CS:old_int9hOffset, BX
964 0769  2E: 8C 06 0168r       mov     word ptr CS:old_int9hSegment, ES
965 076E  B8 2509    mov AX, 2509h    ; установим вектор на 09
966 0771  BA 045Cr    mov DX, offset new_int9h      ; прерывание
967 0774  CD 21    int 21h
968
969          ;== int 1Ch ==;
970 0776  B8 351C        mov AX,351Ch       ; получить в ES:BX вектор 1C
971 0779  CD 21    int 21h    ; прерывания
972 077B  2E: 89 1E 016Ar       mov     word ptr CS:old_int1ChOffset, BX
973 0780  2E: 8C 06 016Cr       mov     word ptr CS:old_int1ChSegment, ES
974 0785  B8 251C        mov AX, 251Ch       ; установим вектор на 1C
975 0788  BA 0539r        mov DX, offset new_int1Ch    ; прерывание
976 078B  CD 21        int 21h
977
978          ;== int 2Fh ==;
979 078D  B8 352F        mov AX,352Fh       ; получить в ES:BX вектор 1C
980 0790  CD 21    int 21h    ; прерывания
981 0792  2E: 89 1E 016Er       mov     word ptr CS:old_int2FhOffset, BX
982 0797  2E: 8C 06 0170r       mov     word ptr CS:old_int2FhSegment, ES
983 079C  B8 252F        mov AX, 252Fh       ; установим вектор на 2F
984 079F  BA 0568r        mov DX, offset new_int2Fh    ; прерывание
985 07A2  CD 21        int 21h
986
987 07A4  E8 FC13        call changeFx
988 07A7  BA 0329r    mov DX, offset installedMsg      ; выводим что все ок
989 07AA  B4 09    mov AH, 9
```

```
 990 07AC  CD 21    int 21h
 991         ; Оставить в ОП резидентом (027H)
 992
 993 07AE  BA 0709r    mov DX, offset _initTSR       ; остаемся в памяти резидентом
 994 07B1  CD 27    int 27h    ; и выходим
 995       ; конец основной программы
 996         ; Выгрузка резидента (сигнал в TSR)
 997
 998 07B3        _remove: ; выгрузка программы из памяти
 999 07B3  B4 FF         mov AH, 0FFh
1000 07B5  B0 01         mov AL, 1
1001 07B7  CD 2F         int 2Fh
1002 07B9  EB 12 90         jmp _exit
1003 07BC        _alreadyInstalled:
1004 07BC  B4 09         mov AH, 09h
1005 07BE  BA 033Cr         lea DX, alreadyInstalledMsg
1006 07C1  CD 21         int 21h
1007 07C3  EB 08 90         jmp _exit
1008 07C6        _notMem:     ; не хватает памяти, чтобы остаться резидентом
1009 07C6  BA 0352r    mov DX, offset noMemMsg
1010 07C9  B4 09    mov AH, 9
1011 07CB  CD 21    int 21h
1012 07CD        _exit:      ; выход
1013 07CD  CD 20    int 20h
1014
1015         ;=== Процедура проверки параметров ком. строки ===;
1016         ;===
1017         ; Проверка и разбор параметров
1018
1019 07CF        commandParamsParser proc
1020 07CF  0E         push CS
1021 07D0  07         pop ES
1022 07D1  C6 06 0172r 00       mov unloadTSR, 0
1023 07D6  C6 06 0173r 00       mov notLoadTSR, 0
1024
1025 07DB  BE 0080         mov SI, 80h        ;SI=смещение командной строки.
1026 07DE  AC         lodsb          ;Получим кол-во символов.
1027 07DF  0A C0         or AL, AL         ;Если 0 символов введено,
1028 07E1  74 3F         jz _exitHelp        ;то все в порядке.
1029
1030 07E3         _nextChar:
1031
1032 07E3  46         inc SI          ;Теперь SI указывает на первый символ  +
1033         строки.
1034
1035 07E4  80 3C 0D         cmp [SI], BYTE ptr 13
1036 07E7  74 39         je _exitHelp
1037
1038
1039 07E9  AD          lodsw         ;Получаем два символа
1040 07EA  3D 3F2F          cmp AX, '?/'        ;Это '/?' (данные расположены в     +
1041         обратном порядк, т.е. AL:AH вместо AH:AL)
1042 07ED  74 08          je _question
1043 07EF  3D 752F          cmp AX, 'u/'
1044 07F2  74 1A          je _finishTSR
1045 07F4  EB 2C 90          jmp _exitHelp
1046         ; Вывод справки
1047
1048
1049 07F7         _question:
1050           ; вывод строки помощи
1051 07F7  B4 03          mov AH,03
1052 07F9  CD 10          int 10h
1053 07FB  BD 01A2r          lea BP, helpMsg
1054 07FE  B9 0146          mov CX, helpMsg_length
1055 0801  B3 07          mov BL, 0111b
1056 0803  B8 1301          mov AX, 1301h
1057 0806  CD 10          int 10h
1058           ; конец вывода строки помощи
1059 0808  F6 16 0173r         not notLoadTSR      ;флаг того, что необходимо не загружать резидент
1060 080C  EB D5          jmp _nextChar
1061
1062          ;@ === Удаление резидента из памяти ===
1063          ;@ Если по варианту необходимо выгружать резидент по параметру '/u' коммандной строки,
1064          ;@ нужно использовать следующий код, в остальных случаях необходимо закомменитровать
1065          ;@ этот код, кроме названия метки! (по желанию можно избавиться и от метки, но     +
1066         аккуратно просмотреть использование)
```

```
   1067 080E         _finishTSR:
   1068           ;not unloadTSR     ;флаг того, что необходимо выгузить резидент
   1069           ;jmp _nextChar
   1070
   1071 080E  EB 12 90        jmp _exitHelp
   1072
   1073 0811         _errorParam:
   1074           ;вывод строки
   1075 0811  B4 03          mov AH,03
   1076 0813  CD 10          int 10h
   1077 0815  BD 02E8r          lea BP, CS:errorParamMsg
   1078 0818  B9 0025          mov CX, errorParamMsg_length
   1079 081B  B3 07          mov BL, 0111b
   1080 081D  B8 1301          mov AX, 1301h
   1081 0820  CD 10          int 10h
   1082           ;конец вывода строки
   1083 0822         _exitHelp:
   1084 0822  C3        ret
   1085 0823        commandParamsParser endp
   1086
   1087 0823        code ends
   1088
   1089         end _start

Symbol Table
Symbol Name     Type  Value
??DATE      Text  "05/07/24"
??FILENAME     Text  "tsr   "
??TIME      Text  "21:17:46"
??VERSION     Number 030A
@CPU      Text  0101H
@CURSEG      Text  CODE
@FILENAME     Text  TSR
@WORDSIZE     Text  2
ALREADYINSTALLEDMSG    Byte  CODE:033C
CHANGEFONT     Near  CODE:06EF
CHANGEFX     Near  CODE:03BA
CHARTOCURSIVEINDEX    Byte  CODE:0155
COMMANDPARAMSPARSER    Near  CODE:07CF
COUNTER      Word  CODE:0174
CURSIVEENABLED     Byte  CODE:0144
CURSIVESYMBOL     Byte  CODE:0145
ERRORPARAMMSG     Byte  CODE:02E8
ERRORPARAMMSG_LENGTH    Number 0025
F1_TXT      Byte  CODE:03B2
F2_TXT      Byte  CODE:03B4
F3_TXT      Byte  CODE:03B6
F9_TXT      Byte  CODE:03B8
FX_LENGTH     Number 0002
HELPMSG      Byte  CODE:01A2
HELPMSG_LENGTH     Number 0146
IGNOREDCHARS     Byte  CODE:0103
IGNOREDLENGTH     Number 0034
IGNOREENABLED     Byte  CODE:0137
INSTALLEDMSG     Byte  CODE:0329
LINE1_LENGTH     Number 000E
LINE2_LENGTH     Number 000E
LINE3_LENGTH     Number 000E
NEW_INT1CH     Far  CODE:0539
NEW_INT2FH     Near  CODE:0568
NEW_INT9H     Far  CODE:045C
NOMEMMSG     Byte  CODE:0352
NOREMOVEMSG     Byte  CODE:0395
NOREMOVEMSG_LENGTH    Number 001D
NOTINSTALLEDMSG     Byte  CODE:0366
NOTLOADTSR     Byte  CODE:0173
OLD_INT1CHOFFSET    Word  CODE:016A
OLD_INT1CHSEGMENT    Word  CODE:016C
OLD_INT2FHOFFSET    Word  CODE:016E
OLD_INT2FHSEGMENT    Word  CODE:0170
OLD_INT9HOFFSET     Word  CODE:0166
OLD_INT9HSEGMENT    Word  CODE:0168
PRINTDELAY     Number 0007
PRINTPOS     Word  CODE:0176
PRINTSIGNATURE     Near  CODE:05F5
REMOVEDMSG     Byte  CODE:0384
REMOVEDMSG_LENGTH    Number 0011
```

```
SAVEDSYMBOL     Byte  CODE:0156
SAVEFONT     Near  CODE:06FC
SETCURSIVE     Near  CODE:06A2
SIGNATURELINE1     Byte  CODE:0178
SIGNATURELINE2     Byte  CODE:0186
SIGNATURELINE3     Byte  CODE:0194
SIGNATUREPRINTINGENABLED   Byte  CODE:0143
TABLEBOTTOM     Byte  CODE:031B
TABLEBOTTOM_LENGTH    Number 000E
TABLETOP     Byte  CODE:030D
TABLETOP_LENGTH     Number 000E
TRANSLATEENABLED    Byte  CODE:0142
TRANSLATEFROM     Byte  CODE:0138
TRANSLATELENGTH     Number 0005
TRANSLATETO     Byte  CODE:013D
TRUE      Number 00FF
UNLOADTSR     Byte  CODE:0172
_2FH_EXIT     Near  CODE:05F0
_2FH_STD     Near  CODE:0578
_ACTUALPRINT     Near  CODE:0633
_ALREADYINSTALLED    Near  CODE:07BC
_ALREADY_INSTALLED    Near  CODE:057D
_BLOCK      Near  CODE:0507
_CHECKF1     Near  CODE:03C9
_CHECKF2     Near  CODE:03ED
_CHECKF3     Near  CODE:040E
_CHECKF9     Near  CODE:042F
_CHECK_IGNORED     Near  CODE:04FB
_CHECK_TRANSLATE    Near  CODE:050F
_CHECK_TRANSLATE_LOOP    Near  CODE:051C
_DONTPRINT     Near  CODE:0561
_ERRORPARAM     Near  CODE:0811
_EXIT      Near  CODE:07CD
_EXITHELP     Near  CODE:0822
_EXITSETCURSIVE     Near  CODE:06EC
_EXIT_TMP     Near  CODE:0754
_F1      Near  CODE:04A4
_F2      Near  CODE:04B5
_F3      Near  CODE:04C3
_F9      Near  CODE:0496
_FINISHTSR     Near  CODE:080E
_GO      Near  CODE:04EB
_GREENF1     Near  CODE:03E6
_GREENF2     Near  CODE:040A
_GREENF3     Near  CODE:042B
_GREENF9     Near  CODE:044C
_INITTSR     Near  CODE:0709
_LETSPRINT     Near  CODE:0554
_NEXTCHAR     Near  CODE:07E3
_NOTMEM      Near  CODE:07C6
_NOTREMOVE     Near  CODE:05CB
_NOTREMOVINGNOW     Near  CODE:073F
_NOTTOPRINT     Near  CODE:0566
_OUTFX      Near  CODE:0450
_PRINTBOTTOM     Near  CODE:062C
_PRINTCENTER     Near  CODE:0625
_PRINTTOP     Near  CODE:061E
_QUESTION     Near  CODE:07F7
_QUIT      Near  CODE:0531
_REDF1      Near  CODE:03DF
_REDF2      Near  CODE:0403
_REDF3      Near  CODE:0424
_REDF9      Near  CODE:0445
_REMOVE      Near  CODE:07B3
_REMOVINGONPARAMETER    Near  CODE:072B
_RESTORESYMBOL     Near  CODE:06DD
_SHIFTTABLE     Near  CODE:06B4
_START      Near  CODE:0100
_TEST_FX     Near  CODE:0494
_TMP      Near  CODE:0757
_TRANSLATE     Near  CODE:0528
_TRANSLATE_OR_IGNORE    Near  CODE:04D1
_UNINSTALL     Near  CODE:0580
_UNLOADED     Near  CODE:05DF

Groups & Segments    Bit Size Align  Combine Class
CODE      16  0823 Para   none   CODE
```