Московский государственный технический университет
им. Н.Э. Баумана

УТВЕРЖДАЮ:

Большаков С.А.
"__"_____2020 г.

Курсовая работа по курсу «Системное программирование»

Исходный текст программного продукта
(вид документа)

писчая бумага
(вид носителя)

14
(количество листов)

ИСПОЛНИТЕЛЬ:

студенты группы ИУ5-41 _____
Алехин С.С.
"__"_____2020 г.

Москва – 2020

Содержание

1. Файл tsr.lst

```
Turbo Assembler  Version 3.1     05/22/20 17:48:24     Page 1
tsr.ASM


 1         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 2         ; ИУ5-41Б Алехин С.С. Вариант №2
 3         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 4
 5 0000        code segment    'code'
 6          assume  CS:code, DS:code
 7          org     100h
 8 0100         _start:
 9
10 0100  E9 05D5        jmp _initTSR             +
11         ;Прыгаем на начало программы
12
13 0103  80 81 82 83 84 85 F0+ ignoredRussianChars      DB      'АБВГДЕЁЖЗИЙКЛМОПРСТУФЧЦЧШЩЪЫЬЭЮЯ'      ;Список+
14       86 87 88 89 8A 8B 8C+  игнорируемых символов
15       8E 8F 90 91 92 93 94+
16       97 96 97 98 99 9A 9B+
17       9C 9D 9E 9F
18       =0020        ignoredRussianCharsLenght  equ $-ignoredRussianChars        ;Длина +
19         строки ignoredRussianChars
20 0123  00        isIgnoredRussianChars  DB  0           +
21         ;Флаг функции игнорирования ввода
22
23 0124  54 3A 50 42 52       translateStrFrom        DB      'T:PBR'        +
24         ;Символы для замены
25 0129  85 86 87 88 8A       translateStrTo        DB      'ЕЖЗИК'        +
26         ;Символы на которые будет идти замена
27       =0005        translateSirFromLength      equ     $-translateStrTo        +
28         ;Длина строки trasnlateFrom
29 012E  00        isTranslateStr        DB      0         +
30         ;Флаг функции перевода
31
32       =0005        signatureFormDelay        equ     5         +
33         ;Задержка перед выводом "подписи" в секундах
34 012F  0000        signatureCounter        DW      0
35 0131  00        isSignatureForm        DB      0         +
36         ;Флаг функции вывода информации об авторе
37 0132  B3 80 AB A5 E5 A8 AD+      signatureLine1        DB      179, 'Алехин Сергей Сергеевич     +
38       20 91 A5 E0 A3 A5 A9+  ', 179
39       20 91 A5 E0 A3 A5 A5+
40       A2 A8 E7 20 20 20 20+
41       20 20 20 20 20 20 20+
42       20 20 20 20 20 20 20+
43       20 20 20 20 20 20 20+
44       20 20 B3
45       =0034        signatureLine1Length      equ     $-signatureLine1
46 0166  B3 88 93 35 2D 34 31+      signatureLine2        DB      179, 'ИУ5-41Б       +
47       81 20 20 20 20 20 20+  ', 179
48       20 20 20 20 20 20 20+
49       20 20 20 20 20 20 20+
50       20 20 20 20 20 20 20+
51       20 20 20 20 20 20 20+
52       20 20 20 20 20 20 20+
53       20 20 B3
54       =0034        signatureLine2Length      equ     $-signatureLine2
55 019A  B3 82 A0 E0 A8 A0 AD+      signatureLine3        DB      179, 'Вариант #2       +
56       E2 20 23 32 20 20 20+  ', 179
57       20 20 20 20 20 20 20+
58       20 20 20 20 20 20 20+
59       20 20 20 20 20 20 20+
60       20 20 20 20 20 20 20+
61       20 20 20 20 20 20 20+
62       20 20 B3
63       =0034        signatureLine3Length      equ     $-signatureLine3
64 01CE  DA 32*(C4) BF       tableTop          DB      218, signatureLine1Length-2 dup+
65         (196), 191
66       =0034        tableTopLength        equ     $-tableTop
67 0202  C0 32*(C4) D9       tableBottom         DB      192, signatureLine1Length-2 dup+
68         (196), 217
69       =0034        tableBottomLength        equ  $-tableBottom
70
71 0236  9A        charToCursiveIndex        DB 'ъ'         +
72         ;Символ для замены
73 0237  00        cursiveSymbol        DB 00000000b
74 0238  00               DB 00000000b
75 0239  F0               DB 11110000b
```

```
 76 023A  F0               DB 11110000b
 77 023B  30               DB 00110000b
 78 023C  30               DB 00110000b
 79 023D  60               DB 01100000b
 80 023E  7C               DB 01111100b
 81 023F  66               DB 01100110b
 82 0240  C3               DB 11000011b
 83 0241  C6               DB 11000110b
 84 0242  FC               DB 11111100b
 85 0243  00               DB 00000000b
 86 0244  00               DB 00000000b
 87 0245  00               DB 00000000b
 88 0246  00        isCursiveChar        DB      0         +
 89         ;Флаг перевода символа в курсив
 90 0247  10*(FF)        savedSymbol        DB 16 dup(0FFh)        +
 91         ;Переменная для хранения старого символа
 92
 93       =00FF        true          equ     0FFh       +
 94         ;Константа истинности
 95 0257  ????        old_int9hOffset        DW      ?         +
 96         ;Адрес старого обработчика int 9h
 97 0259  ????        old_int9hSegment        DW      ?         +
 98         ;Сегмент старого обработчика int 9h
 99 025B  ????        old_int1ChOffset        DW      ?         +
100         ;Адрес старого обработчика int 1Ch
101 025D  ????        old_int1ChSegment        DW      ?         +
102         ;Сегмент старого обработчика int 1Ch
103 025F  ????        old_int2FhOffset        DW      ?         +
104         ;Адрес старого обработчика int 2Fh
105 0261  ????        old_int2FhSegment        DW      ?         +
106         ;Сегмент старого обработчика int 2Fh
107
108 0263  00        unloadTSR          DB      0        +
109         ;1 - выгрузить резидент
110 0264  00        notLoadTSR          DB      0        +
111         ;1 - не загружать
112
113 0265  3E 74 73 72 2E 63 6F+      helpMsg DB '>tsr.com [/?]', 10, 13
114       6D 20 5B 2F 3F 5D 0A+
115       0D
116 0274  20 5B 2F 3F 5D 20 2D+        DB ' [/?] - вывод справки', 10, 13
117       20 A2 EB A2 AE A4 20+
118       63 AF E0 A0 A2 AA A8+
119       0A 0D
120 028B  20 20 46 38 20 20 2D+        DB '  F8  - вывод ФИО, группы и варианта по таймеру в верхей части     +
121       20 A2 EB A2 AE A4 20+  экрана', 10, 13
122       94 88 8E 2C 20 A3 E0+
123       E3 AF AF EB 20 A8 20+
124       A2 A0 E0 A8 A0 AD E2+
125       A0 20 AF AE 20 E2 A0+
126       A9 AC A5 E0 E3 20 A2+
127       20 A2 A5 E0 E5 A5 A9+
128       20 E7 A0 E1 E2 A8 20+
129       ED AA E0 A0 AD A0 0A+
130       0D
131 02D2  20 20 46 39 20 20 2D+        DB '  F9  - включение и отключения курсивного вывода русского символа  +
132       20 A2 AA AB EE E7 A5+  ь', 10, 13
133       AD A8 A5 20 A8 20 AE+
134       E2 AA AB EE E7 A5 AD+
135       A8 EF 20 AA E3 E0 E1+
136       A8 A2 AD AE A3 AE 20+
137       A2 EB A2 AE A4 A0 20+
138       E0 E3 E1 E1 AA AE A3+
139       AE 20 E1 A8 AC A2 AE+
140       AB A0 20 9A 0A 0D
141 0317  20 20 46 31 20 20 2D+        DB '  F1  - включение и отключение частичной русификации клавиатуры    +
142       20 A2 AA AB EE E7 A5+  "ЕЖЗИК"', 10, 13
143       AD A8 A5 20 A8 20 AE+
144       E2 AA AB EE E7 A5 AD+
145       A8 A5 20 E7 A0 E1 E2+
146       A8 E7 AD AE A9 20 E0+
147       E3 E1 A8 E4 A8 AA A0+
148       E6 A8 A8 20 AA AB A0+
149       A2 A8 A0 E2 E3 E0 EB+
150       20 20 22 85 86 87 88+
151       8A 22 0A 0D
152 0361  20 20 46 32 20 20 2D+        DB '  F2  - включение и отключение режима блокировки ввода прописных   +
153       20 A2 AA AB EE E7 A5+  русских букв', 10, 13
154       AD A8 A5 20 A8 20 AE+
155       E2 AA AB EE E7 A5 AD+
156       A8 A5 20 E0 A5 A6 A8+
157       AC A0 20 A1 AB AE AA+
```

```
158       A8 E0 AE A2 AA A8 20+
159       A2 A2 AE A4 A0 20 AF+
160       E0 AE AF A8 E1 AD EB+
161       E5 20 E0 E3 E1 E1 AA+
162       A8 E5 20 A1 E3 AA A2+
163       0A 0D
164       =014B        helpMsgLength        equ  $-helpMsg
165
166 03B0  8E E8 A8 A1 AA A0 20+      errorParamMsg        DB      'Ошибка параметров коммандной строки', +
167       AF A0 E0 A0 AC A5 E2+  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        errorParamMsgLength       equ     $-errorParamMsg
173 03D5  90 A5 A7 A8 A4 A5 AD+      installedMsg        DB  'Резидент загружен$'
174       E2 20 A7 A0 A3 E0 E3+
175       A6 A5 AD 24
176 03E7  90 A5 A7 A8 A4 A5 AD+      alreadyInstalledMsg       DB  'Резидент уже был загружен$'
177       E2 20 E3 A6 A5 20 A1+
178       EB AB 20 A7 A0 A3 E0+
179       E3 A6 A5 AD 24
180 0401  8D A5 A4 AE E1 E2 A0+      noMemMsg          DB  'Недостаточно памяти$'
181       E2 AE E7 AD AE 20 AF+
182       A0 AC EF E2 A8 24
183 0415  8D A5 20 E3 A4 A0 AB+      notInstalledMsg        DB  'Не удалось загрузить резидент$'
184       AE E1 EC 20 A7 A0 A3+
185       E0 E3 A7 A8 E2 EC 20+
186       E0 A5 A7 A8 A4 A5 AD+
187       E2 24
188 0433  90 A5 A7 A8 A4 A5 AD+      removedMsg          DB  'Резидент выгружен'
189       E2 20 A2 EB A3 E0 E3+
190       A6 A5 AD
191       =0011        removedMsgLength        equ     $-removedMsg
192 0444  8D A5 20 E3 A4 A0 AB+      noRemoveMsg         DB  'Не удалось выгрузить резидент'
193       AE E1 EC 20 A2 EB A3+
194       E0 E3 A7 A8 E2 EC 20+
195       E0 A5 A7 A8 A4 A5 AD+
196       E2
197       =001D        noRemoveMsgLength        equ     $-noRemoveMsg
198
199         ;=== Обработчик прерывания int 9h ===;
200 0461        new_int9h proc far
201          ;Сохраняем значения всех, изменяемых регистров в стэке
202 0461  56        push SI
203 0462  50        push AX
204 0463  53        push BX
205 0464  51        push CX
206 0465  52        push DX
207 0466  06        push ES
208 0467  1E        push DS
209          ;Синхронизируем CS и DS
210 0468  0E        push CS
211 0469  1F        pop     DS
212
213 046A  B8 0040        mov     AX, 40h            ;40h-сегмент,  +
214         где хранятся флаги сост-я клавиатуры, кольц. буфер ввода
215 046D  8E C0        mov     ES, AX
216 046F  E4 60        in      AL, 60h            ;Записываем в  +
217         AL скан-код нажатой клавиши
218
219          ;Обработка ctrl + u/U
220 0471  3C 16         cmp     AL, 22           ;Проверка на   +
221         нажатие клавишы u/U
222 0473  75 24         jne     _testFxButtons
223 0475  26: 8A 26 0017        mov     AH, ES:[17h]         ;Флаги клавиатуры
224 047A  80 E4 0F         and     AH, 00001111b
225 047D  80 FC 04         cmp     AH, 00000100b         ;Проверка на нажатие   +
226         клавишы ctrl
227 0480  75 17         jne     _testFxButtons
228
229          ;Выгрузка программы
230 0482  B4 FF         mov AH, 0FFh
231 0484  B0 01         mov AL, 01h
232 0486  CD 2F         int 2Fh            ;Завершаем     +
233         обработку нажатия
234
235 0488  E4 61         in      AL, 61h           ;Контроллер    +
236         состояния клавиатуры
237 048A  0C 80         or      AL, 10000000b         ;Пометим, что клавишу  +
238         нажали
239 048C  E6 61         out     61h, AL
```

```
240 048E  24 7F         and     AL, 01111111b         ;Пометим, что клавишу  +
241         отпустили
242 0490  E6 61         out     61h, AL
243 0492  B0 20         mov     AL, 20h
244 0494  E6 20         out     20h, AL           ;Отправим в    +
245         контроллер прерываний признак конца прерывания
246
247 0496  E9 0091         jmp _quit            ;Выход
248
249 0499         _testFxButtons:           ;Проверка F1, F2, F8, F9
250 0499  2C 3A         sub AL, 58            ;В AL теперь   +
251         номер функциональной клавиши
252 049B           _F8:
253 049B  3C 08          cmp AL, 8 ; F8
254 049D  75 07          jne _F9
255 049F  F6 16 0131r         not isSignatureForm        ;Смена флага     +
256         isSignatureForm
257 04A3  EB 25 90          jmp _startCheck
258 04A6           _F9:
259 04A6  3C 09          cmp AL, 9 ; F9
260 04A8  75 0A          jne _F1
261 04AA  F6 16 0246r         not isCursiveChar         ;Смена флага     +
262         isCursiveChar
263 04AE  E8 01C0          call setCursive         ;Перевод символа в     +
264         курсив и обратно в зависимости от флага isCursiveChar
265 04B1  EB 17 90          jmp _startCheck
266 04B4           _F1:
267 04B4  3C 01          cmp AL, 1 ; F1
268 04B6  75 07          jne _F2
269 04B8  F6 16 012Er         not isTranslateStr         ;Смена флага     +
270         isTranslateStr
271 04BC  EB 0C 90          jmp _startCheck
272 04BF           _F2:
273 04BF  3C 02          cmp AL, 2 ; F2
274 04C1  75 07          jne _startCheck
275 04C3  F6 16 0123r         not isIgnoredRussianChars       ;Смена флага      +
276         isIgnoredRussianChars
277 04C7  EB 01 90          jmp _startCheck
278
279 04CA         _startCheck:
280 04CA  9C          pushf
281 04CB  2E: FF 1E 0257r        call dword ptr CS:[old_int9hOffset] ;Вызываем стандартный обработчик прерывания
282 04D0  B8 0040         mov     AX, 40h           ;40h-сегмент,  +
283         где хранятся флаги сост-я клавы,кольц. буфер ввода
284 04D3  8E C0         mov     ES, AX
285 04D5  26: 8B 1E 001C        mov     BX, ES:[1Ch]         ;Адрес хвоста
286 04DA  4B          dec     BX             +
287         ;Сместимся назад к последнему введённому символу
288 04DB  4B          dec     BX
289 04DC  83 FB 1E          cmp     BX, 1Eh           ;Не вышли ли мы+
290         за пределы буфера?
291 04DF  73 03         jae     _checkIgnored
292 04E1  BB 003C          mov     BX, 3Ch           ;Хвост вышел за+
293         пределы буфера, значит последний введённый символ
294                   +
295         ;Находится      в конце буфера
296 04E4         _checkIgnored:
297 04E4  26: 8B 17         mov DX, ES:[BX]          ;В DX 0 введённый     +
298         символ
299 04E7  80 3E 0123r FF        cmp isIgnoredRussianChars, true      ;Проверка на включенный режим     +
300         блокировки ввода русских символов
301 04EC  75 1A         jne _check_translate
302
303 04EE  BE 0000          mov SI, 0            ;Если режим    +
304         включен
305 04F1  B9 0020         mov CX, ignoredRussianCharsLenght      ;Количество игнорируемых символов
306
307 04F4         _checkIgnoredChars:         ;Проверка на присутсвие     +
308         игнорируемых букв
309 04F4  3A 94 0103r        cmp DL, ignoredRussianChars[SI]
310 04F8  74 06         je _block
311 04FA  46         inc SI
312 04FB  E2 F7         loop _checkIgnoredChars
313 04FD  EB 09 90          jmp _check_translate
314
315 0500         _block:             ;Блокируем     +
316         введенный символ
317 0500  26: 89 1E 001C        mov ES:[1Ch], BX
318 0505  EB 23 90          jmp _quit
319
320 0508         _check_translate:
321 0508  80 3E 012Er FF        cmp isTranslateStr, true        ;Проверка на включенный режим  +
```

```
322         перевода
323 050D  75 1B         jne _quit
324
325 050F  BE 0000         mov SI, 0            ;Если режим    +
326         включен
327 0512  B9 0005         mov CX, translateSirFromLength      ;Количество символов для перевода
328
329 0515        _checkTranslateChars:         ;Проверка на присутсвие     +
330         переводимых букв букв
331 0515  3A 94 0124r        cmp DL, translateStrFrom[SI]
332 0519  74 06         je _translate
333 051B  46         inc SI
334 051C  E2 F7        loop _checkTranslateChars
335 051E  EB 0A 90         jmp _quit
336
337 0521        _translate:            ;Переводим     +
338         введенный символ
339 0521  33 C0         xor AX, AX
340 0523  8A 84 0129r        mov AL, translateStrTo[SI]
341 0527  26: 89 07         mov ES:[BX], AX          ;Замена символа
342
343 052A        _quit:
344           ;Восстанавливаем все регистры
345 052A  1F         pop     DS
346 052B  07         pop     ES
347 052C  5A         pop DX
348 052D  59         pop CX
349 052E  5B         pop     BX
350 052F  58         pop     AX
351 0530  5E         pop SI
352 0531  CF         iret
353 0532       new_int9h endp
354
355         ;=== Обработчик прерывания int 1Ch ===;
356 0532       new_int1Ch proc far
357 0532  50        push AX
358 0533  0E        push CS
359 0534  1F        pop DS
360
361 0535  9C        pushf
362 0536  2E: FF 1E 025Br       call dword ptr CS:[old_int1ChOffset]
363
364 053B  80 3E 0131r FF       cmp isSignatureForm, true         ;Проверка на включенный режим  +
365         вывода окна автора
366 0540  75 1C        jne _dontIsSugnatureForm
367 0542  83 3E 012Fr 5B        cmp signatureCounter, signatureFormDelay*1000/55 + 1 ;Если кол-во "тактов"     +
368         эквивалентно %signatureFormDelay% секундам
369 0547  74 03         je _printSignatureForm
370 0549  EB 0E 90         jmp _dontPrintSignatureForm
371
372 054C         _printSignatureForm:        ;Вывод формы автора
373 054C  F6 16 0131r         not isSignatureForm        ;Установка флага     +
374         isSignatureForm
375 0550  C7 06 012Fr 0000         mov signatureCounter, 0
376 0556  E8 0094          call printSignature
377
378 0559         _dontPrintSignatureForm:        ;Форма автора не выводится
379 0559  83 06 012Fr 01         add signatureCounter, 1
380
381 055E        _dontIsSugnatureForm:
382
383 055E  58        pop AX
384
385 055F  CF        iret
386 0560       new_int1Ch endp
387
388         ;=== Обработчик прерывания int 2Fh ===;
389 0560       new_int2Fh proc
390 0560  80 FC FF        cmp     AH, 0FFh            ;Проверка на   +
391         нашу функцию
392 0563  75 0B        jne     _2Fh_std            ;Если нет, то  +
393         переходим на старый обработчик
394 0565  3C 00        cmp     AL, 0            ;Проверка     +
395         загрузки резидента в память
396 0567  74 0C        je      _already_installed          ;Вывод информации о    +
397         загрузки резидента
398 0569  3C 01        cmp     AL, 1            ;подфункция    +
399         выгрузки из памяти?
400 056B  74 0B        je      _uninstall
401 056D  EB 01 90        jmp     _2Fh_std            ;Если нет, то  +
402         переходим на старый обработчик
403
```

```
0570        _2Fh_std:
0570  2E: FF 2E 025Fr       jmp     dword ptr CS:[old_int2FhOffset]      ;Вызов старого обработчика

0575        _already_installed:
0575  B4 69         mov     AH, 'i'            ;Вернём 'i',   +
        если резидент загружен в память
0577  CF         iret

0578        _uninstall:
0578  1E         push    DS
0579  06         push    ES
057A  52         push    DX
057B  53         push    BX

057C  33 DB         xor BX, BX

          ;CS = ES, для доступа к переменным
057E  0E         push CS
057F  07         pop ES

0580  B8 2509         mov     AX, 2509h
0583  26: 8B 16 0257r       mov DX, ES:old_int9hOffset        ;Возвращаем вектор прерывания
0588  26: 8E 1E 0259r   mov DS, ES:old_int9hSegment       ;на место
058D  CD 21         int     21h

058F  B8 251C         mov     AX, 251Ch
0592  26: 8B 16 025Br       mov DX, ES:old_int1ChOffset      ;Возвращаем вектор прерывания
0597  26: 8E 1E 025Dr   mov DS, ES:old_int1ChSegment       ;на место
059C  CD 21         int     21h

059E  B8 252F         mov     AX, 252Fh
05A1  26: 8B 16 025Fr       mov DX, ES:old_int2FhOffset      ;Возвращаем вектор прерывания
05A6  26: 8E 1E 0261r   mov DS, ES:old_int2FhSegment       ;на место
05AB  CD 21         int     21h

05AD  2E: 8E 06 002C       mov     ES, CS:2Ch             ;Загрузим в ES +
        адрес окружения
05B2  B4 49         mov     AH, 49h            ;Выгрузим из   +
        памяти окружение
05B4  CD 21         int     21h
05B6  72 0B         jc _notRemove

05B8  0E         push CS
05B9  07         pop     ES              ;В ES -+
        адрес резидентной программы
05BA  B4 49         mov     AH, 49h            ;Выгрузим из   +
        памяти резидент
05BC  CD 21         int     21h
05BE  72 03         jc _notRemove
05C0  EB 15 90         jmp _unloaded

05C3        _notRemove:            ;Не удалось выполнить  +
        выгрузку
     ;Вывод сообщения о неудачной выгрузке
05C3  B4 03         mov AH, 03h            ;Получаем     +
        позицию курсора
05C5  CD 10         int 10h
05C7  BD 0444r         lea BP, noRemoveMsg
05CA  B9 001D         mov CX, noRemoveMsgLength
05CD  B3 07         mov BL, 0111b
05CF  B8 1301         mov AX, 1301h
05D2  CD 10         int 10h
05D4  EB 12 90         jmp _2Fh_exit

05D7        _unloaded:              ;Выгрузка     +
        прошла успешно
     ;Вывод сообщения об удачной выгрузке
05D7  B4 03         mov AH, 03h            ;Получаем     +
        позицию курсора
05D9  CD 10         int 10h
05DB  BD 0433r         lea BP, removedMsg
05DE  B9 0011         mov CX, removedMsgLength
05E1  B3 07         mov BL, 0111b
05E3  B8 1301         mov AX, 1301h
05E6  CD 10         int 10h

05E8        _2Fh_exit:
05E8  5B         pop BX
05E9  5A         pop     DX
05EA  07         pop     ES
05EB  1F         pop     DS
05EC  CF         iret
```

```
05ED        new_int2Fh endp

        ;=== Процедура вывода подписи ===;
05ED        printSignature proc
05ED  50        push AX
05EE  52        push DX
05EF  51        push CX
05F0  53        push BX
05F1  06        push ES
05F2  54        push SP
05F3  55        push BP
05F4  56        push SI
05F5  57        push DI

05F6  33 C0        xor AX, AX
05F8  33 DB        xor BX, BX
05FA  33 D2        xor DX, DX

05FC  B4 03        mov AH, 03h             ;Чтение текущей+
        позиции курсора
05FE  CD 10        int 10h
0600  52        push DX             ;Помещаем     +
        информацию о положении курсора в стек

0601  B6 00        mov DH, 0              ;Установка     +
        позиции на экране
0603  B2 0F        mov DL, 15

0605  B4 0F        mov AH, 0Fh             ;Чтение     +
        текущего видеорежима. в BH - текущая страница
0607  CD 10        int 10h

0609  0E        push CS
060A  07        pop ES              ;Указываем ES  +
        на CS

          ;Вывод 'верхушки' таблицы
060B  52         push DX
060C  BD 01CEr         lea BP, tableTop          ;Помещаем в BP     +
        указатель на выводимую строку
060F  B9 0034         mov CX, tableTopLength        ;В CX - длина строки
0612  B3 07         mov BL, 0111b          ;Цвет выводимого текста
0614  B8 1301         mov AX, 1301h          ;AH=13h - номер ф-ии,  +
        AL=01h - курсор перемещается при выводе каждого из символов строки
0617  CD 10         int 10h
0619  5A         pop DX
061A  FE C6         inc DH

          ;Вывод первой линии
061C  52         push DX
061D  BD 0132r         lea BP, signatureLine1
0620  B9 0034         mov CX, signatureLine1Length
0623  B3 07         mov BL, 0111b
0625  B8 1301         mov AX, 1301h
0628  CD 10         int 10h
062A  5A         pop DX
062B  FE C6         inc DH

          ;Вывод второй линии
062D  52         push DX
062E  BD 0166r         lea BP, signatureLine2
0631  B9 0034         mov CX, signatureLine2Length
0634  B3 07         mov BL, 0111b
0636  B8 1301         mov AX, 1301h
0639  CD 10         int 10h
063B  5A         pop DX
063C  FE C6         inc DH

          ;Вывод третьей линии
063E  52         push DX
063F  BD 019Ar         lea BP, signatureLine3
0642  B9 0034         mov CX, signatureLine3Length
0645  B3 07         mov BL, 0111b
0647  B8 1301         mov AX, 1301h
064A  CD 10         int 10h
064C  5A         pop DX
064D  FE C6         inc DH

          ;Вывод 'низа' таблицы
064F  52         push DX
0650  BD 0202r         lea BP, tableBottom
0653  B9 0034         mov CX, tableBottomLength
```

```
568 0656  B3 07         mov BL, 0111b
569 0658  B8 1301         mov AX, 1301h
570 065B  CD 10         int 10h
571 065D  5A         pop DX
572 065E  FE C6         inc DH
573
574 0660  33 DB        xor BX, BX
575 0662  5A        pop DX              +
576         ;Восстанавливаем из стека прежнее положение курсора
577 0663  B4 02        mov AH, 02h             ;Меняем+
578         положение курсора на первоначальное
579 0665  CD 10        int 10h
580
581 0667  5F        pop DI
582 0668  5E        pop SI
583 0669  5D        pop BP
584 066A  5C        pop SP
585 066B  07        pop ES
586 066C  5B        pop BX
587 066D  59        pop CX
588 066E  5A        pop DX
589 066F  58        pop AX
590
591 0670  C3        ret
592 0671       printSignature endp
593
594         ;=== Процедура смены начертания символа ===;
595 0671        setCursive proc
596 0671  06        push ES              +
597         ;Сохраняем регистры
598 0672  50        push AX
599 0673  0E        push CS
600 0674  07        pop ES
601
602 0675  80 3E 0246r FF       cmp isCursiveChar, true          ;Проверка на включенный+
603         режим курсива буквы
604 067A  75 30        jne _restoreSymbol
605
606 067C  E8 004C        call saveFont            ;Получаем     +
607         таблицу
608 067F  8A 0E 0236r       mov CL, charToCursiveIndex
609 0683        _shifTtable:             ;Мы получаем в +
610         BP таблицу всех символов. адрес указывает на символ 0
611 0683  83 C5 10        add BP, 16              +
612         ;Поэтому нуэно совершить сдвиг 16*X - где X - код символа
613 0686  E2 FB        loop _shiftTable
614
615          ;Swap(ES, DS) и сохранение старого значения DS
616 0688  1E        push DS
617 0689  58        pop AX
618 068A  06        push ES
619 068B  1F        pop DS
620 068C  50        push AX
621 068D  07        pop ES
622 068E  50        push AX
623
624 068F  8B F5        mov SI, BP
625 0691  BF 0247r        lea DI, savedSymbol           ;Сохраняем в   +
626         переменную savedSymbol
627 0694  B9 0010        mov CX, 16              +
628         ;Таблицу нужного символа
629 0697  F3> A4        rep movsb              ;Movsb +
630         из DS:SI в ES:DI
631 0699  1F        pop DS              +
632         ;Восстановление DS
633
634          ;Замена написания символа на курсив
635 069A  B9 0001        mov CX, 1
636 069D  B6 00        mov DH, 0
637 069F  8A 16 0236r       mov DL, charToCursiveIndex
638 06A3  BD 0237r        lea BP, cursiveSymbol
639 06A6  E8 0015        call changeFont
640 06A9  EB 10 90        jmp _exitSetCursive
641
642 06AC        _restoreSymbol:             ;Если флаг     +
643         равен 0, выполняем замену курсивного символа на старый вариант
644 06AC  B9 0001        mov CX, 1
645 06AF  B6 00        mov DH, 0
646 06B1  8A 16 0236r       mov DL, charToCursiveIndex
647 06B5  BD 0247r        lea bp, savedSymbol
648 06B8  E8 0003        call changeFont
649
```

```
650 06BB        _exitSetCursive:
651 06BB  58        pop AX
652 06BC  07        pop ES
653 06BD  C3        ret
654 06BE        setCursive endp
655
656 06BE        changeFont proc
657 06BE  50        push AX
658 06BF  53        push BX
659 06C0  B8 1100       mov AX, 1100h
660 06C3  BB 1000       mov BX, 1000h
661 06C6  CD 10       int 10h
662 06C8  58        pop AX
663 06C9  5B        pop BX
664 06CA  C3        ret
665 06CB        changeFont endp
666
667 06CB        saveFont proc
668 06CB  50        push AX
669 06CC  53        push BX
670 06CD  B8 1130       mov AX, 1130h
671 06D0  BB 0600       mov BX, 0600h
672 06D3  CD 10       int 10h
673 06D5  58        pop AX
674 06D6  5B        pop BX
675 06D7  C3        ret
676 06D8        saveFont endp
677
678
679         ;=== Начало работы резидента ===;
680 06D8        _initTSR:
681 06D8  B4 03       mov AH, 03h
682 06DA  CD 10       int 10h
683 06DC  52        push DX
684 06DD  B4 00       mov AH, 00h          ;Установка видеорежима (83h    +
685         текст  80x25  16/8  CGA,EGA  b800 Comp,RGB,Enhanced), без очистки экрана
686 06DF  B0 83       mov AL, 83h
687 06E1  CD 10       int 10h
688 06E3  5A        pop DX
689 06E4  B4 02       mov AH, 02h
690 06E6  CD 10       int 10h
691
692 06E8  E8 0092    call commandParamsParser      ;Обработка параметров командной строки
693 06EB  B8 3509       mov AX, 3509h        ;Получить в ES:BX вектор 09
694 06EE  CD 21    int 21h        ;Прерывания
695
696 06F0  80 3E 0264r FF       cmp notLoadTSR, true       ;Если были введены параметры командной строки
697 06F5  74 0E       je _exit_tmp
698
699 06F7  B4 FF       mov AH, 0FFh
700 06F9  B0 00       mov AL, 0
701 06FB  CD 2F       int 2Fh
702 06FD  80 FC 69       cmp AH, 'i'         ;Проверка того, загружена ли уже     +
703         программа
704 0700  74 68       je _alreadyInstalled
705
706 0702  EB 04 90       jmp _tmp
707
708 0705        _exit_tmp:
709 0705  EB 74 90        jmp _exit
710
711 0708        _tmp:
712 0708  06        push ES
713 0709  A1 002C    mov AX, DS:[2Ch]
714 070C  8E C0    mov ES, AX
715 070E  B4 49    mov AH, 49h    ;Проверка на наличие памяти
716 0710  CD 21    int 21h
717 0712  07    pop ES
718 0713  72 5F    jc _notMem    ;Выход при нехватке памяти
719
720          ;== int 09h ==;
721 0715  2E: 89 1E 0257r       mov     word ptr CS:old_int9hOffset, BX
722 071A  2E: 8C 06 0259r       mov     word ptr CS:old_int9hSegment, ES
723 071F  B8 2509    mov AX, 2509h    ;Установим вектор на 09
724 0722  BA 0461r    mov DX, offset new_int9h  ;Прерывание
725 0725  CD 21    int 21h
726
727          ;== int 1Ch ==;
728 0727  B8 351C       mov AX,351Ch       ;Получить в ES:BX вектор 1C
729 072A  CD 21    int 21h    ;Прерывания
730 072C  2E: 89 1E 025Br       mov     word ptr CS:old_int1ChOffset, BX
731 0731  2E: 8C 06 025Dr       mov     word ptr CS:old_int1ChSegment, ES
```

```
732 0736  B8 251C        mov AX, 251Ch       ;Установим вектор на 1C
733 0739  BA 0532r        mov DX, offset new_int1Ch      ;Прерывание
734 073C  CD 21        int 21h
735
736          ;== int 2Fh ==;
737 073E  B8 352F        mov AX,352Fh       ;Получить в ES:BX вектор 1C
738 0741  CD 21    int 21h    ;Прерывания
739 0743  2E: 89 1E 025Fr       mov     word ptr CS:old_int2FhOffset, BX
740 0748  2E: 8C 06 0261r       mov     word ptr CS:old_int2FhSegment, ES
741 074D  B8 252F        mov AX, 252Fh       ;Установим вектор на 2F
742 0750  BA 0560r        mov DX, offset new_int2Fh      ;Прерывание
743 0753  CD 21        int 21h
744
745 0755  BA 03D5r    mov DX, offset installedMsg  ;Вывод сообщения о загрузке программы
746 0758  B4 09    mov AH, 9
747 075A  CD 21    int 21h
748 075C  BA 06D8r    mov DX, offset _initTSR      ;Остаемся в памяти резидентом
749 075F  CD 27    int 27h    ;Выход
750
751          ;Конец основной программы
752 0761        _remove:            ;Выгрузка программы из памяти
753 0761  B4 FF        mov AH, 0FFh
754 0763  B0 01        mov AL, 1
755 0765  CD 2F        int 2Fh
756 0767  EB 12 90        jmp _exit
757
758 076A        _alreadyInstalled:
759 076A  B4 09        mov AH, 09h
760 076C  BA 03E7r        lea DX, alreadyInstalledMsg
761 076F  CD 21        int 21h
762 0771  EB 08 90        jmp _exit
763
764 0774        _notMem:     ;Не хватает памяти, чтобы остаться резидентом
765 0774  BA 0401r    mov DX, offset noMemMsg
766 0777  B4 09    mov AH, 9
767 0779  CD 21    int 21h
768
769 077B        _exit:     ;Выход
770 077B  CD 20    int 20h
771
772         ;=== Процедура проверки параметров командной строки ===;
773 077D        commandParamsParser proc
774 077D  0E         push CS
775 077E  07         pop ES
776 077F  C6 06 0263r 00       mov unloadTSR, 0
777 0784  C6 06 0264r 00       mov notLoadTSR, 0
778
779 0789  BE 0080        mov SI, 80h        ;SI = смещение командной строки.
780 078C  AC         lodsb         ;Получим кол-во символов.
781 078D  0A C0        or AL, AL         ;Если 0 символов введено,
782 078F  74 3E        jz _exitParser        ;то все в порядке.
783
784 0791  46        inc SI         ;Теперь SI указывает на первый символ строки.
785
786
787 0792  80 3C 0D        cmp [SI], BYTE ptr 13
788 0795  74 38        je _exitParser
789 0797  AD         lodsw        ;Получаем два символа
790
791 0798  3D 3F2F         cmp AX, '?/'       ;Проверка на '/?'
792 079B  74 05         je _question
793 079D  75 1B         jne _errorParam
794
795 079F  EB 2E 90         jmp _exitParser       ;Выхох из парсера в случае отстутствия     +
796         параметров
797
798
799      ;Вывод строки помощи
800 07A2         _question:
801 07A2  B4 03          mov AH,03
802 07A4  CD 10          int 10h
803 07A6  BD 0265r          lea BP, helpMsg
804 07A9  B9 014B          mov CX, helpMsgLength
805 07AC  B3 07          mov BL, 0111b
806 07AE  B8 1301          mov AX, 1301h
807 07B1  CD 10          int 10h
808
809 07B3  F6 16 0264r       not notLoadTSR
810
811 07B7  EB 16 90        jmp _exitParser
812
813          ;Вывод строки ошибки
```

```
    814 07BA         _errorParam:
    815 07BA  B4 03          mov AH,03
    816 07BC  CD 10          int 10h
    817 07BE  BD 03B0r          lea BP, CS:errorParamMsg
    818 07C1  B9 0025         mov CX, errorParamMsgLength
    819 07C4  B3 07          mov BL, 0111b
    820 07C6  B8 1301          mov AX, 1301h
    821 07C9  CD 10          int 10h
    822
    823 07CB  F6 16 0264r       not notLoadTSR
    824
    825 07CF         _exitParser:
    826 07CF  C3        ret
    827 07D0        commandParamsParser endp
    828
    829 07D0        code ends
    830         end _start
Symbol Table



Symbol Name     Type  Value

??DATE      Text  "05/22/20"
??FILENAME     Text  "tsr   "
??TIME      Text  "17:48:24"
??VERSION     Number 030A
@CPU      Text  0101H
@CURSEG      Text  CODE
@FILENAME     Text  TSR
@WORDSIZE     Text  2
ALREADYINSTALLEDMSG    Byte  CODE:03E7
CHANGEFONT     Near  CODE:06BE
CHARTOCURSIVEINDEX    Byte  CODE:0236
COMMANDPARAMSPARSER    Near  CODE:077D
CURSIVESYMBOL     Byte  CODE:0237
ERRORPARAMMSG     Byte  CODE:03B0
ERRORPARAMMSGLENGTH    Number 0025
HELPMSG      Byte  CODE:0265
HELPMSGLENGTH     Number 014B
IGNOREDRUSSIANCHARS    Byte  CODE:0103
IGNOREDRUSSIANCHARSLENGHT   Number 0020
INSTALLEDMSG     Byte  CODE:03D5
ISCURSIVECHAR     Byte  CODE:0246
ISIGNOREDRUSSIANCHARS    Byte  CODE:0123
ISSIGNATUREFORM     Byte  CODE:0131
ISTRANSLATESTR     Byte  CODE:012E
NEW_INT1CH     Far  CODE:0532
NEW_INT2FH     Near  CODE:0560
NEW_INT9H     Far  CODE:0461
NOMEMMSG     Byte  CODE:0401
NOREMOVEMSG     Byte  CODE:0444
NOREMOVEMSGLENGTH    Number 001D
NOTINSTALLEDMSG     Byte  CODE:0415
NOTLOADTSR     Byte  CODE:0264
OLD_INT1CHOFFSET    Word  CODE:025B
OLD_INT1CHSEGMENT    Word  CODE:025D
OLD_INT2FHOFFSET    Word  CODE:025F
OLD_INT2FHSEGMENT    Word  CODE:0261
OLD_INT9HOFFSET     Word  CODE:0257
OLD_INT9HSEGMENT    Word  CODE:0259
PRINTSIGNATURE     Near  CODE:05ED
REMOVEDMSG     Byte  CODE:0433
REMOVEDMSGLENGTH    Number 0011
SAVEDSYMBOL     Byte  CODE:0247
SAVEFONT     Near  CODE:06CB
SETCURSIVE     Near  CODE:0671
SIGNATURECOUNTER    Word  CODE:012F
SIGNATUREFORMDELAY    Number 0005
SIGNATURELINE1     Byte  CODE:0132
SIGNATURELINE1LENGTH    Number 0034
SIGNATURELINE2     Byte  CODE:0166
SIGNATURELINE2LENGTH    Number 0034
SIGNATURELINE3     Byte  CODE:019A
SIGNATURELINE3LENGTH    Number 0034
TABLEBOTTOM     Byte  CODE:0202
TABLEBOTTOMLENGTH    Number 0034
TABLETOP     Byte  CODE:01CE
TABLETOPLENGTH     Number 0034
TRANSLATESIRFROMLENGTH    Number 0005
TRANSLATESTRFROM    Byte  CODE:0124
```

```
TRANSLATESTRTO     Byte  CODE:0129
TRUE       Number 00FF
UNLOADTSR      Byte  CODE:0263
_2FH_EXIT      Near  CODE:05E8
_2FH_STD     Near  CODE:0570
_ALREADYINSTALLED     Near  CODE:076A
_ALREADY_INSTALLED     Near  CODE:0575
_BLOCK       Near  CODE:0500
_CHECKIGNORED     Near  CODE:04E4
_CHECKIGNOREDCHARS     Near  CODE:04F4
_CHECKTRANSLATECHARS     Near  CODE:0515
_CHECK_TRANSLATE     Near  CODE:0508
_DONTISSUGNATUREFORM     Near  CODE:055E
_DONTPRINTSIGNATUREFORM     Near  CODE:0559
_ERRORPARAM      Near  CODE:07BA
_EXIT       Near  CODE:077B
_EXITPARSER      Near  CODE:07CF
_EXITSETCURSIVE      Near  CODE:06BB
_EXIT_TMP      Near  CODE:0705
_F1       Near  CODE:04B4
_F2       Near  CODE:04BF
_F8       Near  CODE:049B
_F9       Near  CODE:04A6
_INITTSR      Near  CODE:06D8
_NOTMEM       Near  CODE:0774
_NOTREMOVE      Near  CODE:05C3
_PRINTSIGNATUREFORM     Near  CODE:054C
_QUESTION      Near  CODE:07A2
_QUIT       Near  CODE:052A
_REMOVE       Near  CODE:0761
_RESTORESYMBOL      Near  CODE:06AC
_SHIFTTABLE      Near  CODE:0683
_START       Near  CODE:0100
_STARTCHECK      Near  CODE:04CA
_TESTFXBUTTONS      Near  CODE:0499
_TMP       Near  CODE:0708
_TRANSLATE      Near  CODE:0521
_UNINSTALL      Near  CODE:0578
_UNLOADED      Near  CODE:05D7

Groups & Segments     Bit Size Align  Combine Class

CODE       16  07D0 Para   none   CODE
```