Московский государственный технический университет

им. Н.Э. Баумана

|  |  |
| --- | --- |
| УТВЕРЖДАЮ: |  |
| Большаков С.А. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_201**X** г. |

**Курсовая работа по курсу «Системное программирование»**

Исходный текст программного продукта

(вид документа)

писчая бумага

(вид носителя)

21

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛИ: |  |
| студенты группы ИУ5-44 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Большаков С.А. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_201**X** г. |

Москва – 201**X**

**Содержание**

[**1.** **Файл tsr.lst** 3](#_Toc778704)

[**2.** **Файл unloader.lst** 20](#_Toc778705)

1. **Файл tsr.lst**

Turbo Assembler Version 3.1

tsr.asm

1 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

2 ; tsr.asm

3 ;

4 ; Сборка:

5 ; tasm.exe /l tsr.asm

6 ; tlink /t /x tsr.obj

7 ;

8 ; Примечания:

9 ; 1) комменатрии, 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

18 0000 code segment 'code'

19 assume CS:code, DS:code

20 org 100h

21 0100 \_start:

22

23 0100 E9 06D8 jmp \_initTSR ; на начало программы

24

25 ; данные

26 0103 61 62 63 64 65 66 67+ ignoredChars DB 'abcdefghijklmnopqrstuvwxyz' +

27 68 69 6A 6B 6C 6D 6E+ ;@ список игнорируемых символов

28 6F 70 71 72 73 74 75+

29 76 77 78 79 7A

30 =001A ignoredLength equ $-ignoredChars +

31 ; длина строки ignoredChars

32 011D 00 ignoreEnabled DB 0 +

33 ; флаг функции игнорирования ввода

34 011E 46 3C 44 55 4C translateFrom DB 'F<DUL' +

35 ;@ символы для замены (АБВГД на англ. раскладке)

36 0123 80 81 82 83 84 translateTo DB 'АБВГД' +

37 ;@ символы на которые будет идти замена

38 =0005 translateLength equ $-translateTo +

39 ; длина строки trasnlateFrom

40 0128 00 translateEnabled DB 0 +

41 ; флаг функции перевода

42

43 0129 00 signaturePrintingEnabled DB 0 +

44 ; флаг функции вывода информации об авторе

45 012A 00 cursiveEnabled DB 0 +

46 ; флаг перевода символа в курсив

47

48 012B 00 cursiveSymbol DB 00000000b +

49 ;@ символ, составленный из единичек (его курсивный вариант)

50 012C 00 DB 00000000b

51 012D 00 DB 00000000b

52 012E 3E DB 00111110b

53 012F 3F DB 00111111b

54 0130 33 DB 00110011b

55 0131 66 DB 01100110b

56 0132 66 DB 01100110b

57 0133 7C DB 01111100b

58 0134 C6 DB 11000110b

59 0135 C6 DB 11000110b

60 0136 C6 DB 11000110b

61 0137 FC DB 11111100b

62 0138 00 DB 00000000b

63 0139 00 DB 00000000b

64 013A 00 DB 00000000b

65

66 013B 82 charToCursiveIndex DB 'В' +

67 ;@ символ для замены

68 013C 10\*(FF) savedSymbol DB 16 dup(0FFh) +

69 ; переменная для хранения старого символа

70

71 =00FF true equ 0FFh +

72 ; константа истинности

73 014C ???? old\_int9hOffset DW ? +

74 ; адрес старого обработчика int 9h

75 014E ???? old\_int9hSegment DW ? +

76 ; сегмент старого обработчика int 9h

77 0150 ???? old\_int1ChOffset DW ? +

78 ; адрес старого обработчика int 1Ch

79 0152 ???? old\_int1ChSegment DW ? +

80 ; сегмент старого обработчика int 1Ch

81 0154 ???? old\_int2FhOffset DW ? +

82 ; адрес старого обработчика int 2Fh

83 0156 ???? old\_int2FhSegment DW ? +

84 ; сегмент старого обработчика int 2Fh

85

86 0158 00 unloadTSR DB 0 +

87 ; 1 - выгрузить резидент

88 0159 00 notLoadTSR DB 0 +

89 ; 1 - не загружать

90 015A 0000 counter DW 0

91 =0002 printDelay equ 2 +

92 ;@ задержка перед выводом "подписи" в секундах

93 015C 0001 printPos DW 1 +

94 ;@ положение подписи на экране. 0 - верх, 1 - центр, 2 - низ

95

96 ;@ заменить на собственные данные. формирование таблицы идет по строке большей длины +

97 (1я строка).

98 015E B3 88 A3 AE E0 EC 20+ signatureLine1 DB 179, 'Игорь Латкин, Алексей Леонтьев, +

99 8B A0 E2 AA A8 AD 2C+ Константин Назаров', 179

100 20 80 AB A5 AA E1 A5+

101 A9 20 8B A5 AE AD E2+

102 EC A5 A2 2C 20 8A AE+

103 AD E1 E2 A0 AD E2 A8+

104 AD 20 8D A0 A7 A0 E0+

105 AE A2 B3

106 =0034 Line1\_length equ $-signatureLine1

107 0192 B3 88 93 35 2D 34 34+ signatureLine2 DB 179, 'ИУ5-44 +

108 20 20 20 20 20 20 20+ ', 179

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 20 20 20 20 20+

114 20 20 B3

115 =0034 Line2\_length equ $-signatureLine2

116 01C6 B3 82 A0 E0 A8 A0 AD+ signatureLine3 DB 179, 'Вариант #0 +

117 E2 20 23 30 20 20 20+ ', 179

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 20 20 20 20 20+

123 20 20 B3

124 =0034 Line3\_length equ $-signatureLine3

125 01FA 3E 74 73 72 2E 63 6F+ helpMsg DB '>tsr.com [/?] [/u]', 10, 13

126 6D 20 5B 2F 3F 5D 20+

127 5B 2F 75 5D 0A 0D

128 020E 20 5B 2F 3F 5D 20 2D+ DB ' [/?] - вывод данной справки', 10, 13

129 20 A2 EB A2 AE A4 20+

130 A4 A0 AD AD AE A9 20+

131 E1 AF E0 A0 A2 AA A8+

132 0A 0D

133 022C 20 5B 2F 75 5D 20 2D+ DB ' [/u] - выгрузка резидента из памяти', 10, 13

134 20 A2 EB A3 E0 E3 A7+

135 AA A0 20 E0 A5 A7 A8+

136 A4 A5 AD E2 A0 20 A8+

137 A7 20 AF A0 AC EF E2+

138 A8 0A 0D

139 0252 20 20 46 31 20 20 2D+ DB ' F1 - вывод ФИО и группы по таймеру в центре экрана', 10, 13

140 20 A2 EB A2 AE A4 20+

141 94 88 8E 20 A8 20 A3+

142 E0 E3 AF AF EB 20 AF+

143 AE 20 E2 A0 A9 AC A5+

144 E0 E3 20 A2 20 E6 A5+

145 AD E2 E0 A5 20 ED AA+

146 E0 A0 AD A0 0A 0D

147 0289 20 20 46 32 20 20 2D+ DB ' F2 - включение и отключения курсивного вывода русского символа +

148 20 A2 AA AB EE E7 A5+ В', 10, 13

149 AD A8 A5 20 A8 20 AE+

150 E2 AA AB EE E7 A5 AD+

151 A8 EF 20 AA E3 E0 E1+

152 A8 A2 AD AE A3 AE 20+

153 A2 EB A2 AE A4 A0 20+

154 E0 E3 E1 E1 AA AE A3+

155 AE 20 E1 A8 AC A2 AE+

156 AB A0 20 82 0A 0D

157 02CE 20 20 46 33 20 20 2D+ DB ' F3 - включение и отключение частичной русификации клавиатуры +

158 20 A2 AA AB EE E7 A5+ (F<DUL -> АБВГД)', 10, 13

159 AD A8 A5 20 A8 20 AE+

160 E2 AA AB EE E7 A5 AD+

161 A8 A5 20 E7 A0 E1 E2+

162 A8 E7 AD AE A9 20 E0+

163 E3 E1 A8 E4 A8 AA A0+

164 E6 A8 A8 20 AA AB A0+

165 A2 A8 A0 E2 E3 E0 EB+

166 28 46 3C 44 55 4C 20+

167 2D 3E 20 80 81 82 83+

168 84 29 0A 0D

169 031F 20 20 46 34 20 20 2D+ DB ' F4 - включение и отключение режима блокировки ввода латинских +

170 20 A2 AA AB EE E7 A5+ строчных букв', 10, 13

171 AD A8 A5 20 A8 20 AE+

172 E2 AA AB EE E7 A5 AD+

173 A8 A5 20 E0 A5 A6 A8+

174 AC A0 20 A1 AB AE AA+

175 A8 E0 AE A2 AA A8 20+

176 A2 A2 AE A4 A0 20 AB+

177 A0 E2 A8 AD E1 AA A8+

178 E5 20 E1 E2 E0 AE E7+

179 AD EB E5 20 A1 E3 AA+

180 A2 0A 0D

181

182 =0175 helpMsg\_length equ $-helpMsg

183 036F 8E E8 A8 A1 AA A0 20+ errorParamMsg DB 'Ошибка параметров коммандной +

184 AF A0 E0 A0 AC A5 E2+ строки', 10, 13

185 E0 AE A2 20 AA AE AC+

186 AC A0 AD A4 AD AE A9+

187 20 E1 E2 E0 AE AA A8+

188 0A 0D

189 =0025 errorParamMsg\_length equ $-errorParamMsg

190

191 0394 DA 32\*(C4) BF tableTop DB 218, Line1\_length-2 dup+

192 (196), 191

193 =0034 tableTop\_length equ $-tableTop

194 03C8 C0 32\*(C4) D9 tableBottom DB 192, Line1\_length-2 dup (196), +

195 217

196 =0034 tableBottom\_length equ $-tableBottom

197

198 ; сообщения

199 03FC 90 A5 A7 A8 A4 A5 AD+ installedMsg DB 'Резидент загружен!$'

200 E2 20 A7 A0 A3 E0 E3+

201 A6 A5 AD 21 24

202 040F 90 A5 A7 A8 A4 A5 AD+ alreadyInstalledMsg DB 'Резидент уже загружен$'

203 E2 20 E3 A6 A5 20 A7+

204 A0 A3 E0 E3 A6 A5 AD+

205 24

206 0425 8D A5 A4 AE E1 E2 A0+ noMemMsg DB 'Недостаточно памяти$'

207 E2 AE E7 AD AE 20 AF+

208 A0 AC EF E2 A8 24

209 0439 8D A5 20 E3 A4 A0 AB+ notInstalledMsg DB 'Не удалось загрузить резидент$'

210 AE E1 EC 20 A7 A0 A3+

211 E0 E3 A7 A8 E2 EC 20+

212 E0 A5 A7 A8 A4 A5 AD+

213 E2 24

214

215 0457 90 A5 A7 A8 A4 A5 AD+ removedMsg DB 'Резидент выгружен'

216 E2 20 A2 EB A3 E0 E3+

217 A6 A5 AD

218 =0011 removedMsg\_length equ $-removedMsg

219

220 0468 8D A5 20 E3 A4 A0 AB+ noRemoveMsg DB 'Не удалось выгрузить резидент'

221 AE E1 EC 20 A2 EB A3+

222 E0 E3 A7 A8 E2 EC 20+

223 E0 A5 A7 A8 A4 A5 AD+

224 E2

225 =001D noRemoveMsg\_length equ $-noRemoveMsg

226

227 0485 46 31 f1\_txt DB 'F1'

228 0487 46 32 f2\_txt DB 'F2'

229 0489 46 33 f3\_txt DB 'F3'

230 048B 46 34 f4\_txt DB 'F4'

231 =0002 fx\_length equ $-f4\_txt

232

233 048D changeFx proc

234 048D 50 push AX

235 048E 53 push BX

236 048F 51 push CX

237 0490 52 push DX

238 0491 55 push BP

239 0492 06 push ES

240 0493 33 DB xor BX, BX

241

242 0495 B4 03 mov AH, 03h

243 0497 CD 10 int 10h

244 0499 52 push DX

245

246 049A 0E push CS

247 049B 07 pop ES

248

249 049C \_checkF1:

250 049C BD 0485r lea BP, f1\_txt

251 049F B9 0002 mov CX, fx\_length

252 04A2 B7 00 mov BH, 0

253 04A4 B6 00 mov DH, 0

254 04A6 B2 4E mov DL, 78

255 04A8 B8 1301 mov AX, 1301h

256

257 04AB 80 3E 0129r FF cmp signaturePrintingEnabled, true

258 04B0 74 07 je \_greenF1

259

260 04B2 \_redF1:

261 04B2 B3 4F mov BL, 01001111b ; red

262 04B4 CD 10 int 10h

263 04B6 EB 08 90 jmp \_checkF2

264

265 04B9 \_greenF1:

266 04B9 BD 0485r lea BP, f1\_txt

267 04BC B3 2F mov BL, 00101111b ; green

268 04BE CD 10 int 10h

269

270 04C0 \_checkF2:

271 04C0 BD 0487r lea BP, f2\_txt

272 04C3 B9 0002 mov CX, fx\_length

273 04C6 B7 00 mov BH, 0

274 04C8 B6 01 mov DH, 1

275 04CA B2 4E mov DL, 78

276 04CC B8 1301 mov AX, 1301h

277

278 04CF 80 3E 012Ar FF cmp cursiveEnabled, true

279 04D4 74 07 je \_greenF2

280

281 04D6 \_redF2:

282 04D6 B3 4F mov BL, 01001111b ; red

283 04D8 CD 10 int 10h

284 04DA EB 05 90 jmp \_checkF3

285

286 04DD \_greenF2:

287 04DD B3 2F mov BL, 00101111b ; green

288 04DF CD 10 int 10h

289

290 04E1 \_checkF3:

291 04E1 BD 0489r lea BP, f3\_txt

292 04E4 B9 0002 mov CX, fx\_length

293 04E7 B7 00 mov BH, 0

294 04E9 B6 02 mov DH, 2

295 04EB B2 4E mov DL, 78

296 04ED B8 1301 mov AX, 1301h

297

298 04F0 80 3E 0128r FF cmp translateEnabled, true

299 04F5 74 07 je \_greenF3

300

301 04F7 \_redF3:

302 04F7 B3 4F mov BL, 01001111b ; red

303 04F9 CD 10 int 10h

304 04FB EB 05 90 jmp \_checkF4

305

306 04FE \_greenF3:

307 04FE B3 2F mov BL, 00101111b ; green

308 0500 CD 10 int 10h

309

310 0502 \_checkF4:

311 0502 BD 048Br lea BP, f4\_txt

312 0505 B9 0002 mov CX, fx\_length

313 0508 B7 00 mov BH, 0

314 050A B6 03 mov DH, 3

315 050C B2 4E mov DL, 78

316 050E B8 1301 mov AX, 1301h

317

318 0511 80 3E 011Dr FF cmp ignoreEnabled, true

319 0516 74 07 je \_greenF4

320

321 0518 \_redF4:

322 0518 B3 4F mov BL, 01001111b ; red

323 051A CD 10 int 10h

324 051C EB 05 90 jmp \_outFx

325

326 051F \_greenF4:

327 051F B3 2F mov BL, 00101111b ; green

328 0521 CD 10 int 10h

329

330 0523 \_outFx:

331 0523 5A pop DX

332 0524 B4 02 mov AH, 02h

333 0526 CD 10 int 10h

334

335 0528 07 pop ES

336 0529 5D pop BP

337 052A 5A pop DX

338 052B 59 pop CX

339 052C 5B pop BX

340 052D 58 pop AX

341 052E C3 ret

342 052F changeFx endp

343

344 ;новый обработчик

345 052F new\_int9h proc far

346 ; сохраняем значения всех, изменяемых регистров в стэке

347 052F 56 push SI

348 0530 50 push AX

349 0531 53 push BX

350 0532 51 push CX

351 0533 52 push DX

352 0534 06 push ES

353 0535 1E push DS

354 ; синхронизируем CS и DS

355 0536 0E push CS

356 0537 1F pop DS

357

358 0538 B8 0040 mov AX, 40h ; 40h-сегмент,где хранятся флаги сост-я клавиатуры, кольц. +

359 буфер ввода

360 053B 8E C0 mov ES, AX

361 053D E4 60 in AL, 60h ; записываем в AL скан-код нажатой клавиши

362

363 ;@ проверка на Ctrl+U, только для ИУ5-41

364 053F 3C 16 cmp AL, 22 ; была нажата клавиша U?

365 0541 75 24 jne \_test\_Fx

366 0543 26: 8A 26 0017 mov AH, ES:[17h] ; флаги клавиатуры

367 0548 80 E4 0F and AH, 00001111b

368 054B 80 FC 04 cmp AH, 00000100b ; был ли нажат ctrl?

369 054E 75 17 jne \_test\_Fx

370 ; выгрузка

371 0550 B4 FF mov AH, 0FFh

372 0552 B0 01 mov AL, 01h

373 0554 CD 2F int 2Fh

374 ; завершаем обработку нажатия

375

376 0556 E4 61 in AL, 61h ;контроллер состояния клавиатуры

377 0558 0C 80 or AL, 10000000b ;пометим, что клавишу нажали

378 055A E6 61 out 61h, AL

379 055C 24 7F and AL, 01111111b ;пометим, что клавишу отпустили

380 055E E6 61 out 61h, AL

381 0560 B0 20 mov AL, 20h

382 0562 E6 20 out 20h, AL ;отправим в контроллер прерываний признак конца +

383 прерывания

384

385 ; выходим

386 0564 E9 009D jmp \_quit

387

388 ;@ далее - код для всех вариантов

389

390 ;проверка F1-F4

391 0567 \_test\_Fx:

392 0567 2C 3A sub AL, 58 ; в AL теперь номер функциональной клавиши

393 0569 \_F1:

394 0569 3C 01 cmp AL, 1 ; F1

395 056B 75 0A jne \_F2

396 056D F6 16 0129r not signaturePrintingEnabled

397 0571 E8 FF19 call changeFx

398 0574 EB 2E 90 jmp \_translate\_or\_ignore

399 0577 \_F2:

400 0577 3C 02 cmp AL, 2 ; F2

401 0579 75 0D jne \_F3

402 057B F6 16 012Ar not cursiveEnabled

403 057F E8 FF0B call changeFx

404 0582 E8 01EF call setCursive ; перевод символа в курсив и обратно в зависимости от +

405 флага cursiveEnabled

406 0585 EB 1D 90 jmp \_translate\_or\_ignore

407 0588 \_F3:

408 0588 3C 03 cmp AL, 3 ; F3

409 058A 75 0A jne \_F4

410 058C F6 16 0128r not translateEnabled

411 0590 E8 FEFA call changeFx

412 0593 EB 0F 90 jmp \_translate\_or\_ignore

413 0596 \_F4:

414 0596 3C 04 cmp AL, 4 ; F4

415 0598 75 0A jne \_translate\_or\_ignore

416 059A F6 16 011Dr not ignoreEnabled

417 059E E8 FEEC call changeFx

418 05A1 EB 01 90 jmp \_translate\_or\_ignore

419

420 ;игнорирование и перевод

421 05A4 \_translate\_or\_ignore:

422

423 05A4 9C pushf

424 05A5 2E: FF 1E 014Cr call dword ptr CS:[old\_int9hOffset] ; вызываем стандартный обработчик прерывания

425 05AA B8 0040 mov AX, 40h ; 40h-сегмент,где хранятся флаги сост-я клавы,кольц. +

426 буфер ввода

427 05AD 8E C0 mov ES, AX

428 05AF 26: 8B 1E 001C mov BX, ES:[1Ch] ; адрес хвоста

429 05B4 4B dec BX ; сместимся назад к последнему

430 05B5 4B dec BX ; введённому символу

431 05B6 83 FB 1E cmp BX, 1Eh ; не вышли ли мы за пределы буфера?

432 05B9 73 03 jae \_go

433 05BB BB 003C mov BX, 3Ch ; хвост вышел за пределы буфера, значит последний введённый +

434 символ

435 ; находится в конце буфера

436

437 05BE \_go:

438 05BE 26: 8B 17 mov DX, ES:[BX] ; в DX 0 введённый символ

439 ;включен ли режим блокировки ввода?

440 05C1 80 3E 011Dr FF cmp ignoreEnabled, true

441 05C6 75 1A jne \_check\_translate

442

443 ; да, включен

444 05C8 BE 0000 mov SI, 0

445 05CB B9 001A mov CX, ignoredLength ;кол-во игнорируемых символов

446

447 ; проверяем, присутствует ли текущий символ в списке игнорируемых

448 05CE \_check\_ignored:

449 05CE 3A 94 0103r cmp DL,ignoredChars[SI]

450 05D2 74 06 je \_block

451 05D4 46 inc SI

452 05D5 E2 F7 loop \_check\_ignored

453 05D7 EB 09 90 jmp \_check\_translate

454

455 ; блокируем

456 05DA \_block:

457 05DA 26: 89 1E 001C mov ES:[1Ch], BX ;блокировка ввода символа

458 ;@ если по варианту нужно не блокировать ввод символа,

459 ;@ а заменять одни символы другими,

460 ;@ замените строку выше строкой

461 ;@ mov ES:[BX], AX

462 ;@ на месте AX может быть '\*' для замены всех символов множества ignoredChars +

463 на звёздочки

464 ;@ или, для перевода одних символов в другие - завести массив

465 ;@ replaceWith DB '...', где перечислить символы, на которые пойдёт замена

466 ;@ и раскомментировать строки ниже:

467 ;@ xor AX, AX

468 ;@ mov AL, replaceWith[SI]

469 ;@ mov ES:[BX], AX ; замена символа

470 05DF EB 23 90 jmp \_quit

471

472 05E2 \_check\_translate:

473 ; включен ли режим перевода?

474 05E2 80 3E 0128r FF cmp translateEnabled, true

475 05E7 75 1B jne \_quit

476

477 ; да, включен

478 05E9 BE 0000 mov SI, 0

479 05EC B9 0005 mov CX, translateLength ; кол-во символов для перевода

480 ; проверяем, присутствует ли текущий символ в списке для перевода

481 05EF \_check\_translate\_loop:

482 05EF 3A 94 011Er cmp DL, translateFrom[SI]

483 05F3 74 06 je \_translate

484 05F5 46 inc SI

485 05F6 E2 F7 loop \_check\_translate\_loop

486 05F8 EB 0A 90 jmp \_quit

487

488 ; переводим

489 05FB \_translate:

490 05FB 33 C0 xor AX, AX

491 05FD 8A 84 0123r mov AL, translateTo[SI]

492 0601 26: 89 07 mov ES:[BX], AX ; замена символа

493

494 0604 \_quit:

495 ; восстанавливаем все регистры

496 0604 1F pop DS

497 0605 07 pop ES

498 0606 5A pop DX

499 0607 59 pop CX

500 0608 5B pop BX

501 0609 58 pop AX

502 060A 5E pop SI

503 060B CF iret

504 060C new\_int9h endp

505

506 ;=== Обработчик прерывания int 1Ch ===;

507 ;=== Вызывается каждые 55 мс ===;

508 060C new\_int1Ch proc far

509 060C 50 push AX

510 060D 0E push CS

511 060E 1F pop DS

512

513 060F 9C pushf

514 0610 2E: FF 1E 0150r call dword ptr CS:[old\_int1ChOffset]

515

516 0615 80 3E 0129r FF cmp signaturePrintingEnabled, true ; если нажата управляющая клавиша (в данном случае +

517 F1)

518 061A 75 1C jne \_notToPrint

519

520 061C 83 3E 015Ar 25 cmp counter, printDelay\*1000/55 + 1 ; если кол-во "тактов" эквивалентно +

521 %printDelay% секундам

522 0621 74 03 je \_letsPrint

523

524 0623 EB 0E 90 jmp \_dontPrint

525

526 0626 \_letsPrint:

527 0626 F6 16 0129r not signaturePrintingEnabled

528 062A C7 06 015Ar 0000 mov counter, 0

529 0630 E8 0094 call printSignature

530

531 0633 \_dontPrint:

532 0633 83 06 015Ar 01 add counter, 1

533

534 0638 \_notToPrint:

535

536 0638 58 pop AX

537

538 0639 CF iret

539 063A new\_int1Ch endp

540

541 ;=== Обработчик прерывания int 2Fh ===;

542 ;=== Служит для:

543 ;=== 1) проверки факта присутствия TSR в памяти (при AH=0FFh, AL=0)

544 ;=== будет возвращён AH='i' в случае, если TSR уже загружен

545 ;=== 2) выгрузки TSR из памяти (при AH=0FFh, AL=1)

546 ;===

547 063A new\_int2Fh proc

548 063A 80 FC FF cmp AH, 0FFh ;наша функция?

549 063D 75 0B jne \_2Fh\_std ;нет - на старый обработчик

550 063F 3C 00 cmp AL, 0 ;подфункция проверки, загружен ли резидент в память?

551 0641 74 0C je \_already\_installed

552 0643 3C 01 cmp AL, 1 ;подфункция выгрузки из памяти?

553 0645 74 0B je \_uninstall

554 0647 EB 01 90 jmp \_2Fh\_std ;нет - на старый обработчик

555

556 064A \_2Fh\_std:

557 064A 2E: FF 2E 0154r jmp dword ptr CS:[old\_int2FhOffset] ;вызов старого обработчика

558

559 064F \_already\_installed:

560 064F B4 69 mov AH, 'i' ;вернём 'i', если резидент загружен в память

561 0651 CF iret

562

563 0652 \_uninstall:

564 0652 1E push DS

565 0653 06 push ES

566 0654 52 push DX

567 0655 53 push BX

568

569 0656 33 DB xor BX, BX

570

571 ; CS = ES, для доступа к переменным

572 0658 0E push CS

573 0659 07 pop ES

574

575 065A B8 2509 mov AX, 2509h

576 065D 26: 8B 16 014Cr mov DX, ES:old\_int9hOffset ; возвращаем вектор прерывания

577 0662 26: 8E 1E 014Er mov DS, ES:old\_int9hSegment ; на место

578 0667 CD 21 int 21h

579

580 0669 B8 251C mov AX, 251Ch

581 066C 26: 8B 16 0150r mov DX, ES:old\_int1ChOffset ; возвращаем вектор прерывания

582 0671 26: 8E 1E 0152r mov DS, ES:old\_int1ChSegment ; на место

583 0676 CD 21 int 21h

584

585 0678 B8 252F mov AX, 252Fh

586 067B 26: 8B 16 0154r mov DX, ES:old\_int2FhOffset ; возвращаем вектор прерывания

587 0680 26: 8E 1E 0156r mov DS, ES:old\_int2FhSegment ; на место

588 0685 CD 21 int 21h

589

590 0687 2E: 8E 06 002C mov ES, CS:2Ch ; загрузим в ES адрес окружения

591 068C B4 49 mov AH, 49h ; выгрузим из памяти окружение

592 068E CD 21 int 21h

593 0690 72 0B jc \_notRemove

594

595 0692 0E push CS

596 0693 07 pop ES ;в ES - адрес резидентной программы

597 0694 B4 49 mov AH, 49h ;выгрузим из памяти резидент

598 0696 CD 21 int 21h

599 0698 72 03 jc \_notRemove

600 069A EB 15 90 jmp \_unloaded

601

602 069D \_notRemove: ; не удалось выполнить выгрузку

603 ; вывод сообщения о неудачной выгрузке

604 069D B4 03 mov AH, 03h ; получаем позицию курсора

605 069F CD 10 int 10h

606 06A1 BD 0468r lea BP, noRemoveMsg

607 06A4 B9 001D mov CX, noRemoveMsg\_length

608 06A7 B3 07 mov BL, 0111b

609 06A9 B8 1301 mov AX, 1301h

610 06AC CD 10 int 10h

611 06AE EB 12 90 jmp \_2Fh\_exit

612

613 06B1 \_unloaded: ; выгрузка прошла успешно

614 ; вывод сообщения об удачной выгрузке

615 06B1 B4 03 mov AH, 03h ; получаем позицию курсора

616 06B3 CD 10 int 10h

617 06B5 BD 0457r lea BP, removedMsg

618 06B8 B9 0011 mov CX, removedMsg\_length

619 06BB B3 07 mov BL, 0111b

620 06BD B8 1301 mov AX, 1301h

621 06C0 CD 10 int 10h

622

623 06C2 \_2Fh\_exit:

624 06C2 5B pop BX

625 06C3 5A pop DX

626 06C4 07 pop ES

627 06C5 1F pop DS

628 06C6 CF iret

629 06C7 new\_int2Fh endp

630

631 ;=== Процедура вывода подписи (ФИО, группа)

632 ;=== Настраивается значениями переменных в начале исходника

633 ;===

634 06C7 printSignature proc

635 06C7 50 push AX

636 06C8 52 push DX

637 06C9 51 push CX

638 06CA 53 push BX

639 06CB 06 push ES

640 06CC 54 push SP

641 06CD 55 push BP

642 06CE 56 push SI

643 06CF 57 push DI

644

645 06D0 33 C0 xor AX, AX

646 06D2 33 DB xor BX, BX

647 06D4 33 D2 xor DX, DX

648

649 06D6 B4 03 mov AH, 03h ;чтение текущей позиции курсора

650 06D8 CD 10 int 10h

651 06DA 52 push DX ;помещаем информацию о +

652 положении курсора в стек

653

654 06DB 83 3E 015Cr 00 cmp printPos, 0

655 06E0 74 0E je \_printTop

656

657 06E2 83 3E 015Cr 01 cmp printPos, 1

658 06E7 74 0E je \_printCenter

659

660 06E9 83 3E 015Cr 02 cmp printPos, 2

661 06EE 74 0E je \_printBottom

662

663 ;все числа подобраны на глаз...

664 06F0 \_printTop:

665 06F0 B6 00 mov DH, 0

666 06F2 B2 0F mov DL, 15

667 06F4 EB 0F 90 jmp \_actualPrint

668

669 06F7 \_printCenter:

670 06F7 B6 09 mov DH, 9

671 06F9 B2 0F mov DL, 15

672 06FB EB 08 90 jmp \_actualPrint

673

674 06FE \_printBottom:

675 06FE B6 13 mov DH, 19

676 0700 B2 0F mov DL, 15

677 0702 EB 01 90 jmp \_actualPrint

678

679 0705 \_actualPrint:

680 0705 B4 0F mov AH, 0Fh ;чтение текущего видеорежима. в+

681 BH - текущая страница

682 0707 CD 10 int 10h

683

684 0709 0E push CS

685 070A 07 pop ES ;указываем ES на CS

686

687 ;вывод 'верхушки' таблицы

688 070B 52 push DX

689 070C BD 0394r lea BP, tableTop ;помещаем в BP указатель на +

690 выводимую строку

691 070F B9 0034 mov CX, tableTop\_length ;в CX - длина строки

692 0712 B3 07 mov BL, 0111b ;цвет выводимого текста ref: +

693 http://en.wikipedia.org/wiki/BIOS\_color\_attributes

694 0714 B8 1301 mov AX, 1301h ;AH=13h - номер ф-ии, AL=01h - +

695 курсор перемещается при выводе каждого из символов строки

696 0717 CD 10 int 10h

697 0719 5A pop DX

698 071A FE C6 inc DH

699

700

701 ;вывод первой линии

702 071C 52 push DX

703 071D BD 015Er lea BP, signatureLine1

704 0720 B9 0034 mov CX, Line1\_length

705 0723 B3 07 mov BL, 0111b

706 0725 B8 1301 mov AX, 1301h

707 0728 CD 10 int 10h

708 072A 5A pop DX

709 072B FE C6 inc DH

710

711 ;вывод второй линии

712 072D 52 push DX

713 072E BD 0192r lea BP, signatureLine2

714 0731 B9 0034 mov CX, Line2\_length

715 0734 B3 07 mov BL, 0111b

716 0736 B8 1301 mov AX, 1301h

717 0739 CD 10 int 10h

718 073B 5A pop DX

719 073C FE C6 inc DH

720

721 ;вывод третьей линии

722 073E 52 push DX

723 073F BD 01C6r lea BP, signatureLine3

724 0742 B9 0034 mov CX, Line3\_length

725 0745 B3 07 mov BL, 0111b

726 0747 B8 1301 mov AX, 1301h

727 074A CD 10 int 10h

728 074C 5A pop DX

729 074D FE C6 inc DH

730

731 ;вывод 'низа' таблицы

732 074F 52 push DX

733 0750 BD 03C8r lea BP, tableBottom

734 0753 B9 0034 mov CX, tableBottom\_length

735 0756 B3 07 mov BL, 0111b

736 0758 B8 1301 mov AX, 1301h

737 075B CD 10 int 10h

738 075D 5A pop DX

739 075E FE C6 inc DH

740

741 0760 33 DB xor BX, BX

742 0762 5A pop DX ;восстанавливаем из стека +

743 прежнее положение курсора

744 0763 B4 02 mov AH, 02h ;меняем положение курсора на +

745 первоначальное

746 0765 CD 10 int 10h

747 0767 E8 FD23 call changeFx

748

749 076A 5F pop DI

750 076B 5E pop SI

751 076C 5D pop BP

752 076D 5C pop SP

753 076E 07 pop ES

754 076F 5B pop BX

755 0770 59 pop CX

756 0771 5A pop DX

757 0772 58 pop AX

758

759 0773 C3 ret

760 0774 printSignature endp

761

762 ;=== Функция, которая в зависимости от флага cursiveEnabled меняет начертание символа с курсива+

763 на обычное и наоброт

764 ;=== Сама смена происходит в процедуре changeFont, а здесь подготавливаются данные

765 0774 setCursive proc

766 0774 06 push ES ; сохраняем регистры

767 0775 50 push AX

768 0776 0E push CS

769 0777 07 pop ES

770

771 0778 80 3E 012Ar FF cmp cursiveEnabled, true

772 077D 75 30 jne \_restoreSymbol

773 ; если флаг равен true, выполняем замену символа на курсивный вариант,

774 ; предварительно сохраняя старый символ в savedSymbol

775

776 077F E8 004C call saveFont

777 0782 8A 0E 013Br mov CL, charToCursiveIndex

778 0786 \_shifTtable:

779 ; мы получаем в BP таблицу всех символов. адрес указывает на символ 0

780 ; поэтому нуэно совершить сдвиг 16\*X - где X - код символа

781 0786 83 C5 10 add BP, 16

782 0789 E2 FB loop \_shiftTable

783

784 ; пpи savefont смещается pегистp ES

785 ; поэтомy пpиходится делать такие махинации, чтобы

786 ; записать полyченный элемент в savedSymbol

787 ; swap(ES, DS) и сохранение старого значения DS

788 078B 1E push DS

789 078C 58 pop AX

790 078D 06 push ES

791 078E 1F pop DS

792 078F 50 push AX

793 0790 07 pop ES

794 0791 50 push AX

795

796 0792 8B F5 mov SI, BP

797 0794 BF 013Cr lea DI, savedSymbol

798 ; сохpаняем в пеpеменнyю savedSymbol

799 ; таблицy нyжного символа

800 0797 B9 0010 mov CX, 16

801 ; movsb из DS:SI в ES:DI

802 079A F3> A4 rep movsb

803 ; исходные позиции сегментов возвpащены

804 079C 1F pop DS ; восстановление DS

805

806 ; заменим написание символа на кypсив

807 079D B9 0001 mov CX, 1

808 07A0 B6 00 mov DH, 0

809 07A2 8A 16 013Br mov DL, charToCursiveIndex

810 07A6 BD 012Br lea BP, cursiveSymbol

811 07A9 E8 0015 call changeFont

812 07AC EB 10 90 jmp \_exitSetCursive

813

814 07AF \_restoreSymbol:

815 ; если флаг равен 0, выполняем замену курсивного символа на старый вариант

816

817 07AF B9 0001 mov CX, 1

818 07B2 B6 00 mov DH, 0

819 07B4 8A 16 013Br mov DL, charToCursiveIndex

820 07B8 BD 013Cr lea bp, savedSymbol

821 07BB E8 0003 call changeFont

822

823 07BE \_exitSetCursive:

824 07BE 58 pop AX

825 07BF 07 pop ES

826 07C0 C3 ret

827 07C1 setCursive endp

828

829 ;=== Функция смены начертания символа (курсив/нормальное)

830 ;===

831 ; \*\*\* входные данные

832 ; DL = номер символа для замены

833 ; CX = Кол-во символов заменяемых изображений символов

834 ; (начиная с символа указанного в DX)

835 ; ES:bp = адрес таблицы

836 ;

837 ; \*\*\* описание работы процедуры

838 ; Происходит вызов int 10h (видеосервис)

839 ; с функцией AH = 11h (функции знакогенератора)

840 ; Параметр AL = 0 сообщает, что будет заменено изображение

841 ; символа для текущего шрифта

842 ; В случаях, когда AL = 1 или 2, будет заменено изображение

843 ; только для опредленного шрифта (8x14 и 8x8 соответственно)

844 ; Параметр BH = 0Eh сообщает, что на опредление каждого изображения символа

845 ; расходуется по 14 байт (режим 8x14 бит как раз 14 байт)

846 ; Параметр BL = 0 - блок шрифта для загрузки (от 0 до 4)

847 ;

848 ; \*\*\* результат

849 ; изображение указанного(ых) символа(ов) будет заменено

850 ; на предложенное пользователем.

851 ; Изменению подвергнутся все символы, находящиеся на экране,

852 ; то есть если изображение заменено, старый вариант нигде уже не проявится

853

854 07C1 changeFont proc

855 07C1 50 push AX

856 07C2 53 push BX

857 07C3 B8 1100 mov AX, 1100h

858 07C6 BB 1000 mov BX, 1000h

859 07C9 CD 10 int 10h

860 07CB 58 pop AX

861 07CC 5B pop BX

862 07CD C3 ret

863 07CE changeFont endp

864

865 ;=== Функция сохранения нормального начертания символа

866 ;===

867 ; \*\*\* входные данные

868 ; BH - тип возвращаемой символьной таблицы

869 ; 0 - таблица из int 1fh

870 ; 1 - таблица из int 44h

871 ; 2-5 - таблица из 8x14, 8x8, 8x8 (top), 9x14

872 ; 6 - 8x16

873 ;

874 ; \*\*\* описание работы процедуры

875 ; Происходит вызов int 10h (видеосервис)

876 ; с функцией AH = 11h (функции знакогенератора)

877 ; Параметр AL = 30 - подфункция получения информации о EGA

878 ;

879 ; \*\*\* результат

880 ; в ES:BP находится таблица символов (полная)

881 ; в CX находится байт на символ

882 ; в DL количество экранных строк

883 ; ВАЖНО! Происходит сдвиг регистра ES

884 ; ( ES становится равным C000h )

885

886 07CE saveFont proc

887 07CE 50 push AX

888 07CF 53 push BX

889 07D0 B8 1130 mov AX, 1130h

890 07D3 BB 0600 mov BX, 0600h

891 07D6 CD 10 int 10h

892 07D8 58 pop AX

893 07D9 5B pop BX

894 07DA C3 ret

895 07DB saveFont endp

896

897

898 ;=== Отсюда начинается выполнение основной части программы ===;

899 ;===

900 07DB \_initTSR: ; старт резидента

901 07DB B4 03 mov AH, 03h

902 07DD CD 10 int 10h

903 07DF 52 push DX

904 07E0 B4 00 mov AH,00h ; установка видеорежима (83h текст +

905 80x25 16/8 CGA,EGA b800 Comp,RGB,Enhanced), без очистки экрана

906 07E2 B0 83 mov AL,83h

907 07E4 CD 10 int 10h

908 07E6 5A pop DX

909 07E7 B4 02 mov AH, 02h

910 07E9 CD 10 int 10h

911

912

913 07EB E8 00B3 call commandParamsParser

914 07EE B8 3509 mov AX,3509h ; получить в ES:BX вектор 09

915 07F1 CD 21 int 21h ; прерывания

916

917 ;@ === Удаление резидента из памяти ===

918 ;@ Если по варианту необходимо выгружать резидент по повторному запуску приложений,

919 ;@ нужно закомментировать следующие 3 строки, а также

920 ;@ содержимое метки \_finishTSR ф-ии commandParamsParser, но не саму метку!

921 07F3 80 3E 0158r FF cmp unloadTSR, true

922 07F8 74 03 je \_removingOnParameter

923 07FA EB 15 90 jmp \_notRemovingNow

924

925 07FD \_removingOnParameter:

926 07FD B4 FF mov AH, 0FFh

927 07FF B0 00 mov AL, 0

928 0801 CD 2F int 2Fh

929 0803 80 FC 69 cmp AH, 'i' ; проверка того, загружена ли уже программа

930 0806 74 7D je \_remove

931 0808 B4 09 mov AH, 09h ;@ для выгрузки резидента по повторному+

932 запуску закомментировать эту строку

933 080A BA 0439r lea DX, notInstalledMsg ;@ для выгрузки резидента по повторному запуску +

934 закомментировать эту строку

935 080D CD 21 int 21h ;@ для выгрузки резидента по повторному+

936 запуску закомментировать эту строку

937 080F CD 20 int 20h ;@ для выгрузки резидента по повторному+

938 запуску закомментировать эту строку

939

940 0811 \_notRemovingNow:

941

942 0811 80 3E 0159r FF cmp notLoadTSR, true ; если была выведена справка

943 0816 74 0E je \_exit\_tmp ; просто выходим

944

945 ;@ Если по варианту необходимо выгружать резидент по повторному запуску, то +

946 комментируем 5 строк ниже

947 ;@ если необходимо выгружать по параметру коммандной строки, то оставляем их

948 0818 B4 FF mov AH, 0FFh

949 081A B0 00 mov AL, 0

950 081C CD 2F int 2Fh

951 081E 80 FC 69 cmp AH, 'i' ; проверка того, загружена ли уже программа

952 0821 74 6B je \_alreadyInstalled

953

954 0823 EB 04 90 jmp \_tmp

955

956 0826 \_exit\_tmp:

957 0826 EB 77 90 jmp \_exit

958

959 0829 \_tmp:

960 0829 06 push ES

961 082A A1 002C mov AX, DS:[2Ch] ; psp

962 082D 8E C0 mov ES, AX

963 082F B4 49 mov AH, 49h ; хватит памяти чтоб остаться

964 0831 CD 21 int 21h ; резидентом?

965 0833 07 pop ES

966 0834 72 62 jc \_notMem ; не хватило - выходим

967

968 ;== int 09h ==;

969

970 0836 2E: 89 1E 014Cr mov word ptr CS:old\_int9hOffset, BX

971 083B 2E: 8C 06 014Er mov word ptr CS:old\_int9hSegment, ES

972 0840 B8 2509 mov AX, 2509h ; установим вектор на 09

973 0843 BA 052Fr mov DX, offset new\_int9h ; прерывание

974 0846 CD 21 int 21h

975

976 ;== int 1Ch ==;

977 0848 B8 351C mov AX,351Ch ; получить в ES:BX вектор 1C

978 084B CD 21 int 21h ; прерывания

979 084D 2E: 89 1E 0150r mov word ptr CS:old\_int1ChOffset, BX

980 0852 2E: 8C 06 0152r mov word ptr CS:old\_int1ChSegment, ES

981 0857 B8 251C mov AX, 251Ch ; установим вектор на 1C

982 085A BA 060Cr mov DX, offset new\_int1Ch ; прерывание

983 085D CD 21 int 21h

984

985 ;== int 2Fh ==;

986 085F B8 352F mov AX,352Fh ; получить в ES:BX вектор 1C

987 0862 CD 21 int 21h ; прерывания

988 0864 2E: 89 1E 0154r mov word ptr CS:old\_int2FhOffset, BX

989 0869 2E: 8C 06 0156r mov word ptr CS:old\_int2FhSegment, ES

990 086E B8 252F mov AX, 252Fh ; установим вектор на 2F

991 0871 BA 063Ar mov DX, offset new\_int2Fh ; прерывание

992 0874 CD 21 int 21h

993

994 0876 E8 FC14 call changeFx

995 0879 BA 03FCr mov DX, offset installedMsg ; выводим что все ок

996 087C B4 09 mov AH, 9

997 087E CD 21 int 21h

998 0880 BA 07DBr mov DX, offset \_initTSR ; остаемся в памяти резидентом

999 0883 CD 27 int 27h ; и выходим

1000 ; конец основной программы

1001 0885 \_remove: ; выгрузка программы из памяти

1002 0885 B4 FF mov AH, 0FFh

1003 0887 B0 01 mov AL, 1

1004 0889 CD 2F int 2Fh

1005 088B EB 12 90 jmp \_exit

1006 088E \_alreadyInstalled:

1007 088E B4 09 mov AH, 09h

1008 0890 BA 040Fr lea DX, alreadyInstalledMsg

1009 0893 CD 21 int 21h

1010 0895 EB 08 90 jmp \_exit

1011 0898 \_notMem: ; не хватает памяти, чтобы остаться резидентом

1012 0898 BA 0425r mov DX, offset noMemMsg

1013 089B B4 09 mov AH, 9

1014 089D CD 21 int 21h

1015 089F \_exit: ; выход

1016 089F CD 20 int 20h

1017

1018 ;=== Процедура проверки параметров ком. строки ===;

1019 ;===

1020 08A1 commandParamsParser proc

1021 08A1 0E push CS

1022 08A2 07 pop ES

1023 08A3 C6 06 0158r 00 mov unloadTSR, 0

1024 08A8 C6 06 0159r 00 mov notLoadTSR, 0

1025

1026 08AD BE 0080 mov SI, 80h ;SI=смещение командной строки.

1027 08B0 AC lodsb ;Получим кол-во символов.

1028 08B1 0A C0 or AL, AL ;Если 0 символов введено,

1029 08B3 74 45 jz \_exitHelp ;то все в порядке.

1030

1031 08B5 \_nextChar:

1032

1033 08B5 46 inc SI ;Теперь SI указывает на первый символ +

1034 строки.

1035

1036 08B6 80 3C 0D cmp [SI], BYTE ptr 13

1037 08B9 74 3F je \_exitHelp

1038

1039

1040 08BB AD lodsw ;Получаем два символа

1041 08BC 3D 3F2F cmp AX, '?/' ;Это '/?' (данные расположены в +

1042 обратном порядк, т.е. AL:AH вместо AH:AL)

1043 08BF 74 08 je \_question

1044 08C1 3D 752F cmp AX, 'u/'

1045 08C4 74 1A je \_finishTSR

1046

1047 ;cmp AH, '/'

1048 ;je \_errorParam

1049

1050 08C6 EB 32 90 jmp \_exitHelp

1051

1052 08C9 \_question:

1053 ; вывод строки помощи

1054 08C9 B4 03 mov AH,03

1055 08CB CD 10 int 10h

1056 08CD BD 01FAr lea BP, helpMsg

1057 08D0 B9 0175 mov CX, helpMsg\_length

1058 08D3 B3 07 mov BL, 0111b

1059 08D5 B8 1301 mov AX, 1301h

1060 08D8 CD 10 int 10h

1061 ; конец вывода строки помощи

1062 08DA F6 16 0159r not notLoadTSR ;флаг того, что необходимо не загружать резидент

1063 08DE EB D5 jmp \_nextChar

1064

1065 ;@ === Удаление резидента из памяти ===

1066 ;@ Если по варианту необходимо выгружать резидент по параметру '/u' коммандной строки,

1067 ;@ нужно использовать следующий код, в остальных случаях необходимо закомменитровать

1068 ;@ этот код, кроме названия метки! (по желанию можно избавиться и от метки, но +

1069 аккуратно просмотреть использование)

1070 08E0 \_finishTSR:

1071 08E0 F6 16 0158r not unloadTSR ;флаг того, что необходимо выгузить резидент

1072 08E4 EB CF jmp \_nextChar

1073

1074 08E6 EB 12 90 jmp \_exitHelp

1075

1076 08E9 \_errorParam:

1077 ;вывод строки

1078 08E9 B4 03 mov AH,03

1079 08EB CD 10 int 10h

1080 08ED BD 036Fr lea BP, CS:errorParamMsg

1081 08F0 B9 0025 mov CX, errorParamMsg\_length

1082 08F3 B3 07 mov BL, 0111b

1083 08F5 B8 1301 mov AX, 1301h

1084 08F8 CD 10 int 10h

1085 ;конец вывода строки

1086 08FA \_exitHelp:

1087 08FA C3 ret

1088 08FB commandParamsParser endp

1089

1090 08FB code ends

1091 end \_start

Symbol Table

Symbol Name Type Value

??DATE Text "04/30/13"

??FILENAME Text "tsr "

??TIME Text "00:45:16"

??VERSION Number 030A

@CPU Text 0101H

@CURSEG Text CODE

@FILENAME Text TSR

@WORDSIZE Text 2

ALREADYINSTALLEDMSG Byte CODE:040F

CHANGEFONT Near CODE:07C1

CHANGEFX Near CODE:048D

CHARTOCURSIVEINDEX Byte CODE:013B

COMMANDPARAMSPARSER Near CODE:08A1

COUNTER Word CODE:015A

CURSIVEENABLED Byte CODE:012A

CURSIVESYMBOL Byte CODE:012B

ERRORPARAMMSG Byte CODE:036F

ERRORPARAMMSG\_LENGTH Number 0025

F1\_TXT Byte CODE:0485

F2\_TXT Byte CODE:0487

F3\_TXT Byte CODE:0489

F4\_TXT Byte CODE:048B

FX\_LENGTH Number 0002

HELPMSG Byte CODE:01FA

HELPMSG\_LENGTH Number 0175

IGNOREDCHARS Byte CODE:0103

IGNOREDLENGTH Number 001A

IGNOREENABLED Byte CODE:011D

INSTALLEDMSG Byte CODE:03FC

LINE1\_LENGTH Number 0034

LINE2\_LENGTH Number 0034

LINE3\_LENGTH Number 0034

NEW\_INT1CH Far CODE:060C

NEW\_INT2FH Near CODE:063A

NEW\_INT9H Far CODE:052F

NOMEMMSG Byte CODE:0425

NOREMOVEMSG Byte CODE:0468

NOREMOVEMSG\_LENGTH Number 001D

NOTINSTALLEDMSG Byte CODE:0439

NOTLOADTSR Byte CODE:0159

OLD\_INT1CHOFFSET Word CODE:0150

OLD\_INT1CHSEGMENT Word CODE:0152

OLD\_INT2FHOFFSET Word CODE:0154

OLD\_INT2FHSEGMENT Word CODE:0156

OLD\_INT9HOFFSET Word CODE:014C

OLD\_INT9HSEGMENT Word CODE:014E

PRINTDELAY Number 0002

PRINTPOS Word CODE:015C

PRINTSIGNATURE Near CODE:06C7

REMOVEDMSG Byte CODE:0457

REMOVEDMSG\_LENGTH Number 0011

SAVEDSYMBOL Byte CODE:013C

SAVEFONT Near CODE:07CE

SETCURSIVE Near CODE:0774

SIGNATURELINE1 Byte CODE:015E

SIGNATURELINE2 Byte CODE:0192

SIGNATURELINE3 Byte CODE:01C6

SIGNATUREPRINTINGENABLED Byte CODE:0129

TABLEBOTTOM Byte CODE:03C8

TABLEBOTTOM\_LENGTH Number 0034

TABLETOP Byte CODE:0394

TABLETOP\_LENGTH Number 0034

TRANSLATEENABLED Byte CODE:0128

TRANSLATEFROM Byte CODE:011E

TRANSLATELENGTH Number 0005

TRANSLATETO Byte CODE:0123

TRUE Number 00FF

UNLOADTSR Byte CODE:0158

\_2FH\_EXIT Near CODE:06C2

\_2FH\_STD Near CODE:064A

\_ACTUALPRINT Near CODE:0705

\_ALREADYINSTALLED Near CODE:088E

\_ALREADY\_INSTALLED Near CODE:064F

\_BLOCK Near CODE:05DA

\_CHECKF1 Near CODE:049C

\_CHECKF2 Near CODE:04C0

\_CHECKF3 Near CODE:04E1

\_CHECKF4 Near CODE:0502

\_CHECK\_IGNORED Near CODE:05CE

\_CHECK\_TRANSLATE Near CODE:05E2

\_CHECK\_TRANSLATE\_LOOP Near CODE:05EF

\_DONTPRINT Near CODE:0633

\_ERRORPARAM Near CODE:08E9

\_EXIT Near CODE:089F

\_EXITHELP Near CODE:08FA

\_EXITSETCURSIVE Near CODE:07BE

\_EXIT\_TMP Near CODE:0826

\_F1 Near CODE:0569

\_F2 Near CODE:0577

\_F3 Near CODE:0588

\_F4 Near CODE:0596

\_FINISHTSR Near CODE:08E0

\_GO Near CODE:05BE

\_GREENF1 Near CODE:04B9

\_GREENF2 Near CODE:04DD

\_GREENF3 Near CODE:04FE

\_GREENF4 Near CODE:051F

\_INITTSR Near CODE:07DB

\_LETSPRINT Near CODE:0626

\_NEXTCHAR Near CODE:08B5

\_NOTMEM Near CODE:0898

\_NOTREMOVE Near CODE:069D

\_NOTREMOVINGNOW Near CODE:0811

\_NOTTOPRINT Near CODE:0638

\_OUTFX Near CODE:0523

\_PRINTBOTTOM Near CODE:06FE

\_PRINTCENTER Near CODE:06F7

\_PRINTTOP Near CODE:06F0

\_QUESTION Near CODE:08C9

\_QUIT Near CODE:0604

\_REDF1 Near CODE:04B2

\_REDF2 Near CODE:04D6

\_REDF3 Near CODE:04F7

\_REDF4 Near CODE:0518

\_REMOVE Near CODE:0885

\_REMOVINGONPARAMETER Near CODE:07FD

\_RESTORESYMBOL Near CODE:07AF

\_SHIFTTABLE Near CODE:0786

\_START Near CODE:0100

\_TEST\_FX Near CODE:0567

\_TMP Near CODE:0829

\_TRANSLATE Near CODE:05FB

\_TRANSLATE\_OR\_IGNORE Near CODE:05A4

\_UNINSTALL Near CODE:0652

\_UNLOADED Near CODE:06B1

Groups & Segments Bit Size Align Combine Class

CODE 16 08FB Para none CODE

1. **Файл unloader.lst**

Turbo Assembler Version 3.1

unloader.asm

1 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

2 ; unloader.asm

3 ;

4 ; Сборка:

5 ; tasm.exe /l unloader.asm

6 ; tlink /t /x unloader.obj

7 ;

8 ; Программа для выгрузки TSR из памяти

9 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

10

11 0000 code segment 'code'

12 assume CS:code, DS:code

13 org 100h

14 0100 \_start:

15

16 0100 B4 FF mov AH, 0FFh

17 0102 B0 01 mov AL, 1

18 0104 CD 2F int 2Fh ; наше прерывание

19 0106 CD 20 int 20h ; выходим

20

21 0108 code ends

22 end \_start

Symbol Table

Symbol Name Type Value

??DATE Text "04/28/13"

??FILENAME Text "unloader"

??TIME Text "19:37:59"

??VERSION Number 030A

@CPU Text 0101H

@CURSEG Text CODE

@FILENAME Text UNLOADER

@WORDSIZE Text 2

\_START Near CODE:0100

Groups & Segments Bit Size Align Combine Class

CODE 16 0108 Para none CODE