# Содержание

[Содержание 1](#_Toc784796)

[tsrbuff.asm 1](#_Toc784797)

[tsrinout.asm 31](#_Toc784798)

[unloader.asm 61](#_Toc784799)

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 1

# tsrbuff.asm

1 ; =======================================

2 ; tsrbuff.asm

3 ;

4 ; Сборка:

5 ; > tasm.exe /l tsrbuff.asm

6 ; > tlink /t /x tsrbuff.obj

7 ; =======================================

8

9 0000 code segment 'code'

10 assume CS:code, DS:code

11 org 100h

12

13 0100 \_start:

14 0100 E9 0582 jmp \_initTSR ; на начало программы

15

16 0103 61 62 63 64 65 ignoredChars DB 'abcde' ; игнорируемые символы

17 0108 05 ignoredLength DB 5 ; длина строки ignoredChars

18 0109 00 ignoreEnabled DB 0 ; флаг функции игнорирования ввода

19 010A 51 57 45 52 54 59 translateFrom DB 'QWERTY' ; заменяемые символы

20 0110 89 96 93 8A 85 8D translateTo DB 'ЙЦУКЕН' ; символы, на которые будет происходить замена

21 0116 06 translateLength DB 6 ; длина строки translateFrom

22 0117 00 translateEnabled DB 0 ; флаг функции перевода

23

24 0118 00 signaturePrintingEnabled DB 0 ; флаг вывода подписи

25 0119 0000 counter DW 0

26 =0002 printDelay EQU 2 ; задержка перед выводом "подписи" в секундах

27

28 011B 0034 signatureLineLength DW 52 ; длина одной строчки подписи

29 011D B3 88 A2 A0 AD AE A2+ signatureLine1 DB 179, 'Иванов Иван Иванович ', +

30 20 88 A2 A0 AD 20 88+ 179

31 A2 A0 AD AE A2 A8 E7+

32 20 20 20 20 20 20 20+

33 20 20 20 20 20 20 20+

34 20 20 20 20 20 20 20+

35 20 20 20 20 20 20 20+

36 20 20 B3

37 0151 B3 88 93 35 2D 34 58+ signatureLine2 DB 179, 'ИУ5-4X ', +

38 20 20 20 20 20 20 20+ 179

39 20 20 20 20 20 20 20+

40 20 20 20 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 0185 B3 82 A0 E0 A8 A0 AD+ signatureLine3 DB 179, 'Вариант #0 ', +

46 E2 20 23 30 20 20 20+ 179

47 20 20 20 20 20 20 20+

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 B3

53 01B9 DA 32\*(C4) BF tableTop DB '┌', 50 dup ('─'), '┐'

54 01ED C0 32\*(C4) D9 tableBottom DB '└', 50 dup ('─'), '┘'

55

56 0221 3E 20 6B 72 2E 63 6F+ helpMsg DB '> kr.com [/?] [/u]', 10, 13

57 6D 20 5B 2F 3F 5D 20+

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 2

tsrbuff.asm

58 5B 2F 75 5D 0A 0D

59 0235 20 5B 2F 3F 5D 20 2D+ DB ' [/?] - вывод данной справки', 10, 13

60 20 A2 EB A2 AE A4 20+

61 A4 A0 AD AD AE A9 20+

62 E1 AF E0 A0 A2 AA A8+

63 0A 0D

64 0253 20 5B 2F 75 5D 20 2D+ DB ' [/u] - выгрузка резидента из памяти', 10, 13

65 20 A2 EB A3 E0 E3 A7+

66 AA A0 20 E0 A5 A7 A8+

67 A4 A5 AD E2 A0 20 A8+

68 A7 20 AF A0 AC EF E2+

69 A8 0A 0D

70 0279 20 20 46 31 20 20 2D+ DB ' F1 - вывод ФИО и группы по таймеру в центре экрана', 10, 13

71 20 A2 EB A2 AE A4 20+

72 94 88 8E 20 A8 20 A3+

73 E0 E3 AF AF EB 20 AF+

74 AE 20 E2 A0 A9 AC A5+

75 E0 E3 20 A2 20 E6 A5+

76 AD E2 E0 A5 20 ED AA+

77 E0 A0 AD A0 0A 0D

78 02B0 20 20 46 32 20 20 2D+ DB ' F2 - включение/отключения курсивного вывода русского символа В', 10, 13

79 20 A2 AA AB EE E7 A5+

80 AD A8 A5 2F AE E2 AA+

81 AB EE E7 A5 AD A8 EF+

82 20 AA E3 E0 E1 A8 A2+

83 AD AE A3 AE 20 A2 EB+

84 A2 AE A4 A0 20 E0 E3+

85 E1 E1 AA AE A3 AE 20+

86 E1 A8 AC A2 AE AB A0+

87 20 82 0A 0D

88 02F3 20 20 46 33 20 20 2D+ DB ' F3 - включение/отключение частичной русификации клавиатуры: QWERTY -> +

89 20 A2 AA AB EE E7 A5+ ЙЦУКЕН', 10, 13

90 AD A8 A5 2F AE E2 AA+

91 AB EE E7 A5 AD A8 A5+

92 20 E7 A0 E1 E2 A8 E7+

93 AD AE A9 20 E0 E3 E1+

94 A8 E4 A8 AA A0 E6 A8+

95 A8 20 AA AB A0 A2 A8+

96 A0 E2 E3 E0 EB 3A 20+

97 51 57 45 52 54 59 20+

98 2D 3E 20 89 96 93 8A+

99 85 8D 0A 0D

100 0344 20 20 46 34 20 20 2D+ DB ' F4 - включение/отключение режима блокировки ввода букв abcde', 10, 13, 0

101 20 A2 AA AB EE E7 A5+

102 AD A8 A5 2F AE E2 AA+

103 AB EE E7 A5 AD A8 A5+

104 20 E0 A5 A6 A8 AC A0+

105 20 A1 AB AE AA A8 E0+

106 AE A2 AA A8 20 A2 A2+

107 AE A4 A0 20 A1 E3 AA+

108 A2 20 61 62 63 64 65+

109 0A 0D 00

110

111 =0165 helpMsgLength EQU $-helpMsg

112 0386 00 commandLineResult DB 0

113

114 0387 00 cursiveEnabled DB 0 ; флаг перевода символа в курсив

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 3

tsrbuff.asm

115 0388 00 cursiveSymbol DB 00000000b ; символ, составленный из единиц (его курсивный+

116 вариант)

117 0389 00 DB 00000000b

118 038A 00 DB 00000000b

119 038B 3E DB 00111110b

120 038C 3F DB 00111111b

121 038D 33 DB 00110011b

122 038E 66 DB 01100110b

123 038F 66 DB 01100110b

124 0390 7C DB 01111100b

125 0391 C6 DB 11000110b

126 0392 C6 DB 11000110b

127 0393 C6 DB 11000110b

128 0394 FC DB 11111100b

129 0395 00 DB 00000000b

130 0396 00 DB 00000000b

131 0397 00 DB 00000000b

132

133 0398 82 charToCursiveIndex DB 'В' ; символ для замены

134 0399 10\*(FF) savedSymbol DB 16 dup(0FFh) ; переменная для хранения старого символа

135

136 03A9 ???? old\_int9hOffset DW ? ; адрес старого обработчика int 9h

137 03AB ???? old\_int9hSegment DW ? ; сегмент старого обработчика int 9h

138 03AD ???? old\_int1ChOffset DW ? ; адрес старого обработчика int 1Ch

139 03AF ???? old\_int1ChSegment DW ? ; сегмент старого обработчика int 1Ch

140 03B1 ???? old\_int2FhOffset DW ? ; адрес старого обработчика int 2Fh

141 03B3 ???? old\_int2FhSegment DW ? ; сегмент старого обработчика int 2Fh

142

143 03B5 90 A5 A7 A8 A4 A5 AD+ installedMsg DB 'Резидент загружен.', 0

144 E2 20 A7 A0 A3 E0 E3+

145 A6 A5 AD 2E 00

146 03C8 90 A5 A7 A8 A4 A5 AD+ alreadyInstalledMsg DB 'Резидент уже был загружен.', 0

147 E2 20 E3 A6 A5 20 A1+

148 EB AB 20 A7 A0 A3 E0+

149 E3 A6 A5 AD 2E 00

150 03E3 90 A5 A7 A8 A4 A5 AD+ notInstalledMsg DB 'Резидент не был загружен.$'

151 E2 20 AD A5 20 A1 EB+

152 AB 20 A7 A0 A3 E0 E3+

153 A6 A5 AD 2E 24

154

155 03FD 90 A5 A7 A8 A4 A5 AD+ removedMsg DB 'Резидент выгружен из памяти.'

156 E2 20 A2 EB A3 E0 E3+

157 A6 A5 AD 20 A8 A7 20+

158 AF A0 AC EF E2 A8 2E

159 =001C removedMsg\_length EQU $-removedMsg

160

161 0419 8D A5 20 E3 A4 A0 AB+ noRemoveMsg DB 'Не удалось выгрузить резидент'

162 AE E1 EC 20 A2 EB A3+

163 E0 E3 A7 A8 E2 EC 20+

164 E0 A5 A7 A8 A4 A5 AD+

165 E2

166 =001D noRemoveMsg\_length EQU $-noRemoveMsg

167

168 =00FF true EQU 0FFh ; нужно для удобства использования not с флагами

169 ; 0FFh = 11111111b = инверсия 00000000b

170

171 ; новый обработчик прерывания int 9h

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 4

tsrbuff.asm

172 ; (работа с клавиатурой)

173 0436 new\_int9h proc far

174 0436 56 50 53 51 52 06 1E push SI AX BX CX DX ES DS ; сохраняем значения всех, изменяемых регистров+

175 в стеке

176 043D 0E push CS ; синхронизируем CS и DS

177 043E 1F pop DS

178

179 043F 9C pushf

180 0440 2E: FF 1E 03A9r call dword ptr CS:[old\_int9hOffset] ; вызываем стандартный обработчик прерывания

181 0445 B8 0040 mov AX, 40h ; 40h - сегмент, где хранятся флаги состояния +

182 клавиатуры

183 0448 8E C0 mov ES, AX

184 044A 26: 8B 1E 001C mov BX, ES:[1Ch] ; адрес хвоста

185 044F 83 EB 02 sub BX, 2h ; сместимся назад к последнему введённому +

186 символу

187 0452 83 FB 1E cmp BX, 1Eh ; не вышли ли мы за пределы буфера?

188 0455 73 03 jae \_go

189 0457 BB 003C mov BX, 3Ch ; хвост вышел за пределы буфера: значит, +

190 последний

191 ; введённый символ находится в конце буфера

192 045A \_go:

193 045A 26: 8B 17 mov DX, ES:[BX] ; в DX 0 введённый символ

194

195 045D \_test\_Fx: ; проверка F1-F4

196 045D \_F1:

197 045D 80 FE 3B cmp DH, 3Bh ; F1

198 0460 75 0C jne \_F2

199 0462 F6 16 0118r not signaturePrintingEnabled

200 0466 26: 89 1E 001C mov ES:[1Ch], BX ; блокировка ввода символа

201 046B E9 008E jmp \_quit

202 046E \_F2:

203 046E 80 FE 3C cmp DH, 3Ch ; F2

204 0471 75 0F jne \_F3

205 0473 26: 89 1E 001C mov ES:[1Ch], BX ; блокировка ввода символа

206 0478 F6 16 0387r not cursiveEnabled

207 047C E8 0085 call toggleCursive ; перевод символа в курсив и обратно

208 ; в зависимости от флага cursiveEnabled

209 047F EB 7B 90 jmp \_quit

210 0482 \_F3:

211 0482 80 FE 3D cmp DH, 3Dh ; F3

212 0485 75 0C jne \_F4

213 0487 F6 16 0117r not translateEnabled

214 048B 26: 89 1E 001C mov ES:[1Ch], BX ; блокировка ввода символа

215 0490 EB 6A 90 jmp \_quit

216 0493 \_F4:

217 0493 80 FE 3E cmp DH, 3Eh ; F4

218 0496 75 0C jne \_translateOrIgnore

219 0498 F6 16 0109r not ignoreEnabled

220 049C 26: 89 1E 001C mov ES:[1Ch], BX ; блокировка ввода символа

221 04A1 EB 59 90 jmp \_quit

222

223 04A4 \_translateOrIgnore: ; просто выводим набранный символ на экран

224 ; @@@ следующий блок отвечает за выгрузку по Ctrl-U: @@@

225 04A4 80 FA 15 cmp DL, 15h ; проверяем, что введённый символ - это [Ctrl+U]

226 04A7 75 0E jne \_notCtrlU

227 04A9 26: 89 1E 001C mov ES:[1Ch], BX ; блокируем символ [Ctrl+U]

228 04AE B4 FF mov AH, 0FFh ; выгрузка

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 5

tsrbuff.asm

229 04B0 B0 01 mov AL, 01h

230 04B2 CD 2F int 2Fh

231 04B4 EB 46 90 jmp \_quit

232

233 04B7 \_notCtrlU:

234 ; @@@ конец блока @@@

235 04B7 80 3E 0109r FF cmp ignoreEnabled, true ; включен ли режим блокировки ввода?

236 04BC 75 1B jne \_checkTranslate

237

238 04BE BE 0000 mov SI, 0 ; да, включен

239 04C1 8A 0E 0108r mov CL, ignoredLength ; количество игнорируемых символов

240

241 04C5 \_checkIgnored:

242 04C5 3A 94 0103r cmp DL, ignoredChars[SI] ; проверяем, присутствует ли текущий символ в +

243 списке игнорируемых

244 04C9 74 06 je \_block

245 04CB 46 inc SI

246 04CC E2 F7 loop \_checkIgnored ; зацикливаем ignoredLength раз

247 04CE EB 09 90 jmp \_checkTranslate

248

249 ; блокируем

250 04D1 \_block:

251 04D1 26: 89 1E 001C mov ES:[1Ch], BX ; блокировка ввода символа

252 ; если по варианту нужно не блокировать ввод символа,

253 ; а заменять одни символы другими, замените строку выше строкой

254 ; mov ES:[BX], AX

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

256 звёздочки

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

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

259 ; и раскомментировать строки ниже:

260 ; xor AX, AX

261 ; mov AL, replaceWith[SI]

262 ; mov ES:[BX], AX ; замена символа

263 04D6 EB 24 90 jmp \_quit

264

265 04D9 \_checkTranslate:

266 04D9 80 3E 0117r FF cmp translateEnabled, true ; включен ли режим перевода?

267 04DE 75 1C jne \_quit

268

269 04E0 BE 0000 mov SI, 0 ; да, включен

270 04E3 8A 0E 0116r mov CL, translateLength ; кол-во символов для перевода

271

272 04E7 \_checkTranslateLoop:

273 04E7 3A 94 010Ar cmp DL, translateFrom[SI] ; присутствует ли текущий символ в списке для +

274 перевода?

275 04EB 74 06 je \_translate

276 04ED 46 inc SI

277 04EE E2 F7 loop \_checkTranslateLoop ; продолжаем, пока не закончим проверять каждый+

278 символ

279 04F0 EB 0A 90 jmp \_quit

280

281 04F3 \_translate:

282 04F3 33 C0 xor AX, AX ; переводим

283 04F5 8A 84 0110r mov AL, translateTo[SI]

284 04F9 26: 89 07 mov ES:[BX], AX ; замена символа

285

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 6

tsrbuff.asm

286 04FC \_quit:

287 04FC 1F 07 5A 59 5B 58 5E pop DS ES DX CX BX AX SI ; восстанавливаем все регистры

288 0503 CF iret

289 0504 new\_int9h endp

290

291 ; в зависимости от флага cursiveEnabled меняет начертание символа на курсив и обратно

292 ; сама смена происходит в процедуре changeFont - здесь же подготавливаются данные

293 0504 toggleCursive proc

294 0504 06 50 push ES AX ; сохраняем регистры

295 0506 0E push CS

296 0507 07 pop ES

297

298 0508 80 3E 0387r FF cmp cursiveEnabled, true ; если флаг равен true,

299 050D 75 30 jne \_restoreSymbol ; выполняем замену символа на курсивный вариант,

300 ; предварительно сохраняя старый символ в +

301 savedSymbol

302

303 050F E8 004E call saveFont

304 0512 8A 0E 0398r mov CL, charToCursiveIndex

305 0516 \_shiftTable:

306 0516 83 C5 10 add BP, 16 ; получаем в BP таблицу всех символов. адрес +

307 указывает на символ 0

308 ; поэтому нужно совершить сдвиг 16\*X - где X - +

309 код символа

310 0519 E2 FB loop \_shiftTable

311

312 051B 1E push DS ; пpи savefont смещается pегистp ES

313 051C 58 pop AX ; поэтомy пpиходится делать такие махинации, +

314 чтобы

315 051D 06 push ES ; записать полyченный элемент в savedSymbol

316 051E 1F pop DS

317 051F 50 push AX ; DS -> AX, ES -> DS, AX -> ES => ES и DS +

318 поменялись местами

319 0520 07 pop ES ; + сохранение старого значения DS в AX

320 0521 50 push AX

321

322 0522 8B F5 mov SI, BP

323 0524 BF 0399r lea DI, savedSymbol ; сохpаняем в пеpеменнyю savedSymbol таблицу +

324 нужного символа

325

326 0527 B9 0010 mov CX, 16 ; movsb из DS:SI в ES:DI

327

328 052A F3> A4 rep movsb ; исходные позиции сегментов возвpащены

329 052C 1F pop DS ; восстановление DS

330

331 052D B9 0001 mov CX, 1 ; заменим написание символа на кypсив

332 0530 B6 00 mov DH, 0

333 0532 8A 16 0398r mov DL, charToCursiveIndex

334 0536 BD 0388r lea BP, cursiveSymbol

335 0539 E8 0015 call changeFont

336 053C EB 10 90 jmp \_exitToggleCursive

337

338 053F \_restoreSymbol:

339 053F B9 0001 mov CX, 1 ; если флаг равен 0, заменяем курсивный символ +

340 на старый вариант

341 0542 B6 00 mov DH, 0

342 0544 8A 16 0398r mov DL, charToCursiveIndex

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 7

tsrbuff.asm

343 0548 BD 0399r lea BP, savedSymbol

344 054B E8 0003 call changeFont

345

346 054E \_exitToggleCursive:

347 054E 58 pop AX

348 054F 07 pop ES

349 0550 C3 ret

350 0551 toggleCursive endp

351

352 ; функция смены начертания символа (курсив/нормальное)

353 ;

354 ; входные данные:

355 ; 1) DL = номер символа для замены

356 ; 2) CX = количество символов заменяемых изображений символов

357 ; (начиная с символа указанного в DX)

358 ; 3) ES:BP = адрес таблицы

359 ;

360 ; описание работы процедуры:

361 ; 1) происходит вызов int 10h (видеосервис)

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

363 ; параметр AL = 0 сообщает, что будет заменено изображение

364 ; символа для текущего шрифта.

365 ; в случаях, когда AL = 1 или 2, будет заменено изображение

366 ; только для определенного шрифта (8x14 и 8x8 соответственно)

367 ; 2) параметр BH = 0Eh сообщает, что на определение каждого изображения символа

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

369 ; 3) параметр BL = 0 - блок шрифта для загрузки (от 0 до 4)

370 ;

371 ; результат:

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

373 ; на предложенное пользователем.

374 ; изменению подвергнутся все символы, находящиеся на экране:

375 ; таким образом, если изображение заменено, старый вариант нигде уже не проявится

376 0551 changeFont proc

377 0551 50 53 52 push AX BX DX

378 0554 B8 1100 mov AX, 1100h

379 0557 BB 1000 mov BX, 1000h

380 055A CD 10 int 10h

381 055C 5A 5B 58 pop DX BX AX

382 055F C3 ret

383 0560 changeFont endp

384

385 ; функция сохранения нормального начертания символа

386 ;

387 ; входные данные:

388 ; BH - тип возвращаемой символьной таблицы

389 ; = 0 - таблица из int 1fh

390 ; = 1 - таблица из int 44h

391 ; = 2..5 - таблица из 8x14, 8x8, 8x8 (top), 9x14

392 ; = 6 - 8x16

393 ;

394 ; описание работы процедуры:

395 ; происходит вызов int 10h (видеосервис)

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

397 ; параметр AL = 30 - подфункция получения информации о EGA

398 ;

399 ; результат:

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 8

tsrbuff.asm

400 ; 1) в ES:BP находится таблица символов (полная)

401 ; 2) в CX находится байт на символ

402 ; 3) в DL количество экранных строк

403 ; важно! происходит сдвиг регистра ES (ES = C000h)

404 0560 saveFont proc

405 0560 50 53 52 push AX BX DX

406 0563 B8 1130 mov AX, 1130h

407 0566 BB 0600 mov BX, 0600h

408 0569 CD 10 int 10h

409 056B 5B 58 5A pop BX AX DX

410 056E C3 ret

411 056F saveFont endp

412

413 ; обработчик прерывания int 2Fh

414 ; служит для:

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

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

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

418 056F new\_int2Fh proc

419 056F 80 FC FF cmp AH, 0FFh ; наша процедура?

420 0572 75 0B jne \_2Fh\_default ; нет - на стандартный обработчик

421 0574 3C 00 cmp AL, 0 ; подпроцедура проверки, загружен ли резидент в+

422 память?

423 0576 74 0C je \_alreadyInstalled2Fh

424 0578 3C 01 cmp AL, 1 ; подпроцедура выгрузки из памяти?

425 057A 74 0B je \_uninstall

426 057C EB 01 90 jmp \_2Fh\_default

427

428 057F \_2Fh\_default:

429 057F 2E: FF 2E 03B1r jmp dword ptr CS:[old\_int2FhOffset] ; вызов стандартного обработчика

430

431 0584 \_alreadyInstalled2Fh:

432 0584 B4 69 mov AH, 'i' ; пусть AH = 'i', если резидент уже загружен в +

433 память

434 0586 CF iret ; конечно, вместо 'i' может быть любое значение

435

436 0587 \_uninstall: ; подпроцедура выгрузки из памяти

437 0587 1E 06 52 53 push DS ES DX BX

438 058B 33 DB xor BX, BX

439

440 058D 0E push CS ; CS = ES, для доступа к переменным

441 058E 07 pop ES

442

443 058F B8 2509 mov AX, 2509h

444 0592 26: 8B 16 03A9r mov DX, ES:old\_int9hOffset ; возвращаем вектор прерывания 09h на место

445 0597 26: 8E 1E 03ABr mov DS, ES:old\_int9hSegment

446 059C CD 21 int 21h

447

448 059E B8 251C mov AX, 251Ch

449 05A1 26: 8B 16 03ADr mov DX, ES:old\_int1ChOffset ; возвращаем вектор прерывания 1Ch на место

450 05A6 26: 8E 1E 03AFr mov DS, ES:old\_int1ChSegment

451 05AB CD 21 int 21h

452

453 05AD B8 252F mov AX, 252Fh

454 05B0 26: 8B 16 03B1r mov DX, ES:old\_int2FhOffset ; возвращаем вектор прерывания 2Fh на место

455 05B5 26: 8E 1E 03B3r mov DS, ES:old\_int2FhSegment

456 05BA CD 21 int 21h

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 9

tsrbuff.asm

457

458 05BC 2E: 8E 06 002C mov ES, CS:2Ch ; загрузим в ES адрес окружения

459 05C1 B4 49 mov AH, 49h ; выгрузим из памяти окружение

460 05C3 CD 21 int 21h

461 05C5 72 0B jc \_notRemove

462

463 05C7 0E push CS

464 05C8 07 pop ES ; в ES - адрес резидентной программы

465 05C9 B4 49 mov AH, 49h ; выгрузим из памяти резидент

466 05CB CD 21 int 21h

467

468 05CD 72 03 jc \_notRemove

469 05CF EB 15 90 jmp \_unloaded

470

471 05D2 \_notRemove: ; не удалось выполнить выгрузку => вывод ошибки

472 05D2 B4 03 mov AH, 03h ; получаем позицию курсора

473 05D4 CD 10 int 10h

474 05D6 BD 0419r lea BP, noRemoveMsg

475 05D9 B9 001D mov CX, noRemoveMsg\_length

476 05DC B3 07 mov BL, 0111b

477 05DE B8 1301 mov AX, 1301h

478 05E1 CD 10 int 10h

479 05E3 EB 12 90 jmp \_2Fh\_exit

480

481 05E6 \_unloaded: ; выгрузка прошла успешно => вывод сообщения

482 05E6 B4 03 mov AH, 03h ; получаем позицию курсора

483 05E8 CD 10 int 10h

484 05EA BD 03FDr lea BP, removedMsg

485 05ED B9 001C mov CX, removedMsg\_length

486 05F0 B3 07 mov BL, 0111b

487 05F2 B8 1301 mov AX, 1301h

488 05F5 CD 10 int 10h

489

490 05F7 \_2Fh\_exit:

491 05F7 5B 5A 07 1F pop BX DX ES DS

492 05FB CF iret

493 05FC new\_int2Fh endp

494

495 ; обработчик прерывания int 1Ch

496 ; вызывается каждые 55 мс

497 05FC new\_int1Ch proc far

498 05FC 50 push AX

499 05FD 0E push CS

500 05FE 1F pop DS

501

502 05FF 9C pushf

503 0600 2E: FF 1E 03ADr call dword ptr CS:[old\_int1ChOffset] ; вызываем стандартный обработчик прерывания

504

505 0605 80 3E 0118r FF cmp signaturePrintingEnabled, true ; если нажата управляющая клавиша (в данном +

506 случае F1)

507 060A 75 1B jne \_notToPrint

508

509 060C 83 3E 0119r 25 cmp counter, printDelay\*1000/55 + 1 ; если кол-во "тактов" равно printDelay секундам

510 0611 74 03 je \_letsPrint

511

512 0613 EB 0E 90 jmp \_dontPrint

513

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 10

tsrbuff.asm

514 0616 \_letsPrint:

515 0616 F6 16 0118r not signaturePrintingEnabled

516 061A C7 06 0119r 0000 mov counter, 0

517 0620 E8 0016 call printSignature ; выводим подпись на экран

518

519 0623 \_dontPrint:

520 0623 FF 06 0119r inc counter ; увеличим значение счетчика на 1

521

522 0627 \_notToPrint:

523 0627 58 pop AX

524 0628 CF iret

525 0629 new\_int1Ch endp

526

527 ; выводит одну строку подписи

528 0629 printSignatureLine proc

529 0629 52 push DX

530 062A 8B 0E 011Br mov CX, signatureLineLength

531 062E B3 07 mov BL, 0111b ; цвет выводимого текста

532 0630 B8 1301 mov AX, 1301h ; AH = 13h - номер ф-ии, AL = 01h - перемещение+

533 курсора

534 0633 CD 10 int 10h

535 0635 5A pop DX

536 0636 FE C6 inc DH

537 0638 C3 ret

538 0639 printSignatureLine endp

539

540 ; процедура вывода подписи

541 0639 printSignature proc

542 0639 50 52 51 53 06 54 55+ push AX DX CX BX ES SP BP SI DI

543 56 57

544

545 0642 33 C0 xor AX, AX ; обнуляем значения регистров

546 0644 33 DB xor BX, BX

547 0646 33 D2 xor DX, DX

548

549 0648 B4 03 mov AH, 03h ; чтение текущей позиции курсора

550 064A CD 10 int 10h

551 064C 52 push DX ; помещаем информацию о положении курсора в стек

552

553 064D BA 090F mov DX, 090Fh ; NB! вверху: 000Fh, посередине: 090Fh, внизу: +

554 130Fh

555

556 0650 \_actualPrint:

557 0650 B4 0F mov AH, 0Fh ; чтение текущего видеорежима. в BH - текущая +

558 страница

559 0652 CD 10 int 10h

560

561 0654 0E push CS

562 0655 07 pop ES ; указываем ES на CS

563

564 0656 BD 01B9r lea BP, tableTop

565 0659 E8 FFCD call printSignatureLine ; выводим верх таблицы

566 065C BD 011Dr lea BP, signatureLine1

567 065F E8 FFC7 call printSignatureLine ; выводим первую строку

568 0662 BD 0151r lea BP, signatureLine2

569 0665 E8 FFC1 call printSignatureLine ; выводим вторую строку

570 0668 BD 0185r lea BP, signatureLine3

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 11

tsrbuff.asm

571 066B E8 FFBB call printSignatureLine ; выводим третью строку

572 066E BD 01EDr lea BP, tableBottom

573 0671 E8 FFB5 call printSignatureLine ; выводим низ таблицы

574

575 0674 33 DB xor BX, BX

576 0676 5A pop DX ; восстанавливаем из стека прежнее положение +

577 курсора

578 0677 B4 02 mov AH, 02h ; меняем положение курсора на первоначальное

579 0679 CD 10 int 10h

580

581 067B 5F 5E 5D 5C 07 5B 59+ pop DI SI BP SP ES BX CX DX AX

582 5A 58

583 0684 C3 ret

584 0685 printSignature endp

585

586 ; Основная часть программы

587 ; 1) установка видеорежима

588 ; 2) проверка, запущен ли резидент

589 ; 3) установка вектора прерываний

590 0685 \_initTSR:

591 0685 B4 03 mov AH, 03h

592 0687 CD 10 int 10h

593 0689 52 push DX

594 068A B4 00 mov AH, 00h ; установка видеорежима

595 068C B0 83 mov AL, 83h

596 068E CD 10 int 10h

597 0690 5A pop DX

598 0691 B4 02 mov AH, 02h

599 0693 CD 10 int 10h

600

601 0695 E8 009F call commandParamsParser ; читаем аргументы командной строки

602 0698 80 3E 0386r 02 cmp commandLineResult, 2 ; если результат = 2, значит была выведена +

603 справка

604 069D 75 03 jne \_shouldContinue ; соответственно, никаких других действий +

605 делать не нужно

606 069F E9 0093 jmp \_exit

607 06A2 \_shouldContinue:

608 ; ### следующий блок отвечает за выгрузку при аргументе командной строки /u и при простом +

609 перезапуске ###

610 06A2 80 3E 0386r 01 cmp commandLineResult, 1 ; проверяем результат работы процедуры

611 06A7 75 14 jne \_go\_on

612 06A9 B4 FF mov AH, 0FFh

613 06AB B0 00 mov AL, 0

614 06AD CD 2F int 2Fh ; проверка того, загружена ли уже программа

615 06AF 80 FC 69 cmp AH, 'i' ; если запущена, то AH = 'i' (см. процедуру +

616 new\_int2Fh)

617 06B2 74 64 je \_remove

618

619 06B4 B4 09 mov AH, 09h

620 06B6 BA 03E3r lea DX, notInstalledMsg ; не была загружена

621 06B9 CD 21 int 21h

622 06BB CD 20 int 20h

623 06BD \_go\_on:

624 ; ### конец блока ###

625 ; @@@ отвечает за выгрузку при перезапуске @@@

626 06BD B4 FF mov AH, 0FFh ; ещё раз проверяем, запущен ли резидент сейчас

627 06BF B0 00 mov AL, 0

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 12

tsrbuff.asm

628 06C1 CD 2F int 2Fh

629 06C3 80 FC 69 cmp AH, 'i' ; если запущена, то AH = 'i' (см. процедуру +

630 new\_int2Fh)

631 06C6 74 64 je \_alreadyInstalled

632 ; @@@ конец блока @@@

633

634 06C8 B8 3509 mov AX, 3509h ; получить в ES:BX прерывания 09h

635 06CB CD 21 int 21h

636 06CD 2E: 89 1E 03A9r mov word ptr CS:old\_int9hOffset, BX ; обработчик прерывания 09h

637 06D2 2E: 8C 06 03ABr mov word ptr CS:old\_int9hSegment, ES

638 06D7 B8 2509 mov AX, 2509h ; установим вектор на прерывание 09h

639 06DA BA 0436r mov DX, offset new\_int9h

640 06DD CD 21 int 21h

641

642 06DF B8 351C mov AX, 351Ch ; получить в ES:BX прерывания 1Ch

643 06E2 CD 21 int 21h

644 06E4 2E: 89 1E 03ADr mov word ptr CS:old\_int1ChOffset, BX ; обработчик прерывания 1Ch

645 06E9 2E: 8C 06 03AFr mov word ptr CS:old\_int1ChSegment, ES

646 06EE B8 251C mov AX, 251Ch ; установим вектор на прерывание 1Ch

647 06F1 BA 05FCr mov DX, offset new\_int1Ch

648 06F4 CD 21 int 21h

649

650 06F6 B8 352F mov AX, 352Fh ; получить в ES:BX прерывания 2Fh

651 06F9 CD 21 int 21h

652 06FB 2E: 89 1E 03B1r mov word ptr CS:old\_int2FhOffset, BX ; обработчик прерывания 2Fh

653 0700 2E: 8C 06 03B3r mov word ptr CS:old\_int2FhSegment, ES

654 0705 B8 252F mov AX, 252Fh ; установим вектор на прерывание 2Fh

655 0708 BA 056Fr mov DX, offset new\_int2Fh

656 070B CD 21 int 21h

657

658 070D BB 03B5r lea BX, installedMsg ; выводим сообщение, что всё ОК

659 0710 E8 0074 call printStr

660

661 0713 BA 0685r mov DX, offset \_initTSR ; остаемся в памяти и выходим из основной части

662 0716 CD 27 int 27h

663

664 0718 \_remove: ; выгрузка из памяти

665 0718 06 push ES

666 0719 A1 002C mov AX, DS:[2Ch] ; PSP

667 071C 8E C0 mov ES, AX

668 071E B4 49 mov AH, 49h ; хватит памяти чтоб остаться резидентом?

669 0720 CD 21 int 21h

670 0722 07 pop ES

671

672 0723 B4 FF mov AH, 0FFh

673 0725 B0 01 mov AL, 1

674 0727 CD 2F int 2Fh

675 0729 EB 0A 90 jmp \_exit

676 072C \_alreadyInstalled: ; резидент уже запущен

677 072C BB 03C8r lea BX, alreadyInstalledMsg

678 072F E8 0055 call printStr

679 0732 EB 01 90 jmp \_exit

680 0735 \_exit: ; выход

681 0735 CD 20 int 20h

682

683 ; парсер аргментов командной строки. выводит справку.

684 ; устанавливает флаг commandLineResult:

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 13

tsrbuff.asm

685 ; 0 = всё ОК; 1 = нужна выгрузка; 2 = была выведена справка, не нужно загружать резидент

686 0737 commandParamsParser proc

687 0737 0E push CS

688 0738 07 pop ES

689

690 0739 BE 0080 mov SI, 80h ; SI = смещение командной строки

691 073C AC lodsb ; получим кол-во символов

692 073D 0A C0 or AL, AL ; если 0 символов введено,

693 073F 74 30 jz \_paramParsingEndWithUnload ; ### то дополнительная проверка, был ли уже +

694 загружен

695 ; ### резидент. в таком случае он выгружается

696

697 0741 \_nextChar:

698 0741 46 inc SI ; теперь SI указывает на первый символ строки

699

700 0742 80 3C 00 cmp [SI], BYTE ptr 0

701 0745 74 3A je \_paramParsingEnd

702

703 0747 AD lodsw ; получаем два символа

704 0748 3D 3F2F cmp AX, '?/'

705 074B 74 16 je \_displayHelp

706 ; @@@ следует раскомментировать, если нужно выгружать по аргументу /u @@@

707 074D 3D 752F cmp AX, 'u/'

708 0750 74 07 je \_finishTSR

709 0752 3D 552F cmp AX, 'U/'

710 0755 74 02 je \_finishTSR

711 0757 EB E8 jmp \_nextChar

712

713 0759 \_finishTSR:

714 0759 C6 06 0386r 01 mov commandLineResult, 1 ; флаг того, что необходимо выгрузить резидент

715 075E EB E1 jmp \_nextChar

716 ; @@@ конец блока @@@

717

718 0760 EB 1F 90 jmp \_paramParsingEnd

719 0763 \_displayHelp:

720 0763 BB 0221r lea BX, helpMsg ; выводим справку

721 0766 E8 001E call printStr

722 0769 C6 06 0386r 02 mov commandLineResult, 2 ; флаг того, что резидент загружать не надо

723

724 ; ### далее - проверка: если резидент уже загружен, то выгрузить ###

725 076E EB 11 90 jmp \_paramParsingEnd

726 0771 \_paramParsingEndWithUnload:

727 0771 B4 FF mov AH, 0FFh

728 0773 B0 00 mov AL, 0

729 0775 CD 2F int 2Fh

730 0777 80 FC 69 cmp AH, 'i' ; проверка того, загружена ли уже программа

731 077A 75 05 jne \_paramParsingEnd

732

733 077C C6 06 0386r 01 mov commandLineResult, 1

734 ; ### конец блока ###

735

736 0781 \_paramParsingEnd:

737 0781 C3 ret

738 0782 commandParamsParser endp

739

740 ; отображает символ из AL

741 0782 printChar proc

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 14

tsrbuff.asm

742 0782 B4 0E mov AH, 0EH

743 0784 CD 10 int 010H

744 0786 C3 ret

745 0787 printChar endp

746

747 ; отображает нуль-терминированную строку из [BX]

748 0787 printStr proc

749 0787 52 50 push DX AX

750 0789 8B 07 mov AX, [BX]

751 078B \_printStrLoop:

752 078B 3C 00 cmp AL, 0

753 078D 74 08 je \_printStrEnd

754 078F E8 FFF0 call printChar

755 0792 43 inc BX

756 0793 8B 07 mov AX, [BX]

757 0795 EB F4 jmp \_printStrLoop

758 0797 \_printStrEnd:

759 0797 58 5A pop AX DX

760 0799 C3 ret

761 079A printStr endp

762

763 079A code ends

764 end \_start

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 15

Symbol Table

Symbol Name Type Value

??DATE Text "04/25/18"

??FILENAME Text "tsrbuff "

??TIME Text "12:50:28"

??VERSION Number 030A

@CPU Text 0101H

@CURSEG Text CODE

@FILENAME Text TSRBUFF

@WORDSIZE Text 2

ALREADYINSTALLEDMSG Byte CODE:03C8

CHANGEFONT Near CODE:0551

CHARTOCURSIVEINDEX Byte CODE:0398

COMMANDLINERESULT Byte CODE:0386

COMMANDPARAMSPARSER Near CODE:0737

COUNTER Word CODE:0119

CURSIVEENABLED Byte CODE:0387

CURSIVESYMBOL Byte CODE:0388

HELPMSG Byte CODE:0221

HELPMSGLENGTH Number 0165

IGNOREDCHARS Byte CODE:0103

IGNOREDLENGTH Byte CODE:0108

IGNOREENABLED Byte CODE:0109

INSTALLEDMSG Byte CODE:03B5

NEW\_INT1CH Far CODE:05FC

NEW\_INT2FH Near CODE:056F

NEW\_INT9H Far CODE:0436

NOREMOVEMSG Byte CODE:0419

NOREMOVEMSG\_LENGTH Number 001D

NOTINSTALLEDMSG Byte CODE:03E3

OLD\_INT1CHOFFSET Word CODE:03AD

OLD\_INT1CHSEGMENT Word CODE:03AF

OLD\_INT2FHOFFSET Word CODE:03B1

OLD\_INT2FHSEGMENT Word CODE:03B3

OLD\_INT9HOFFSET Word CODE:03A9

OLD\_INT9HSEGMENT Word CODE:03AB

PRINTCHAR Near CODE:0782

PRINTDELAY Number 0002

PRINTSIGNATURE Near CODE:0639

PRINTSIGNATURELINE Near CODE:0629

PRINTSTR Near CODE:0787

REMOVEDMSG Byte CODE:03FD

REMOVEDMSG\_LENGTH Number 001C

SAVEDSYMBOL Byte CODE:0399

SAVEFONT Near CODE:0560

SIGNATURELINE1 Byte CODE:011D

SIGNATURELINE2 Byte CODE:0151

SIGNATURELINE3 Byte CODE:0185

SIGNATURELINELENGTH Word CODE:011B

SIGNATUREPRINTINGENABLED Byte CODE:0118

TABLEBOTTOM Byte CODE:01ED

TABLETOP Byte CODE:01B9

TOGGLECURSIVE Near CODE:0504

TRANSLATEENABLED Byte CODE:0117

TRANSLATEFROM Byte CODE:010A

TRANSLATELENGTH Byte CODE:0116

Turbo Assembler Version 3.1 04/25/18 12:50:28 Page 16

Symbol Table

TRANSLATETO Byte CODE:0110

TRUE Number 00FF

\_2FH\_DEFAULT Near CODE:057F

\_2FH\_EXIT Near CODE:05F7

\_ACTUALPRINT Near CODE:0650

\_ALREADYINSTALLED Near CODE:072C

\_ALREADYINSTALLED2FH Near CODE:0584

\_BLOCK Near CODE:04D1

\_CHECKIGNORED Near CODE:04C5

\_CHECKTRANSLATE Near CODE:04D9

\_CHECKTRANSLATELOOP Near CODE:04E7

\_DISPLAYHELP Near CODE:0763

\_DONTPRINT Near CODE:0623

\_EXIT Near CODE:0735

\_EXITTOGGLECURSIVE Near CODE:054E

\_F1 Near CODE:045D

\_F2 Near CODE:046E

\_F3 Near CODE:0482

\_F4 Near CODE:0493

\_FINISHTSR Near CODE:0759

\_GO Near CODE:045A

\_GO\_ON Near CODE:06BD

\_INITTSR Near CODE:0685

\_LETSPRINT Near CODE:0616

\_NEXTCHAR Near CODE:0741

\_NOTCTRLU Near CODE:04B7

\_NOTREMOVE Near CODE:05D2

\_NOTTOPRINT Near CODE:0627

\_PARAMPARSINGEND Near CODE:0781

\_PARAMPARSINGENDWITHUNLOAD Near CODE:0771

\_PRINTSTREND Near CODE:0797

\_PRINTSTRLOOP Near CODE:078B

\_QUIT Near CODE:04FC

\_REMOVE Near CODE:0718

\_RESTORESYMBOL Near CODE:053F

\_SHIFTTABLE Near CODE:0516

\_SHOULDCONTINUE Near CODE:06A2

\_START Near CODE:0100

\_TEST\_FX Near CODE:045D

\_TRANSLATE Near CODE:04F3

\_TRANSLATEORIGNORE Near CODE:04A4

\_UNINSTALL Near CODE:0587

\_UNLOADED Near CODE:05E6

Groups & Segments Bit Size Align Combine Class

CODE 16 079A Para none CODE

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 1

# tsrinout.asm

1 ; =======================================

2 ; tsrinout.asm

3 ;

4 ; Сборка:

5 ; > tasm.exe /l tsrinout.asm

6 ; > tlink /t /x tsrinout.obj

7 ; =======================================

8

9 0000 code segment 'code'

10 assume CS:code, DS:code

11 org 100h

12

13 0100 \_start:

14 0100 E9 0588 jmp \_initTSR ; на начало программы

15

16 0103 61 62 63 64 65 ignoredChars DB 'abcde' ; игнорируемые символы

17 0108 05 ignoredLength DB 5 ; длина строки ignoredChars

18 0109 00 ignoreEnabled DB 0 ; флаг функции игнорирования ввода

19 010A 51 57 45 52 54 59 translateFrom DB 'QWERTY' ; заменяемые символы

20 0110 89 96 93 8A 85 8D translateTo DB 'ЙЦУКЕН' ; символы, на которые будет происходить замена

21 0116 06 translateLength DB 6 ; длина строки translateFrom

22 0117 00 translateEnabled DB 0 ; флаг функции перевода

23

24 0118 00 signaturePrintingEnabled DB 0 ; флаг вывода подписи

25 0119 0000 counter DW 0

26 =0002 printDelay EQU 2 ; задержка перед выводом "подписи" в секундах

27

28 011B 0034 signatureLineLength DW 52 ; длина одной строчки подписи

29 011D B3 88 A2 A0 AD AE A2+ signatureLine1 DB 179, 'Иванов Иван Иванович ', +

30 20 88 A2 A0 AD 20 88+ 179

31 A2 A0 AD AE A2 A8 E7+

32 20 20 20 20 20 20 20+

33 20 20 20 20 20 20 20+

34 20 20 20 20 20 20 20+

35 20 20 20 20 20 20 20+

36 20 20 B3

37 0151 B3 88 93 35 2D 34 58+ signatureLine2 DB 179, 'ИУ5-4X ', +

38 20 20 20 20 20 20 20+ 179

39 20 20 20 20 20 20 20+

40 20 20 20 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 0185 B3 82 A0 E0 A8 A0 AD+ signatureLine3 DB 179, 'Вариант #0 ', +

46 E2 20 23 30 20 20 20+ 179

47 20 20 20 20 20 20 20+

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 B3

53 01B9 DA 32\*(C4) BF tableTop DB '┌', 50 dup ('─'), '┐'

54 01ED C0 32\*(C4) D9 tableBottom DB '└', 50 dup ('─'), '┘'

55

56 0221 3E 20 6B 72 2E 63 6F+ helpMsg DB '> kr.com [/?] [/u]', 10, 13

57 6D 20 5B 2F 3F 5D 20+

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 2

tsrinout.asm

58 5B 2F 75 5D 0A 0D

59 0235 20 5B 2F 3F 5D 20 2D+ DB ' [/?] - вывод данной справки', 10, 13

60 20 A2 EB A2 AE A4 20+

61 A4 A0 AD AD AE A9 20+

62 E1 AF E0 A0 A2 AA A8+

63 0A 0D

64 0253 20 5B 2F 75 5D 20 2D+ DB ' [/u] - выгрузка резидента из памяти', 10, 13

65 20 A2 EB A3 E0 E3 A7+

66 AA A0 20 E0 A5 A7 A8+

67 A4 A5 AD E2 A0 20 A8+

68 A7 20 AF A0 AC EF E2+

69 A8 0A 0D

70 0279 20 20 46 31 20 20 2D+ DB ' F1 - вывод ФИО и группы по таймеру в центре экрана', 10, 13

71 20 A2 EB A2 AE A4 20+

72 94 88 8E 20 A8 20 A3+

73 E0 E3 AF AF EB 20 AF+

74 AE 20 E2 A0 A9 AC A5+

75 E0 E3 20 A2 20 E6 A5+

76 AD E2 E0 A5 20 ED AA+

77 E0 A0 AD A0 0A 0D

78 02B0 20 20 46 32 20 20 2D+ DB ' F2 - включение/отключения курсивного вывода русского символа В', 10, 13

79 20 A2 AA AB EE E7 A5+

80 AD A8 A5 2F AE E2 AA+

81 AB EE E7 A5 AD A8 EF+

82 20 AA E3 E0 E1 A8 A2+

83 AD AE A3 AE 20 A2 EB+

84 A2 AE A4 A0 20 E0 E3+

85 E1 E1 AA AE A3 AE 20+

86 E1 A8 AC A2 AE AB A0+

87 20 82 0A 0D

88 02F3 20 20 46 33 20 20 2D+ DB ' F3 - включение/отключение частичной русификации клавиатуры: QWERTY -> +

89 20 A2 AA AB EE E7 A5+ ЙЦУКЕН', 10, 13

90 AD A8 A5 2F AE E2 AA+

91 AB EE E7 A5 AD A8 A5+

92 20 E7 A0 E1 E2 A8 E7+

93 AD AE A9 20 E0 E3 E1+

94 A8 E4 A8 AA A0 E6 A8+

95 A8 20 AA AB A0 A2 A8+

96 A0 E2 E3 E0 EB 3A 20+

97 51 57 45 52 54 59 20+

98 2D 3E 20 89 96 93 8A+

99 85 8D 0A 0D

100 0344 20 20 46 34 20 20 2D+ DB ' F4 - включение/отключение режима блокировки ввода букв abcde', 10, 13, 0

101 20 A2 AA AB EE E7 A5+

102 AD A8 A5 2F AE E2 AA+

103 AB EE E7 A5 AD A8 A5+

104 20 E0 A5 A6 A8 AC A0+

105 20 A1 AB AE AA A8 E0+

106 AE A2 AA A8 20 A2 A2+

107 AE A4 A0 20 A1 E3 AA+

108 A2 20 61 62 63 64 65+

109 0A 0D 00

110

111 =0165 helpMsgLength EQU $-helpMsg

112 0386 00 commandLineResult DB 0

113

114 0387 00 cursiveEnabled DB 0 ; флаг перевода символа в курсив

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 3

tsrinout.asm

115 0388 00 cursiveSymbol DB 00000000b ; символ, составленный из единиц (его курсивный+

116 вариант)

117 0389 00 DB 00000000b

118 038A 00 DB 00000000b

119 038B 3E DB 00111110b

120 038C 3F DB 00111111b

121 038D 33 DB 00110011b

122 038E 66 DB 01100110b

123 038F 66 DB 01100110b

124 0390 7C DB 01111100b

125 0391 C6 DB 11000110b

126 0392 C6 DB 11000110b

127 0393 C6 DB 11000110b

128 0394 FC DB 11111100b

129 0395 00 DB 00000000b

130 0396 00 DB 00000000b

131 0397 00 DB 00000000b

132

133 0398 82 charToCursiveIndex DB 'В' ; символ для замены

134 0399 10\*(FF) savedSymbol DB 16 dup(0FFh) ; переменная для хранения старого символа

135

136 03A9 ???? old\_int9hOffset DW ? ; адрес старого обработчика int 9h

137 03AB ???? old\_int9hSegment DW ? ; сегмент старого обработчика int 9h

138 03AD ???? old\_int1ChOffset DW ? ; адрес старого обработчика int 1Ch

139 03AF ???? old\_int1ChSegment DW ? ; сегмент старого обработчика int 1Ch

140 03B1 ???? old\_int2FhOffset DW ? ; адрес старого обработчика int 2Fh

141 03B3 ???? old\_int2FhSegment DW ? ; сегмент старого обработчика int 2Fh

142

143 03B5 90 A5 A7 A8 A4 A5 AD+ installedMsg DB 'Резидент загружен.', 0

144 E2 20 A7 A0 A3 E0 E3+

145 A6 A5 AD 2E 00

146 03C8 90 A5 A7 A8 A4 A5 AD+ alreadyInstalledMsg DB 'Резидент уже был загружен.', 0

147 E2 20 E3 A6 A5 20 A1+

148 EB AB 20 A7 A0 A3 E0+

149 E3 A6 A5 AD 2E 00

150 03E3 90 A5 A7 A8 A4 A5 AD+ notInstalledMsg DB 'Резидент не был загружен.$'

151 E2 20 AD A5 20 A1 EB+

152 AB 20 A7 A0 A3 E0 E3+

153 A6 A5 AD 2E 24

154

155 03FD 90 A5 A7 A8 A4 A5 AD+ removedMsg DB 'Резидент выгружен из памяти.'

156 E2 20 A2 EB A3 E0 E3+

157 A6 A5 AD 20 A8 A7 20+

158 AF A0 AC EF E2 A8 2E

159 =001C removedMsg\_length EQU $-removedMsg

160

161 0419 8D A5 20 E3 A4 A0 AB+ noRemoveMsg DB 'Не удалось выгрузить резидент'

162 AE E1 EC 20 A2 EB A3+

163 E0 E3 A7 A8 E2 EC 20+

164 E0 A5 A7 A8 A4 A5 AD+

165 E2

166 =001D noRemoveMsg\_length EQU $-noRemoveMsg

167

168 =00FF true EQU 0FFh ; нужно для удобства использования not с флагами

169 ; 0FFh = 11111111b = инверсия 00000000b

170

171 ; новый обработчик прерывания int 9h

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 4

tsrinout.asm

172 ; (работа с клавиатурой)

173 0436 new\_int9h proc far

174 0436 56 50 53 51 52 06 1E push SI AX BX CX DX ES DS ; сохраняем значения всех, изменяемых регистров+

175 в стеке

176 043D 0E push CS ; синхронизируем CS и DS

177 043E 1F pop DS

178

179 043F B8 0040 mov AX, 40h ; 40h - сегмент, где хранятся флаги состояния +

180 клавиатуры

181 0442 8E C0 mov ES, AX

182 0444 E4 60 in AL, 60h ; записываем в AL сканкод нажатой клавиши

183 ; @@@ следующий блок отвечает за выгрузку по Ctrl-U: @@@

184 0446 3C 16 cmp AL, 22 ; была нажата клавиша U?

185 0448 75 24 jne \_test\_Fx

186 044A 26: 8A 26 0017 mov AH, ES:[17h] ; флаги клавиатуры

187 044F 80 E4 0F and AH, 00001111b

188 0452 80 FC 04 cmp AH, 00000100b ; был ли нажат Ctrl?

189 0455 75 17 jne \_test\_Fx

190

191 0457 B4 FF mov AH, 0FFh ; завершаем обработку нажатия

192 0459 B0 01 mov AL, 01h ; и выгружаем резидент

193 045B CD 2F int 2Fh

194

195 045D E4 61 in AL, 61h ; контроллер состояния клавиатуры

196 045F 0C 80 or AL, 10000000b ; пометим, что клавишу нажали

197 0461 E6 61 out 61h, AL

198 0463 24 7F and AL, 01111111b ; пометим, что клавишу отпустили

199 0465 E6 61 out 61h, AL

200 0467 B0 20 mov AL, 20h

201 0469 E6 20 out 20h, AL ; отправим в контроллер прерываний признак +

202 конца прерывания

203

204 046B E9 0094 jmp \_quit ; выходим из процедуры

205

206 ; @@@ конец блока @@@

207 046E \_test\_Fx: ; проверка F1-F4

208 046E 2C 3A sub AL, 58 ; в AL теперь номер функциональной клавиши

209 0470 \_F1:

210 0470 3C 01 cmp AL, 1 ; F1

211 0472 75 07 jne \_F2

212 0474 F6 16 0118r not signaturePrintingEnabled

213 0478 EB 25 90 jmp \_translateOrIgnore

214 047B \_F2:

215 047B 3C 02 cmp AL, 2 ; F2

216 047D 75 0A jne \_F3

217 047F F6 16 0387r not cursiveEnabled

218 0483 E8 0084 call toggleCursive ; перевод символа в курсив и обратно

219 ; в зависимости от флага cursiveEnabled

220 0486 EB 17 90 jmp \_translateOrIgnore

221 0489 \_F3:

222 0489 3C 03 cmp AL, 3 ; F3

223 048B 75 07 jne \_F4

224 048D F6 16 0117r not translateEnabled

225 0491 EB 0C 90 jmp \_translateOrIgnore

226 0494 \_F4:

227 0494 3C 04 cmp AL, 4 ; F4

228 0496 75 07 jne \_translateOrIgnore

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 5

tsrinout.asm

229 0498 F6 16 0109r not ignoreEnabled

230 049C EB 01 90 jmp \_translateOrIgnore

231

232 049F \_translateOrIgnore: ; просто выводим набранный символ на экран

233 049F 9C pushf

234 04A0 2E: FF 1E 03A9r call dword ptr CS:[old\_int9hOffset] ; вызываем стандартный обработчик прерывания

235 04A5 B8 0040 mov AX, 40h ; 40h - сегмент, где хранятся флаги состояния +

236 клавиатуры

237 04A8 8E C0 mov ES, AX

238 04AA 26: 8B 1E 001C mov BX, ES:[1Ch] ; адрес хвоста

239 04AF 83 EB 02 sub BX, 2h ; сместимся назад к последнему введённому +

240 символу

241 04B2 83 FB 1E cmp BX, 1Eh ; не вышли ли мы за пределы буфера?

242 04B5 73 03 jae \_go

243 04B7 BB 003C mov BX, 3Ch ; хвост вышел за пределы буфера: значит, +

244 последний

245 ; введённый символ находится в конце буфера

246 04BA \_go:

247 04BA 26: 8B 17 mov DX, ES:[BX] ; в DX 0 введённый символ

248 04BD 80 3E 0109r FF cmp ignoreEnabled, true ; включен ли режим блокировки ввода?

249 04C2 75 1B jne \_checkTranslate

250

251 04C4 BE 0000 mov SI, 0 ; да, включен

252 04C7 8A 0E 0108r mov CL, ignoredLength ; количество игнорируемых символов

253

254 04CB \_checkIgnored:

255 04CB 3A 94 0103r cmp DL, ignoredChars[SI] ; проверяем, присутствует ли текущий символ в +

256 списке игнорируемых

257 04CF 74 06 je \_block

258 04D1 46 inc SI

259 04D2 E2 F7 loop \_checkIgnored ; зацикливаем ignoredLength раз

260 04D4 EB 09 90 jmp \_checkTranslate

261

262 ; блокируем

263 04D7 \_block:

264 04D7 26: 89 1E 001C mov ES:[1Ch], BX ; блокировка ввода символа

265 ; если по варианту нужно не блокировать ввод символа,

266 ; а заменять одни символы другими, замените строку выше строкой

267 ; mov ES:[BX], AX

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

269 звёздочки

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

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

272 ; и раскомментировать строки ниже:

273 ; xor AX, AX

274 ; mov AL, replaceWith[SI]

275 ; mov ES:[BX], AX ; замена символа

276 04DC EB 24 90 jmp \_quit

277

278 04DF \_checkTranslate:

279 04DF 80 3E 0117r FF cmp translateEnabled, true ; включен ли режим перевода?

280 04E4 75 1C jne \_quit

281

282 04E6 BE 0000 mov SI, 0 ; да, включен

283 04E9 8A 0E 0116r mov CL, translateLength ; кол-во символов для перевода

284

285 04ED \_checkTranslateLoop:

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 6

tsrinout.asm

286 04ED 3A 94 010Ar cmp DL, translateFrom[SI] ; присутствует ли текущий символ в списке для +

287 перевода?

288 04F1 74 06 je \_translate

289 04F3 46 inc SI

290 04F4 E2 F7 loop \_checkTranslateLoop ; продолжаем, пока не закончим проверять каждый+

291 символ

292 04F6 EB 0A 90 jmp \_quit

293

294 04F9 \_translate:

295 04F9 33 C0 xor AX, AX ; переводим

296 04FB 8A 84 0110r mov AL, translateTo[SI]

297 04FF 26: 89 07 mov ES:[BX], AX ; замена символа

298

299 0502 \_quit:

300 0502 1F 07 5A 59 5B 58 5E pop DS ES DX CX BX AX SI ; восстанавливаем все регистры

301 0509 CF iret

302 050A new\_int9h endp

303

304 ; в зависимости от флага cursiveEnabled меняет начертание символа на курсив и обратно

305 ; сама смена происходит в процедуре changeFont - здесь же подготавливаются данные

306 050A toggleCursive proc

307 050A 06 50 push ES AX ; сохраняем регистры

308 050C 0E push CS

309 050D 07 pop ES

310

311 050E 80 3E 0387r FF cmp cursiveEnabled, true ; если флаг равен true,

312 0513 75 30 jne \_restoreSymbol ; выполняем замену символа на курсивный вариант,

313 ; предварительно сохраняя старый символ в +

314 savedSymbol

315

316 0515 E8 004E call saveFont

317 0518 8A 0E 0398r mov CL, charToCursiveIndex

318 051C \_shiftTable:

319 051C 83 C5 10 add BP, 16 ; получаем в BP таблицу всех символов. адрес +

320 указывает на символ 0

321 ; поэтому нужно совершить сдвиг 16\*X - где X - +

322 код символа

323 051F E2 FB loop \_shiftTable

324

325 0521 1E push DS ; пpи savefont смещается pегистp ES

326 0522 58 pop AX ; поэтомy пpиходится делать такие махинации, +

327 чтобы

328 0523 06 push ES ; записать полyченный элемент в savedSymbol

329 0524 1F pop DS

330 0525 50 push AX ; DS -> AX, ES -> DS, AX -> ES => ES и DS +

331 поменялись местами

332 0526 07 pop ES ; + сохранение старого значения DS в AX

333 0527 50 push AX

334

335 0528 8B F5 mov SI, BP

336 052A BF 0399r lea DI, savedSymbol ; сохpаняем в пеpеменнyю savedSymbol таблицу +

337 нужного символа

338

339 052D B9 0010 mov CX, 16 ; movsb из DS:SI в ES:DI

340

341 0530 F3> A4 rep movsb ; исходные позиции сегментов возвpащены

342 0532 1F pop DS ; восстановление DS

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 7

tsrinout.asm

343

344 0533 B9 0001 mov CX, 1 ; заменим написание символа на кypсив

345 0536 B6 00 mov DH, 0

346 0538 8A 16 0398r mov DL, charToCursiveIndex

347 053C BD 0388r lea BP, cursiveSymbol

348 053F E8 0015 call changeFont

349 0542 EB 10 90 jmp \_exitToggleCursive

350

351 0545 \_restoreSymbol:

352 0545 B9 0001 mov CX, 1 ; если флаг равен 0, заменяем курсивный символ +

353 на старый вариант

354 0548 B6 00 mov DH, 0

355 054A 8A 16 0398r mov DL, charToCursiveIndex

356 054E BD 0399r lea BP, savedSymbol

357 0551 E8 0003 call changeFont

358

359 0554 \_exitToggleCursive:

360 0554 58 pop AX

361 0555 07 pop ES

362 0556 C3 ret

363 0557 toggleCursive endp

364

365 ; функция смены начертания символа (курсив/нормальное)

366 ;

367 ; входные данные:

368 ; 1) DL = номер символа для замены

369 ; 2) CX = количество символов заменяемых изображений символов

370 ; (начиная с символа указанного в DX)

371 ; 3) ES:BP = адрес таблицы

372 ;

373 ; описание работы процедуры:

374 ; 1) происходит вызов int 10h (видеосервис)

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

376 ; параметр AL = 0 сообщает, что будет заменено изображение

377 ; символа для текущего шрифта.

378 ; в случаях, когда AL = 1 или 2, будет заменено изображение

379 ; только для определенного шрифта (8x14 и 8x8 соответственно)

380 ; 2) параметр BH = 0Eh сообщает, что на определение каждого изображения символа

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

382 ; 3) параметр BL = 0 - блок шрифта для загрузки (от 0 до 4)

383 ;

384 ; результат:

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

386 ; на предложенное пользователем.

387 ; изменению подвергнутся все символы, находящиеся на экране:

388 ; таким образом, если изображение заменено, старый вариант нигде уже не проявится

389 0557 changeFont proc

390 0557 50 53 52 push AX BX DX

391 055A B8 1100 mov AX, 1100h

392 055D BB 1000 mov BX, 1000h

393 0560 CD 10 int 10h

394 0562 5A 5B 58 pop DX BX AX

395 0565 C3 ret

396 0566 changeFont endp

397

398 ; функция сохранения нормального начертания символа

399 ;

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 8

tsrinout.asm

400 ; входные данные:

401 ; BH - тип возвращаемой символьной таблицы

402 ; = 0 - таблица из int 1fh

403 ; = 1 - таблица из int 44h

404 ; = 2..5 - таблица из 8x14, 8x8, 8x8 (top), 9x14

405 ; = 6 - 8x16

406 ;

407 ; описание работы процедуры:

408 ; происходит вызов int 10h (видеосервис)

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

410 ; параметр AL = 30 - подфункция получения информации о EGA

411 ;

412 ; результат:

413 ; 1) в ES:BP находится таблица символов (полная)

414 ; 2) в CX находится байт на символ

415 ; 3) в DL количество экранных строк

416 ; важно! происходит сдвиг регистра ES (ES = C000h)

417 0566 saveFont proc

418 0566 50 53 52 push AX BX DX

419 0569 B8 1130 mov AX, 1130h

420 056C BB 0600 mov BX, 0600h

421 056F CD 10 int 10h

422 0571 5B 58 5A pop BX AX DX

423 0574 C3 ret

424 0575 saveFont endp

425

426 ; обработчик прерывания int 2Fh

427 ; служит для:

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

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

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

431 0575 new\_int2Fh proc

432 0575 80 FC FF cmp AH, 0FFh ; наша процедура?

433 0578 75 0B jne \_2Fh\_default ; нет - на стандартный обработчик

434 057A 3C 00 cmp AL, 0 ; подпроцедура проверки, загружен ли резидент в+

435 память?

436 057C 74 0C je \_alreadyInstalled2Fh

437 057E 3C 01 cmp AL, 1 ; подпроцедура выгрузки из памяти?

438 0580 74 0B je \_uninstall

439 0582 EB 01 90 jmp \_2Fh\_default

440

441 0585 \_2Fh\_default:

442 0585 2E: FF 2E 03B1r jmp dword ptr CS:[old\_int2FhOffset] ; вызов стандартного обработчика

443

444 058A \_alreadyInstalled2Fh:

445 058A B4 69 mov AH, 'i' ; пусть AH = 'i', если резидент уже загружен в +

446 память

447 058C CF iret ; конечно, вместо 'i' может быть любое значение

448

449 058D \_uninstall: ; подпроцедура выгрузки из памяти

450 058D 1E 06 52 53 push DS ES DX BX

451 0591 33 DB xor BX, BX

452

453 0593 0E push CS ; CS = ES, для доступа к переменным

454 0594 07 pop ES

455

456 0595 B8 2509 mov AX, 2509h

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 9

tsrinout.asm

457 0598 26: 8B 16 03A9r mov DX, ES:old\_int9hOffset ; возвращаем вектор прерывания 09h на место

458 059D 26: 8E 1E 03ABr mov DS, ES:old\_int9hSegment

459 05A2 CD 21 int 21h

460

461 05A4 B8 251C mov AX, 251Ch

462 05A7 26: 8B 16 03ADr mov DX, ES:old\_int1ChOffset ; возвращаем вектор прерывания 1Ch на место

463 05AC 26: 8E 1E 03AFr mov DS, ES:old\_int1ChSegment

464 05B1 CD 21 int 21h

465

466 05B3 B8 252F mov AX, 252Fh

467 05B6 26: 8B 16 03B1r mov DX, ES:old\_int2FhOffset ; возвращаем вектор прерывания 2Fh на место

468 05BB 26: 8E 1E 03B3r mov DS, ES:old\_int2FhSegment

469 05C0 CD 21 int 21h

470

471 05C2 2E: 8E 06 002C mov ES, CS:2Ch ; загрузим в ES адрес окружения

472 05C7 B4 49 mov AH, 49h ; выгрузим из памяти окружение

473 05C9 CD 21 int 21h

474 05CB 72 0B jc \_notRemove

475

476 05CD 0E push CS

477 05CE 07 pop ES ; в ES - адрес резидентной программы

478 05CF B4 49 mov AH, 49h ; выгрузим из памяти резидент

479 05D1 CD 21 int 21h

480

481 05D3 72 03 jc \_notRemove

482 05D5 EB 15 90 jmp \_unloaded

483

484 05D8 \_notRemove: ; не удалось выполнить выгрузку => вывод ошибки

485 05D8 B4 03 mov AH, 03h ; получаем позицию курсора

486 05DA CD 10 int 10h

487 05DC BD 0419r lea BP, noRemoveMsg

488 05DF B9 001D mov CX, noRemoveMsg\_length

489 05E2 B3 07 mov BL, 0111b

490 05E4 B8 1301 mov AX, 1301h

491 05E7 CD 10 int 10h

492 05E9 EB 12 90 jmp \_2Fh\_exit

493

494 05EC \_unloaded: ; выгрузка прошла успешно => вывод сообщения

495 05EC B4 03 mov AH, 03h ; получаем позицию курсора

496 05EE CD 10 int 10h

497 05F0 BD 03FDr lea BP, removedMsg

498 05F3 B9 001C mov CX, removedMsg\_length

499 05F6 B3 07 mov BL, 0111b

500 05F8 B8 1301 mov AX, 1301h

501 05FB CD 10 int 10h

502

503 05FD \_2Fh\_exit:

504 05FD 5B 5A 07 1F pop BX DX ES DS

505 0601 CF iret

506 0602 new\_int2Fh endp

507

508 ; обработчик прерывания int 1Ch

509 ; вызывается каждые 55 мс

510 0602 new\_int1Ch proc far

511 0602 50 push AX

512 0603 0E push CS

513 0604 1F pop DS

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 10

tsrinout.asm

514

515 0605 9C pushf

516 0606 2E: FF 1E 03ADr call dword ptr CS:[old\_int1ChOffset] ; вызываем стандартный обработчик прерывания

517

518 060B 80 3E 0118r FF cmp signaturePrintingEnabled, true ; если нажата управляющая клавиша (в данном +

519 случае F1)

520 0610 75 1B jne \_notToPrint

521

522 0612 83 3E 0119r 25 cmp counter, printDelay\*1000/55 + 1 ; если кол-во "тактов" равно printDelay секундам

523 0617 74 03 je \_letsPrint

524

525 0619 EB 0E 90 jmp \_dontPrint

526

527 061C \_letsPrint:

528 061C F6 16 0118r not signaturePrintingEnabled

529 0620 C7 06 0119r 0000 mov counter, 0

530 0626 E8 0016 call printSignature ; выводим подпись на экран

531

532 0629 \_dontPrint:

533 0629 FF 06 0119r inc counter ; увеличим значение счетчика на 1

534

535 062D \_notToPrint:

536 062D 58 pop AX

537 062E CF iret

538 062F new\_int1Ch endp

539

540 ; выводит одну строку подписи

541 062F printSignatureLine proc

542 062F 52 push DX

543 0630 8B 0E 011Br mov CX, signatureLineLength

544 0634 B3 07 mov BL, 0111b ; цвет выводимого текста

545 0636 B8 1301 mov AX, 1301h ; AH = 13h - номер ф-ии, AL = 01h - перемещение+

546 курсора

547 0639 CD 10 int 10h

548 063B 5A pop DX

549 063C FE C6 inc DH

550 063E C3 ret

551 063F printSignatureLine endp

552

553 ; процедура вывода подписи

554 063F printSignature proc

555 063F 50 52 51 53 06 54 55+ push AX DX CX BX ES SP BP SI DI

556 56 57

557

558 0648 33 C0 xor AX, AX ; обнуляем значения регистров

559 064A 33 DB xor BX, BX

560 064C 33 D2 xor DX, DX

561

562 064E B4 03 mov AH, 03h ; чтение текущей позиции курсора

563 0650 CD 10 int 10h

564 0652 52 push DX ; помещаем информацию о положении курсора в стек

565

566 0653 BA 090F mov DX, 090Fh ; NB! вверху: 000Fh, посередине: 090Fh, внизу: +

567 130Fh

568

569 0656 \_actualPrint:

570 0656 B4 0F mov AH, 0Fh ; чтение текущего видеорежима. в BH - текущая +

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 11

tsrinout.asm

571 страница

572 0658 CD 10 int 10h

573

574 065A 0E push CS

575 065B 07 pop ES ; указываем ES на CS

576

577 065C BD 01B9r lea BP, tableTop

578 065F E8 FFCD call printSignatureLine ; выводим верх таблицы

579 0662 BD 011Dr lea BP, signatureLine1

580 0665 E8 FFC7 call printSignatureLine ; выводим первую строку

581 0668 BD 0151r lea BP, signatureLine2

582 066B E8 FFC1 call printSignatureLine ; выводим вторую строку

583 066E BD 0185r lea BP, signatureLine3

584 0671 E8 FFBB call printSignatureLine ; выводим третью строку

585 0674 BD 01EDr lea BP, tableBottom

586 0677 E8 FFB5 call printSignatureLine ; выводим низ таблицы

587

588 067A 33 DB xor BX, BX

589 067C 5A pop DX ; восстанавливаем из стека прежнее положение +

590 курсора

591 067D B4 02 mov AH, 02h ; меняем положение курсора на первоначальное

592 067F CD 10 int 10h

593

594 0681 5F 5E 5D 5C 07 5B 59+ pop DI SI BP SP ES BX CX DX AX

595 5A 58

596 068A C3 ret

597 068B printSignature endp

598

599 ; Основная часть программы

600 ; 1) установка видеорежима

601 ; 2) проверка, запущен ли резидент

602 ; 3) установка вектора прерываний

603 068B \_initTSR:

604 068B B4 03 mov AH, 03h

605 068D CD 10 int 10h

606 068F 52 push DX

607 0690 B4 00 mov AH, 00h ; установка видеорежима

608 0692 B0 83 mov AL, 83h

609 0694 CD 10 int 10h

610 0696 5A pop DX

611 0697 B4 02 mov AH, 02h

612 0699 CD 10 int 10h

613

614 069B E8 009F call commandParamsParser ; читаем аргументы командной строки

615 069E 80 3E 0386r 02 cmp commandLineResult, 2 ; если результат = 2, значит была выведена +

616 справка

617 06A3 75 03 jne \_shouldContinue ; соответственно, никаких других действий +

618 делать не нужно

619 06A5 E9 0093 jmp \_exit

620 06A8 \_shouldContinue:

621 ; ### следующий блок отвечает за выгрузку при аргументе командной строки /u и при простом +

622 перезапуске ###

623 06A8 80 3E 0386r 01 cmp commandLineResult, 1 ; проверяем результат работы процедуры

624 06AD 75 14 jne \_go\_on

625 06AF B4 FF mov AH, 0FFh

626 06B1 B0 00 mov AL, 0

627 06B3 CD 2F int 2Fh ; проверка того, загружена ли уже программа

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 12

tsrinout.asm

628 06B5 80 FC 69 cmp AH, 'i' ; если запущена, то AH = 'i' (см. процедуру +

629 new\_int2Fh)

630 06B8 74 64 je \_remove

631

632 06BA B4 09 mov AH, 09h

633 06BC BA 03E3r lea DX, notInstalledMsg ; не была загружена

634 06BF CD 21 int 21h

635 06C1 CD 20 int 20h

636 06C3 \_go\_on:

637 ; ### конец блока ###

638 ; @@@ отвечает за выгрузку при перезапуске @@@

639 06C3 B4 FF mov AH, 0FFh ; ещё раз проверяем, запущен ли резидент сейчас

640 06C5 B0 00 mov AL, 0

641 06C7 CD 2F int 2Fh

642 06C9 80 FC 69 cmp AH, 'i' ; если запущена, то AH = 'i' (см. процедуру +

643 new\_int2Fh)

644 06CC 74 64 je \_alreadyInstalled

645 ; @@@ конец блока @@@

646

647 06CE B8 3509 mov AX, 3509h ; получить в ES:BX прерывания 09h

648 06D1 CD 21 int 21h

649 06D3 2E: 89 1E 03A9r mov word ptr CS:old\_int9hOffset, BX ; обработчик прерывания 09h

650 06D8 2E: 8C 06 03ABr mov word ptr CS:old\_int9hSegment, ES

651 06DD B8 2509 mov AX, 2509h ; установим вектор на прерывание 09h

652 06E0 BA 0436r mov DX, offset new\_int9h

653 06E3 CD 21 int 21h

654

655 06E5 B8 351C mov AX, 351Ch ; получить в ES:BX прерывания 1Ch

656 06E8 CD 21 int 21h

657 06EA 2E: 89 1E 03ADr mov word ptr CS:old\_int1ChOffset, BX ; обработчик прерывания 1Ch

658 06EF 2E: 8C 06 03AFr mov word ptr CS:old\_int1ChSegment, ES

659 06F4 B8 251C mov AX, 251Ch ; установим вектор на прерывание 1Ch

660 06F7 BA 0602r mov DX, offset new\_int1Ch

661 06FA CD 21 int 21h

662

663 06FC B8 352F mov AX, 352Fh ; получить в ES:BX прерывания 2Fh

664 06FF CD 21 int 21h

665 0701 2E: 89 1E 03B1r mov word ptr CS:old\_int2FhOffset, BX ; обработчик прерывания 2Fh

666 0706 2E: 8C 06 03B3r mov word ptr CS:old\_int2FhSegment, ES

667 070B B8 252F mov AX, 252Fh ; установим вектор на прерывание 2Fh

668 070E BA 0575r mov DX, offset new\_int2Fh

669 0711 CD 21 int 21h

670

671 0713 BB 03B5r lea BX, installedMsg ; выводим сообщение, что всё ОК

672 0716 E8 0074 call printStr

673

674 0719 BA 068Br mov DX, offset \_initTSR ; остаемся в памяти и выходим из основной части

675 071C CD 27 int 27h

676

677 071E \_remove: ; выгрузка из памяти

678 071E 06 push ES

679 071F A1 002C mov AX, DS:[2Ch] ; PSP

680 0722 8E C0 mov ES, AX

681 0724 B4 49 mov AH, 49h ; хватит памяти чтоб остаться резидентом?

682 0726 CD 21 int 21h

683 0728 07 pop ES

684

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 13

tsrinout.asm

685 0729 B4 FF mov AH, 0FFh

686 072B B0 01 mov AL, 1

687 072D CD 2F int 2Fh

688 072F EB 0A 90 jmp \_exit

689 0732 \_alreadyInstalled: ; резидент уже запущен

690 0732 BB 03C8r lea BX, alreadyInstalledMsg

691 0735 E8 0055 call printStr

692 0738 EB 01 90 jmp \_exit

693 073B \_exit: ; выход

694 073B CD 20 int 20h

695

696 ; парсер аргментов командной строки. выводит справку.

697 ; устанавливает флаг commandLineResult:

698 ; 0 = всё ОК; 1 = нужна выгрузка; 2 = была выведена справка, не нужно загружать резидент

699 073D commandParamsParser proc

700 073D 0E push CS

701 073E 07 pop ES

702

703 073F BE 0080 mov SI, 80h ; SI = смещение командной строки

704 0742 AC lodsb ; получим кол-во символов

705 0743 0A C0 or AL, AL ; если 0 символов введено,

706 0745 74 30 jz \_paramParsingEndWithUnload ; ### то дополнительная проверка, был ли уже +

707 загружен

708 ; ### резидент. в таком случае он выгружается

709

710 0747 \_nextChar:

711 0747 46 inc SI ; теперь SI указывает на первый символ строки

712

713 0748 80 3C 00 cmp [SI], BYTE ptr 0

714 074B 74 3A je \_paramParsingEnd

715

716 074D AD lodsw ; получаем два символа

717 074E 3D 3F2F cmp AX, '?/'

718 0751 74 16 je \_displayHelp

719 ; @@@ следует раскомментировать, если нужно выгружать по аргументу /u @@@

720 0753 3D 752F cmp AX, 'u/'

721 0756 74 07 je \_finishTSR

722 0758 3D 552F cmp AX, 'U/'

723 075B 74 02 je \_finishTSR

724 075D EB E8 jmp \_nextChar

725

726 075F \_finishTSR:

727 075F C6 06 0386r 01 mov commandLineResult, 1 ; флаг того, что необходимо выгрузить резидент

728 0764 EB E1 jmp \_nextChar

729 ; @@@ конец блока @@@

730

731 0766 EB 1F 90 jmp \_paramParsingEnd

732 0769 \_displayHelp:

733 0769 BB 0221r lea BX, helpMsg ; выводим справку

734 076C E8 001E call printStr

735 076F C6 06 0386r 02 mov commandLineResult, 2 ; флаг того, что резидент загружать не надо

736

737 ; ### далее - проверка: если резидент уже загружен, то выгрузить ###

738 0774 EB 11 90 jmp \_paramParsingEnd

739 0777 \_paramParsingEndWithUnload:

740 0777 B4 FF mov AH, 0FFh

741 0779 B0 00 mov AL, 0

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 14

tsrinout.asm

742 077B CD 2F int 2Fh

743 077D 80 FC 69 cmp AH, 'i' ; проверка того, загружена ли уже программа

744 0780 75 05 jne \_paramParsingEnd

745

746 0782 C6 06 0386r 01 mov commandLineResult, 1

747 ; ### конец блока ###

748

749 0787 \_paramParsingEnd:

750 0787 C3 ret

751 0788 commandParamsParser endp

752

753 ; отображает символ из AL

754 0788 printChar proc

755 0788 B4 0E mov AH, 0EH

756 078A CD 10 int 010H

757 078C C3 ret

758 078D printChar endp

759

760 ; отображает нуль-терминированную строку из [BX]

761 078D printStr proc

762 078D 52 50 push DX AX

763 078F 8B 07 mov AX, [BX]

764 0791 \_printStrLoop:

765 0791 3C 00 cmp AL, 0

766 0793 74 08 je \_printStrEnd

767 0795 E8 FFF0 call printChar

768 0798 43 inc BX

769 0799 8B 07 mov AX, [BX]

770 079B EB F4 jmp \_printStrLoop

771 079D \_printStrEnd:

772 079D 58 5A pop AX DX

773 079F C3 ret

774 07A0 printStr endp

775

776 07A0 code ends

777 end \_start

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 15

Symbol Table

Symbol Name Type Value

??DATE Text "04/25/18"

??FILENAME Text "tsrinout"

??TIME Text "13:03:38"

??VERSION Number 030A

@CPU Text 0101H

@CURSEG Text CODE

@FILENAME Text TSRINOUT

@WORDSIZE Text 2

ALREADYINSTALLEDMSG Byte CODE:03C8

CHANGEFONT Near CODE:0557

CHARTOCURSIVEINDEX Byte CODE:0398

COMMANDLINERESULT Byte CODE:0386

COMMANDPARAMSPARSER Near CODE:073D

COUNTER Word CODE:0119

CURSIVEENABLED Byte CODE:0387

CURSIVESYMBOL Byte CODE:0388

HELPMSG Byte CODE:0221

HELPMSGLENGTH Number 0165

IGNOREDCHARS Byte CODE:0103

IGNOREDLENGTH Byte CODE:0108

IGNOREENABLED Byte CODE:0109

INSTALLEDMSG Byte CODE:03B5

NEW\_INT1CH Far CODE:0602

NEW\_INT2FH Near CODE:0575

NEW\_INT9H Far CODE:0436

NOREMOVEMSG Byte CODE:0419

NOREMOVEMSG\_LENGTH Number 001D

NOTINSTALLEDMSG Byte CODE:03E3

OLD\_INT1CHOFFSET Word CODE:03AD

OLD\_INT1CHSEGMENT Word CODE:03AF

OLD\_INT2FHOFFSET Word CODE:03B1

OLD\_INT2FHSEGMENT Word CODE:03B3

OLD\_INT9HOFFSET Word CODE:03A9

OLD\_INT9HSEGMENT Word CODE:03AB

PRINTCHAR Near CODE:0788

PRINTDELAY Number 0002

PRINTSIGNATURE Near CODE:063F

PRINTSIGNATURELINE Near CODE:062F

PRINTSTR Near CODE:078D

REMOVEDMSG Byte CODE:03FD

REMOVEDMSG\_LENGTH Number 001C

SAVEDSYMBOL Byte CODE:0399

SAVEFONT Near CODE:0566

SIGNATURELINE1 Byte CODE:011D

SIGNATURELINE2 Byte CODE:0151

SIGNATURELINE3 Byte CODE:0185

SIGNATURELINELENGTH Word CODE:011B

SIGNATUREPRINTINGENABLED Byte CODE:0118

TABLEBOTTOM Byte CODE:01ED

TABLETOP Byte CODE:01B9

TOGGLECURSIVE Near CODE:050A

TRANSLATEENABLED Byte CODE:0117

TRANSLATEFROM Byte CODE:010A

TRANSLATELENGTH Byte CODE:0116

Turbo Assembler Version 3.1 04/25/18 13:03:38 Page 16

Symbol Table

TRANSLATETO Byte CODE:0110

TRUE Number 00FF

\_2FH\_DEFAULT Near CODE:0585

\_2FH\_EXIT Near CODE:05FD

\_ACTUALPRINT Near CODE:0656

\_ALREADYINSTALLED Near CODE:0732

\_ALREADYINSTALLED2FH Near CODE:058A

\_BLOCK Near CODE:04D7

\_CHECKIGNORED Near CODE:04CB

\_CHECKTRANSLATE Near CODE:04DF

\_CHECKTRANSLATELOOP Near CODE:04ED

\_DISPLAYHELP Near CODE:0769

\_DONTPRINT Near CODE:0629

\_EXIT Near CODE:073B

\_EXITTOGGLECURSIVE Near CODE:0554

\_F1 Near CODE:0470

\_F2 Near CODE:047B

\_F3 Near CODE:0489

\_F4 Near CODE:0494

\_FINISHTSR Near CODE:075F

\_GO Near CODE:04BA

\_GO\_ON Near CODE:06C3

\_INITTSR Near CODE:068B

\_LETSPRINT Near CODE:061C

\_NEXTCHAR Near CODE:0747

\_NOTREMOVE Near CODE:05D8

\_NOTTOPRINT Near CODE:062D

\_PARAMPARSINGEND Near CODE:0787

\_PARAMPARSINGENDWITHUNLOAD Near CODE:0777

\_PRINTSTREND Near CODE:079D

\_PRINTSTRLOOP Near CODE:0791

\_QUIT Near CODE:0502

\_REMOVE Near CODE:071E

\_RESTORESYMBOL Near CODE:0545

\_SHIFTTABLE Near CODE:051C

\_SHOULDCONTINUE Near CODE:06A8

\_START Near CODE:0100

\_TEST\_FX Near CODE:046E

\_TRANSLATE Near CODE:04F9

\_TRANSLATEORIGNORE Near CODE:049F

\_UNINSTALL Near CODE:058D

\_UNLOADED Near CODE:05EC

Groups & Segments Bit Size Align Combine Class

CODE 16 07A0 Para none CODE

Turbo Assembler Version 3.1 02/11/19 13:37:54 Page 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

Turbo Assembler Version 3.1 02/11/19 13:37:54 Page 2

Symbol Table

Symbol Name Type Value

??DATE Text "02/11/19"

??FILENAME Text "unloader"

??TIME Text "13:37:54"

??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