**Московский государственный технический университет**

**им. Н.Э. Баумана**

Утверждаю:

Большаков С.А.

"\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Курсовая работа по курсу «Системное программирование»

**«Резидентная программа (TSR)»**

Техническое описание

(вид документа)

писчая бумага

(вид носителя)

28

(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-43Б \_\_\_\_\_Пермяков Д. К.\_\_\_\_\_

"15" Мая 2023 г.

Москва – 2023

**СОДЕРЖАНИЕ**

[**1. Файл tsr.lst** 3](#_Toc134996325)

[**2. Файл unloader.lst** 27](#_Toc134996326)

# **1. Файл tsr.lst**

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 1

tsr.asm

1

2 ; =======================================

3 ; kr.asm

4 ;

5 ; Сборка:

6 ; > tasm.exe /l kr.asm

7 ; > tlink /t /x kr.obj

8 ; =======================================

9

10 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

11 ; tsr.asm

12 ; 14.04.23

13 ; Сборка:

14 ; tasm.exe /l tsr.asm

15 ; tlink /t /x tsr.obj

16 ;

17 ; Примечания:

18 ; 1) комменатрии, начинающиеся с символа @ - места, где код зависит от варианта

19 ;

20 ; Авторы:

21 ; МГТУ им. Н.Э. Баумана, ИУ5-44, 2013 г.

22 ; Леонтьев А.В.

23 ; Латкин И.И.

24 ; Назаров К.В.

25 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

26

27 0000 code segment 'code'

28 assume CS:code, DS:code

29 org 100h

30 0100 \_start:

31

32 0100 E9 0671 jmp \_initTSR ; на начало программы

33

34 ; данные

35 0103 41 42 43 44 45 46 47+ ignoredChars DB +

36 48 49 4A 4B 4C 4D 4E+ 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz' ;@ список игнорируемых символов

37 4F 50 51 52 53 54 55+

38 56 57 58 59 5A 61 62+

39 63 64 65 66 67 68 69+

40 6A 6B 6C 6D 6E 6F 70+

41 71 72 73 74 75 76 77+

42 78 79 7A

43 =0034 ignoredLength equ $-ignoredChars +

44 ; длина строки ignoredChars

45 0137 00 ignoreEnabled DB 0 +

46 ; флаг функции игнорирования ввода

47 0138 4B 56 59 4A 47 translateFrom DB 'KVYJG' +

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

49 013D 8B 8C 8D 8E 8F translateTo DB 'ЛМНОП' +

50 ;@ символы на которые будет идти замена

51 =0005 translateLength equ $-translateTo +

52 ; длина строки trasnlateFrom

53 0142 00 translateEnabled DB 0 +

54 ; флаг функции перевода

55

56 0143 00 signaturePrintingEnabled DB 0 +

57 ; флаг функции вывода информации об авторе

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 2

tsr.asm

58 0144 00 cursiveEnabled DB 0 +

59 ; флаг перевода символа в курсив

60

61 0145 00 cursiveSymbol DB 00000000b +

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

63 0146 00 DB 00000000b

64 0147 00 DB 00000000b

65 0148 63 DB 01100011b

66 0149 63 DB 01100011b

67 014A 67 DB 01100111b

68 014B 6B DB 01101011b

69 014C 6B DB 01101011b

70 014D D6 DB 11010110b

71 014E EC DB 11101100b

72 014F CC DB 11001100b

73 0150 CC DB 11001100b

74 0151 CC DB 11001100b

75 0152 00 DB 00000000b

76 0153 00 DB 00000000b

77 0154 00 DB 00000000b

78

79 0155 88 charToCursiveIndex DB 'И' +

80 ;@ символ для замены

81 0156 10\*(FF) savedSymbol DB 16 dup(0FFh) +

82 ; переменная для хранения старого символа

83

84 =00FF true equ 0FFh +

85 ; константа истинности

86 0166 ???? old\_int9hOffset DW ? +

87 ; адрес старого обработчика int 9h

88 0168 ???? old\_int9hSegment DW ? +

89 ; сегмент старого обработчика int 9h

90 016A ???? old\_int1ChOffset DW ? +

91 ; адрес старого обработчика int 1Ch

92 016C ???? old\_int1ChSegment DW ? +

93 ; сегмент старого обработчика int 1Ch

94 016E ???? old\_int2FhOffset DW ? +

95 ; адрес старого обработчика int 2Fh

96 0170 ???? old\_int2FhSegment DW ? +

97 ; сегмент старого обработчика int 2Fh

98

99 0172 00 unloadTSR DB 0 +

100 ; 1 - выгрузить резидент

101 0173 00 notLoadTSR DB 0 +

102 ; 1 - не загружать

103 0174 0000 counter DW 0

104 =0007 printDelay equ 7 +

105 ;@ задержка перед выводом "подписи" в секундах

106 0176 0001 printPos DW 1 +

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

108

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

110 (1я строка).

111 0178 B3 8C A0 AC AE E3 20+ signatureLine1 DB 179, Пермяков Д. К. +

112 80 2E 20 8C 2E 20 20+ ', 179

113 20 20 20 20 20 20 20+

114 20 20 20 20 20 20 20+

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 3

tsr.asm

115 20 20 20 20 20 20 20+

116 B3

117 =0024 Line1\_length equ $-signatureLine1

118 019C B3 88 93 35 2D 34 34+ signatureLine2 DB 179, 'ИУ5-43Б +

119 81 20 20 20 20 20 20+ ', 179

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 B3

124 =0024 Line2\_length equ $-signatureLine2

125 01C0 B3 82 A0 E0 A8 A0 AD+ signatureLine3 DB 179, 'Вариант #11 +

126 E2 20 23 31 31 20 20+ ', 179

127 20 20 20 20 20 20 20+

128 20 20 20 20 20 20 20+

129 20 20 20 20 20 20 20+

130 B3

131 =0024 Line3\_length equ $-signatureLine3

132 01E4 3E 74 73 72 2E 63 6F+ helpMsg DB '>tsr.com [/?]', 10, 13

133 6D 20 5B 2F 3F 5D 0A+

134 0D

135 01F3 20 5B 2F 3F 5D 20 2D+ DB ' [/?] - вывод данной справки', 10, 13

136 20 A2 EB A2 AE A4 20+

137 A4 A0 AD AD AE A9 20+

138 E1 AF E0 A0 A2 AA A8+

139 0A 0D

140 0211 20 20 46 39 20 20 2D+ DB ' F9 - вывод ФИО и группы по таймеру в центре экрана', 10, 13

141 20 A2 EB A2 AE A4 20+

142 94 88 8E 20 A8 20 A3+

143 E0 E3 AF AF EB 20 AF+

144 AE 20 E2 A0 A9 AC A5+

145 E0 E3 20 A2 20 E6 A5+

146 AD E2 E0 A5 20 ED AA+

147 E0 A0 AD A0 0A 0D

148 0248 20 20 46 31 20 20 2D+ DB ' F1 - включение и отключения курсивного вывода русского символа +

149 20 A2 AA AB EE E7 A5+ И', 10, 13

150 AD A8 A5 20 A8 20 AE+

151 E2 AA AB EE E7 A5 AD+

152 A8 EF 20 AA E3 E0 E1+

153 A8 A2 AD AE A3 AE 20+

154 A2 EB A2 AE A4 A0 20+

155 E0 E3 E1 E1 AA AE A3+

156 AE 20 E1 A8 AC A2 AE+

157 AB A0 20 88 0A 0D

158 028D 20 20 46 32 20 20 2D+ DB ' F2 - включение и отключение частичной русификации клавиатуры +

159 20 A2 AA AB EE E7 A5+ (KVYJG -> ЛМНОП)', 10, 13

160 AD A8 A5 20 A8 20 AE+

161 E2 AA AB EE E7 A5 AD+

162 A8 A5 20 E7 A0 E1 E2+

163 A8 E7 AD AE A9 20 E0+

164 E3 E1 A8 E4 A8 AA A0+

165 E6 A8 A8 20 AA AB A0+

166 A2 A8 A0 E2 E3 E0 EB+

167 28 4B 56 59 4A 47 20+

168 2D 3E 20 8B 8C 8D 8E+

169 8F 29 0A 0D

170 02DE 20 20 46 33 20 20 2D+ DB ‘ F3 - включение и отключение режима блокировки ввода латинских +

171 20 A2 AA AB EE E7 A5+ букв', 10, 13

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 4

tsr.asm

172 AD A8 A5 20 A8 20 AE+

173 E2 AA AB EE E7 A5 AD+

174 A8 A5 20 E0 A5 A6 A8+

175 AC A0 20 A1 AB AE AA+

176 A8 E0 AE A2 AA A8 20+

177 A2 A2 AE A4 A0 20 AB+

178 A0 E2 A8 AD E1 AA A8+

179 E5 20 A1 E3 AA A2 0A+

180 0D

181

182 =0141 helpMsg\_length equ $-helpMsg

183 0325 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 034A DA 22\*(C4) BF tableTop DB 218, Line1\_length-2 dup+

192 (196), 191

193 =0024 tableTop\_length equ $-tableTop

194 036E C0 22\*(C4) D9 tableBottom DB 192, Line1\_length-2 dup (196), +

195 217

196 =0024 tableBottom\_length equ $-tableBottom

197

198 ; сообщения

199 0392 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 03A5 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 03BB 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 03CF 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 03ED 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 03FE 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 041B 46 39 f1\_txt DB 'F9'

228 041D 46 31 f2\_txt DB 'F1'

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 5

tsr.asm

229 041F 46 32 f3\_txt DB 'F2'

230 0421 46 33 f4\_txt DB 'F3'

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

232

233 0423 changeFx proc

234 0423 50 push AX

235 0424 53 push BX

236 0425 51 push CX

237 0426 52 push DX

238 0427 55 push BP

239 0428 06 push ES

240 0429 33 DB xor BX, BX

241

242 042B B4 03 mov AH, 03h

243 042D CD 10 int 10h

244 042F 52 push DX

245

246 0430 0E push CS

247 0431 07 pop ES

248

249 0432 \_checkF1:

250 0432 BD 041Br lea BP, f1\_txt

251 0435 B9 0002 mov CX, fx\_length

252 0438 B7 00 mov BH, 0

253 043A B6 00 mov DH, 0

254 043C B2 4E mov DL, 78

255 043E B8 1301 mov AX, 1301h

256

257 0441 80 3E 0143r FF cmp signaturePrintingEnabled, true

258 0446 74 07 je \_greenF1

259

260 0448 \_redF1:

261 0448 B3 4F mov BL, 01001111b ; red

262 044A CD 10 int 10h

263 044C EB 08 90 jmp \_checkF2

264

265 044F \_greenF1:

266 044F BD 041Br lea BP, f1\_txt

267 0452 B3 2F mov BL, 00101111b ; green

268 0454 CD 10 int 10h

269

270 0456 \_checkF2:

271 0456 BD 041Dr lea BP, f2\_txt

272 0459 B9 0002 mov CX, fx\_length

273 045C B7 00 mov BH, 0

274 045E B6 01 mov DH, 1

275 0460 B2 4E mov DL, 78

276 0462 B8 1301 mov AX, 1301h

277

278 0465 80 3E 0144r FF cmp cursiveEnabled, true

279 046A 74 07 je \_greenF2

280

281 046C \_redF2:

282 046C B3 4F mov BL, 01001111b ; red

283 046E CD 10 int 10h

284 0470 EB 05 90 jmp \_checkF3

285

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 6

tsr.asm

286 0473 \_greenF2:

287 0473 B3 2F mov BL, 00101111b ; green

288 0475 CD 10 int 10h

289

290 0477 \_checkF3:

291 0477 BD 041Fr lea BP, f3\_txt

292 047A B9 0002 mov CX, fx\_length

293 047D B7 00 mov BH, 0

294 047F B6 02 mov DH, 2

295 0481 B2 4E mov DL, 78

296 0483 B8 1301 mov AX, 1301h

297

298 0486 80 3E 0142r FF cmp translateEnabled, true

299 048B 74 07 je \_greenF3

300

301 048D \_redF3:

302 048D B3 4F mov BL, 01001111b ; red

303 048F CD 10 int 10h

304 0491 EB 05 90 jmp \_checkF4

305

306 0494 \_greenF3:

307 0494 B3 2F mov BL, 00101111b ; green

308 0496 CD 10 int 10h

309

310 0498 \_checkF4:

311 0498 BD 0421r lea BP, f4\_txt

312 049B B9 0002 mov CX, fx\_length

313 049E B7 00 mov BH, 0

314 04A0 B6 03 mov DH, 3

315 04A2 B2 4E mov DL, 78

316 04A4 B8 1301 mov AX, 1301h

317

318 04A7 80 3E 0137r FF cmp ignoreEnabled, true

319 04AC 74 07 je \_greenF4

320

321 04AE \_redF4:

322 04AE B3 4F mov BL, 01001111b ; red

323 04B0 CD 10 int 10h

324 04B2 EB 05 90 jmp \_outFx

325

326 04B5 \_greenF4:

327 04B5 B3 2F mov BL, 00101111b ; green

328 04B7 CD 10 int 10h

329

330 04B9 \_outFx:

331 04B9 5A pop DX

332 04BA B4 02 mov AH, 02h

333 04BC CD 10 int 10h

334

335 04BE 07 pop ES

336 04BF 5D pop BP

337 04C0 5A pop DX

338 04C1 59 pop CX

339 04C2 5B pop BX

340 04C3 58 pop AX

341 04C4 C3 ret

342 04C5 changeFx endp

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 7

tsr.asm

343

344 ;новый обработчик

345 04C5 new\_int9h proc far

346 ; сохраняем значения всех, изменяемых регистров в стэке

347 04C5 56 push SI

348 04C6 50 push AX

349 04C7 53 push BX

350 04C8 51 push CX

351 04C9 52 push DX

352 04CA 06 push ES

353 04CB 1E push DS

354 04CC 55 push BP

355 ; синхронизируем CS и DS

356 04CD 0E push CS

357 04CE 1F pop DS

358

359 04CF B8 0040 mov AX, 40h ; 40h-сегмент,где хранятся флаги сост-я клавиатуры, кольц. +

360 буфер ввода

361 04D2 8E C0 mov ES, AX

362 04D4 E4 60 in AL, 60h ; записываем в AL скан-код нажатой клавиши

363

364 ;@ проверка на /U, только для ИУ5-43

365 04D6 3C 16 cmp AL, 22 ; была нажата клавиша U?

366 04D8 75 24 jne \_test\_Fx

367 04DA 26: 8A 26 0017 mov AH, ES:[17h] ; флаги клавиатуры

368 04DF 80 E4 0F and AH, 00001111b

369 04E2 80 FC 04 cmp AH, 00000100b ; был ли нажат ctrl?

370 04E5 75 17 jne \_test\_Fx ;!!!!!

371 ; выгрузка по CTRL+U

372 04E7 B4 FF mov AH, 0FFh

373 04E9 B0 01 mov AL, 01h

374 04EB CD 2F int 2Fh

375 ; завершаем обработку нажатия

376

377 04ED E4 61 in AL, 61h ;контроллер состояния клавиатуры

378 04EF 0C 80 or AL, 10000000b ;пометим, что клавишу нажали

379 04F1 E6 61 out 61h, AL

380 04F3 24 7F and AL, 01111111b ;пометим, что клавишу отпустили

381 04F5 E6 61 out 61h, AL

382 04F7 B0 20 mov AL, 20h

383 04F9 E6 20 out 20h, AL ;отправим в контроллер прерываний признак конца +

384 прерывания

385

386 ; выходим

387 04FB E9 009D jmp \_quit

388

389 ;@ далее - код для всех вариантов

390

391 ;проверка F1-F4

392 04FE \_test\_Fx:

393 04FE 2C 3A sub AL, 58 ; в AL теперь номер функциональной клавиши

394 0500 \_F9:

395 0500 3C 09 cmp AL, 9 ; F9

396 0502 75 0A jne \_F1

397 0504 F6 16 0143r not signaturePrintingEnabled

398 0508 E8 FF18 call changeFx

399 050B EB 2E 90 jmp \_translate\_or\_ignore

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 8

tsr.asm

400 050E \_F1:

401 050E 3C 01 cmp AL, 1 ; F2

402 0510 75 0D jne \_F2

403 0512 F6 16 0144r not cursiveEnabled

404 0516 E8 FF0A call changeFx

405 0519 E8 01F1 call setCursive ; перевод символа в курсив и обратно в зависимости от +

406 флага cursiveEnabled

407 051C EB 1D 90 jmp \_translate\_or\_ignore

408 051F \_F2:

409 051F 3C 02 cmp AL, 2 ; F2

410 0521 75 0A jne \_F3

411 0523 F6 16 0142r not translateEnabled

412 0527 E8 FEF9 call changeFx

413 052A EB 0F 90 jmp \_translate\_or\_ignore

414 052D \_F3:

415 052D 3C 03 cmp AL, 3 ; F3

416 052F 75 0A jne \_translate\_or\_ignore

417 0531 F6 16 0137r not ignoreEnabled

418 0535 E8 FEEB call changeFx

419 0538 EB 01 90 jmp \_translate\_or\_ignore

420

421 ;игнорирование и перевод

422 053B \_translate\_or\_ignore:

423

424 053B 9C pushf

425 053C 2E: FF 1E 0166r call dword ptr CS:[old\_int9hOffset] ; вызываем стандартный обработчик прерывания

426 0541 B8 0040 mov AX, 40h ; 40h-сегмент,где хранятся флаги сост-я клавы,кольц. +

427 буфер ввода

428 0544 8E C0 mov ES, AX

429 0546 26: 8B 1E 001C mov BX, ES:[1Ch] ; адрес хвоста

430 054B 4B dec BX ; сместимся назад к последнему

431 054C 4B dec BX ; введённому символу

432 054D 83 FB 1E cmp BX, 1Eh ; не вышли ли мы за пределы буфера?

433 0550 73 03 jae \_go

434 0552 BB 003C mov BX, 3Ch ; хвост вышел за пределы буфера, значит последний введённый +

435 символ

436 ; находится в конце буфера

437

438 0555 \_go:

439 0555 26: 8B 17 mov DX, ES:[BX] ; в DX 0 введённый символ

440 ;включен ли режим блокировки ввода?

441 0558 80 3E 0137r FF cmp ignoreEnabled, true

442 055D 75 1A jne \_check\_translate

443

444 ; да, включен

445 055F BE 0000 mov SI, 0

446 0562 B9 0034 mov CX, ignoredLength ;кол-во игнорируемых символов

447

448 ; проверяем, присутствует ли текущий символ в списке игнорируемых

449 0565 \_check\_ignored:

450 0565 3A 94 0103r cmp DL,ignoredChars[SI]

451 0569 74 06 je \_block

452 056B 46 inc SI

453 056C E2 F7 loop \_check\_ignored

454 056E EB 09 90 jmp \_check\_translate

455

456 ; блокируем

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 9

tsr.asm

457 0571 \_block:

458 0571 26: 89 1E 001C mov ES:[1Ch], BX ;блокировка ввода символа

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

460 ;@ а заменять одни символы другими,

461 ;@ замените строку выше строкой

462 ;@ mov ES:[BX], AX

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

464 на звёздочки

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

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

467 ;@ и раскомментировать строки ниже:

468 ;@ xor AX, AX

469 ;@ mov AL, replaceWith[SI]

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

471 0576 EB 23 90 jmp \_quit

472

473 0579 \_check\_translate:

474 ; включен ли режим перевода?

475 0579 80 3E 0142r FF cmp translateEnabled, true

476 057E 75 1B jne \_quit

477

478 ; да, включен

479 0580 BE 0000 mov SI, 0

480 0583 B9 0005 mov CX, translateLength ; кол-во символов для перевода

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

482 0586 \_check\_translate\_loop:

483 0586 3A 94 0138r cmp DL, translateFrom[SI]

484 058A 74 06 je \_translate

485 058C 46 inc SI

486 058D E2 F7 loop \_check\_translate\_loop

487 058F EB 0A 90 jmp \_quit

488

489 ; переводим

490 0592 \_translate:

491 0592 33 C0 xor AX, AX

492 0594 8A 84 013Dr mov AL, translateTo[SI]

493 0598 26: 89 07 mov ES:[BX], AX ; замена символа

494

495 059B \_quit:

496 ; восстанавливаем все регистры

497 059B 5D pop BP

498 059C 1F pop DS

499 059D 07 pop ES

500 059E 5A pop DX

501 059F 59 pop CX

502 05A0 5B pop BX

503 05A1 58 pop AX

504 05A2 5E pop SI

505 05A3 CF iret

506 05A4 new\_int9h endp

507

508 ;=== Обработчик прерывания int 1Ch ===;

509 ;=== Вызывается каждые 55 мс ===;

510 05A4 new\_int1Ch proc far

511 05A4 50 push AX

512 05A5 0E push CS

513 05A6 1F pop DS

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 10

tsr.asm

514

515 05A7 9C pushf

516 05A8 2E: FF 1E 016Ar call dword ptr CS:[old\_int1ChOffset]

517

518 05AD 80 3E 0143r FF cmp signaturePrintingEnabled, true ; если нажата управляющая клавиша (в данном случае +

519 F1)

520 05B2 75 1D jne \_notToPrint

521

522 05B4 81 3E 0174r 0080 cmp counter, printDelay\*1000/55 + 1 ; если кол-во "тактов" эквивалентно +

523 %printDelay% секундам

524 05BA 74 03 je \_letsPrint

525

526 05BC EB 0E 90 jmp \_dontPrint

527

528 05BF \_letsPrint:

529 05BF F6 16 0143r not signaturePrintingEnabled

530 05C3 C7 06 0174r 0000 mov counter, 0

531 05C9 E8 0094 call printSignature

532

533 05CC \_dontPrint:

534 05CC 83 06 0174r 01 add counter, 1

535

536 05D1 \_notToPrint:

537

538 05D1 58 pop AX

539

540 05D2 CF iret

541 05D3 new\_int1Ch endp

542

543 ;=== Обработчик прерывания int 2Fh ===;

544 ;=== Служит для:

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

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

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

548 ;===

549 05D3 new\_int2Fh proc

550 05D3 80 FC FF cmp AH, 0FFh ;наша функция?

551 05D6 75 0B jne \_2Fh\_std ;нет - на старый обработчик

552 05D8 3C 00 cmp AL, 0 ;подфункция проверки, загружен ли резидент в память?

553 05DA 74 0C je \_already\_installed

554 05DC 3C 01 cmp AL, 1 ;подфункция выгрузки из памяти?

555 05DE 74 0B je \_uninstall

556 05E0 EB 01 90 jmp \_2Fh\_std ;нет - на старый обработчик

557

558 05E3 \_2Fh\_std:

559 05E3 2E: FF 2E 016Er jmp dword ptr CS:[old\_int2FhOffset] ;вызов старого обработчика

560

561 05E8 \_already\_installed:

562 05E8 B4 69 mov AH, 'i' ;вернём 'i', если резидент загружен в память

563 05EA CF iret

564

565 05EB \_uninstall:

566 05EB 1E push DS

567 05EC 06 push ES

568 05ED 52 push DX

569 05EE 53 push BX

570

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 11

tsr.asm

571 05EF 33 DB xor BX, BX

572

573 ; CS = ES, для доступа к переменным

574 05F1 0E push CS

575 05F2 07 pop ES

576

577 05F3 B8 2509 mov AX, 2509h

578 05F6 26: 8B 16 0166r mov DX, ES:old\_int9hOffset ; возвращаем вектор прерывания

579 05FB 26: 8E 1E 0168r mov DS, ES:old\_int9hSegment ; на место

580 0600 CD 21 int 21h

581

582 0602 B8 251C mov AX, 251Ch

583 0605 26: 8B 16 016Ar mov DX, ES:old\_int1ChOffset ; возвращаем вектор прерывания

584 060A 26: 8E 1E 016Cr mov DS, ES:old\_int1ChSegment ; на место

585 060F CD 21 int 21h

586

587 0611 B8 252F mov AX, 252Fh

588 0614 26: 8B 16 016Er mov DX, ES:old\_int2FhOffset ; возвращаем вектор прерывания

589 0619 26: 8E 1E 0170r mov DS, ES:old\_int2FhSegment ; на место

590 061E CD 21 int 21h

591

592 0620 2E: 8E 06 002C mov ES, CS:2Ch ; загрузим в ES адрес окружения

593 0625 B4 49 mov AH, 49h ; выгрузим из памяти окружение

594 0627 CD 21 int 21h

595 0629 72 0B jc \_notRemove

596

597 062B 0E push CS

598 062C 07 pop ES ;в ES - адрес резидентной программы

599 062D B4 49 mov AH, 49h ;выгрузим из памяти резидент

600 062F CD 21 int 21h

601 0631 72 03 jc \_notRemove

602 0633 EB 15 90 jmp \_unloaded

603

604 0636 \_notRemove: ; не удалось выполнить выгрузку

605 ; вывод сообщения о неудачной выгрузке

606 0636 B4 03 mov AH, 03h ; получаем позицию курсора

607 0638 CD 10 int 10h

608 063A BD 03FEr lea BP, noRemoveMsg

609 063D B9 001D mov CX, noRemoveMsg\_length

610 0640 B3 07 mov BL, 0111b

611 0642 B8 1301 mov AX, 1301h

612 0645 CD 10 int 10h

613 0647 EB 12 90 jmp \_2Fh\_exit

614

615 064A \_unloaded: ; выгрузка прошла успешно

616 ; вывод сообщения об удачной выгрузке

617 064A B4 03 mov AH, 03h ; получаем позицию курсора

618 064C CD 10 int 10h

619 064E BD 03EDr lea BP, removedMsg

620 0651 B9 0011 mov CX, removedMsg\_length

621 0654 B3 07 mov BL, 0111b

622 0656 B8 1301 mov AX, 1301h

623 0659 CD 10 int 10h

624

625 065B \_2Fh\_exit:

626 065B 5B pop BX

627 065C 5A pop DX

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 12

tsr.asm

628 065D 07 pop ES

629 065E 1F pop DS

630 065F CF iret

631 0660 new\_int2Fh endp

632

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

634 ;=== Настраивается значениями переменных в начале исходника

635 ;===

636 0660 printSignature proc

637 0660 50 push AX

638 0661 52 push DX

639 0662 51 push CX

640 0663 53 push BX

641 0664 06 push ES

642 0665 54 push SP

643 0666 55 push BP

644 0667 56 push SI

645 0668 57 push DI

646

647 0669 33 C0 xor AX, AX

648 066B 33 DB xor BX, BX

649 066D 33 D2 xor DX, DX

650

651 066F B4 03 mov AH, 03h ;чтение текущей позиции курсора

652 0671 CD 10 int 10h

653 0673 52 push DX ;помещаем информацию о +

654 положении курсора в стек

655

656 0674 83 3E 0176r 00 cmp printPos, 0

657 0679 74 0E je \_printTop

658

659 067B 83 3E 0176r 01 cmp printPos, 1

660 0680 74 0E je \_printCenter

661

662 0682 83 3E 0176r 02 cmp printPos, 2

663 0687 74 0E je \_printBottom

664

665 ;все числа подобраны на глаз...

666 0689 \_printTop:

667 0689 B6 00 mov DH, 0

668 068B B2 0F mov DL, 15

669 068D EB 0F 90 jmp \_actualPrint

670

671 0690 \_printCenter:

672 0690 B6 09 mov DH, 9

673 0692 B2 0F mov DL, 15

674 0694 EB 08 90 jmp \_actualPrint

675

676 0697 \_printBottom:

677 0697 B6 13 mov DH, 19

678 0699 B2 0F mov DL, 15

679 069B EB 01 90 jmp \_actualPrint

680

681 069E \_actualPrint:

682 069E B4 0F mov AH, 0Fh ;чтение текущего видеорежима. в+

683 BH - текущая страница

684 06A0 CD 10 int 10h

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 13

tsr.asm

685

686 06A2 0E push CS

687 06A3 07 pop ES ;указываем ES на CS

688

689 ;вывод 'верхушки' таблицы

690 06A4 52 push DX

691 06A5 BD 034Ar lea BP, tableTop ;помещаем в BP указатель на +

692 выводимую строку

693 06A8 B9 0024 mov CX, tableTop\_length ;в CX - длина строки

694 06AB B3 07 mov BL, 0111b ;цвет выводимого текста ref: +

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

696 06AD B8 1301 mov AX, 1301h ;AH=13h - номер ф-ии, AL=01h - +

697 курсор перемещается при выводе каждого из символов строки

698 06B0 CD 10 int 10h

699 06B2 5A pop DX

700 06B3 FE C6 inc DH

701

702

703 ;вывод первой линии

704 06B5 52 push DX

705 06B6 BD 0178r lea BP, signatureLine1

706 06B9 B9 0024 mov CX, Line1\_length

707 06BC B3 07 mov BL, 0111b

708 06BE B8 1301 mov AX, 1301h

709 06C1 CD 10 int 10h

710 06C3 5A pop DX

711 06C4 FE C6 inc DH

712

713 ;вывод второй линии

714 06C6 52 push DX

715 06C7 BD 019Cr lea BP, signatureLine2

716 06CA B9 0024 mov CX, Line2\_length

717 06CD B3 07 mov BL, 0111b

718 06CF B8 1301 mov AX, 1301h

719 06D2 CD 10 int 10h

720 06D4 5A pop DX

721 06D5 FE C6 inc DH

722

723 ;вывод третьей линии

724 06D7 52 push DX

725 06D8 BD 01C0r lea BP, signatureLine3

726 06DB B9 0024 mov CX, Line3\_length

727 06DE B3 07 mov BL, 0111b

728 06E0 B8 1301 mov AX, 1301h

729 06E3 CD 10 int 10h

730 06E5 5A pop DX

731 06E6 FE C6 inc DH

732

733 ;вывод 'низа' таблицы

734 06E8 52 push DX

735 06E9 BD 036Er lea BP, tableBottom

736 06EC B9 0024 mov CX, tableBottom\_length

737 06EF B3 07 mov BL, 0111b

738 06F1 B8 1301 mov AX, 1301h

739 06F4 CD 10 int 10h

740 06F6 5A pop DX

741 06F7 FE C6 inc DH

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 14

tsr.asm

742

743 06F9 33 DB xor BX, BX

744 06FB 5A pop DX ;восстанавливаем из стека +

745 прежнее положение курсора

746 06FC B4 02 mov AH, 02h ;меняем положение курсора на +

747 первоначальное

748 06FE CD 10 int 10h

749 0700 E8 FD20 call changeFx

750

751 0703 5F pop DI

752 0704 5E pop SI

753 0705 5D pop BP

754 0706 5C pop SP

755 0707 07 pop ES

756 0708 5B pop BX

757 0709 59 pop CX

758 070A 5A pop DX

759 070B 58 pop AX

760

761 070C C3 ret

762 070D printSignature endp

763

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

765 на обычное и наоброт

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

767 070D setCursive proc

768 070D 06 push ES ; сохраняем регистры

769 070E 50 push AX

770 070F 0E push CS

771 0710 07 pop ES

772

773 0711 80 3E 0144r FF cmp cursiveEnabled, true

774 0716 75 30 jne \_restoreSymbol

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

776 ; предварительно сохраняя старый символ в savedSymbol

777

778 0718 E8 004C call saveFont

779 071B 8A 0E 0155r mov CL, charToCursiveIndex

780 071F \_shifTtable:

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

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

783 071F 83 C5 10 add BP, 16

784 0722 E2 FB loop \_shiftTable

785

786 ; пpи savefont смещается pегистp ES

787 ; поэтомy пpиходится делать такие махинации, чтобы

788 ; записать полyченный элемент в savedSymbol

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

790 0724 1E push DS

791 0725 58 pop AX

792 0726 06 push ES

793 0727 1F pop DS

794 0728 50 push AX

795 0729 07 pop ES

796 072A 50 push AX

797

798 072B 8B F5 mov SI, BP

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 15

tsr.asm

799 072D BF 0156r lea DI, savedSymbol

800 ; сохpаняем в пеpеменнyю savedSymbol

801 ; таблицy нyжного символа

802 0730 B9 0010 mov CX, 16

803 ; movsb из DS:SI в ES:DI

804 0733 F3> A4 rep movsb

805 ; исходные позиции сегментов возвpащены

806 0735 1F pop DS ; восстановление DS

807

808 ; заменим написание символа на кypсив

809 0736 B9 0001 mov CX, 1

810 0739 B6 00 mov DH, 0

811 073B 8A 16 0155r mov DL, charToCursiveIndex

812 073F BD 0145r lea BP, cursiveSymbol

813 0742 E8 0015 call changeFont

814 0745 EB 10 90 jmp \_exitSetCursive

815

816 0748 \_restoreSymbol:

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

818

819 0748 B9 0001 mov CX, 1

820 074B B6 00 mov DH, 0

821 074D 8A 16 0155r mov DL, charToCursiveIndex

822 0751 BD 0156r lea bp, savedSymbol

823 0754 E8 0003 call changeFont

824

825 0757 \_exitSetCursive:

826 0757 58 pop AX

827 0758 07 pop ES

828 0759 C3 ret

829 075A setCursive endp

830

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

832 ;===

833 ; \*\*\* входные данные

834 ; DL = номер символа для замены

835 ; CX = Кол-во символов заменяемых изображений символов

836 ; (начиная с символа указанного в DX)

837 ; ES:bp = адрес таблицы

838 ;

839 ; \*\*\* описание работы процедуры

840 ; Происходит вызов int 10h (видеосервис)

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

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

843 ; символа для текущего шрифта

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

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

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

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

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

849 ;

850 ; \*\*\* результат

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

852 ; на предложенное пользователем.

853 ; Изменению подвергнутся все символы, находящиеся на экране,

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

855

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 16

tsr.asm

856 075A changeFont proc

857 075A 50 push AX

858 075B 53 push BX

859 075C B8 1100 mov AX, 1100h

860 075F BB 1000 mov BX, 1000h

861 0762 CD 10 int 10h

862 0764 58 pop AX

863 0765 5B pop BX

864 0766 C3 ret

865 0767 changeFont endp

866

867 ;=== Функция сохранения нормального начертания символа

868 ;===

869 ; \*\*\* входные данные

870 ; BH - тип возвращаемой символьной таблицы

871 ; 0 - таблица из int 1fh

872 ; 1 - таблица из int 44h

873 ; 2-5 - таблица из 8x14, 8x8, 8x8 (top), 9x14

874 ; 6 - 8x16

875 ;

876 ; \*\*\* описание работы процедуры

877 ; Происходит вызов int 10h (видеосервис)

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

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

880 ;

881 ; \*\*\* результат

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

883 ; в CX находится байт на символ

884 ; в DL количество экранных строк

885 ; ВАЖНО! Происходит сдвиг регистра ES

886 ; ( ES становится равным C000h )

887

888 0767 saveFont proc

889 0767 50 push AX

890 0768 53 push BX

891 0769 B8 1130 mov AX, 1130h

892 076C BB 0600 mov BX, 0600h

893 076F CD 10 int 10h

894 0771 58 pop AX

895 0772 5B pop BX

896 0773 C3 ret

897 0774 saveFont endp

898

899

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

901 ;===

902 0774 \_initTSR: ; старт резидента

903 0774 B4 03 mov AH, 03h

904 0776 CD 10 int 10h

905 0778 52 push DX

906 0779 B4 00 mov AH,00h ; установка видеорежима (83h текст +

907 80x25 16/8 CGA,EGA b800 Comp,RGB,Enhanced), без очистки экрана

908 077B B0 83 mov AL,83h

909 077D CD 10 int 10h

910 077F 5A pop DX

911 0780 B4 02 mov AH, 02h

912 0782 CD 10 int 10h

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 17

tsr.asm

913

914

915 0784 E8 00B3 call commandParamsParser

916 0787 B8 3509 mov AX,3509h ; получить в ES:BX вектор 09

917 078A CD 21 int 21h ; прерывания

918

919 ;@ === Удаление резидента из памяти ===

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

921 ;@ нужно закомментировать следующие 3 строки, а также

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

923 078C 80 3E 0172r FF cmp unloadTSR, true

924 0791 74 03 je \_removingOnParameter

925 0793 EB 15 90 jmp \_notRemovingNow

926

927 0796 \_removingOnParameter:

928 0796 B4 FF mov AH, 0FFh

929 0798 B0 00 mov AL, 0

930 079A CD 2F int 2Fh

931 079C 80 FC 69 cmp AH, 'i' ; проверка того, загружена ли уже программа

932 079F 74 7D je \_remove

933 07A1 B4 09 mov AH, 09h ;@ для выгрузки резидента по повторному+

934 запуску закомментировать эту строку

935 07A3 BA 03CFr lea DX, notInstalledMsg ;@ для выгрузки резидента по повторному запуску +

936 закомментировать эту строку

937 07A6 CD 21 int 21h ;@ для выгрузки резидента по повторному+

938 запуску закомментировать эту строку

939 07A8 CD 20 int 20h ;@ для выгрузки резидента по повторному+

940 запуску закомментировать эту строку

941

942 07AA \_notRemovingNow:

943

944 07AA 80 3E 0173r FF cmp notLoadTSR, true ; если была выведена справка

945 07AF 74 0E je \_exit\_tmp ; просто выходим

946

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

948 комментируем 5 строк ниже

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

950 07B1 B4 FF mov AH, 0FFh

951 07B3 B0 00 mov AL, 0

952 07B5 CD 2F int 2Fh

953 07B7 80 FC 69 cmp AH, 'i' ; проверка того, загружена ли уже программа

954 07BA 74 6B je \_alreadyInstalled

955

956 07BC EB 04 90 jmp \_tmp

957

958 07BF \_exit\_tmp:

959 07BF EB 77 90 jmp \_exit

960

961 07C2 \_tmp:

962 07C2 06 push ES

963 07C3 A1 002C mov AX, DS:[2Ch] ; psp

964 07C6 8E C0 mov ES, AX

965 07C8 B4 49 mov AH, 49h ; хватит памяти чтоб остаться

966 07CA CD 21 int 21h ; резидентом?

967 07CC 07 pop ES

968 07CD 72 62 jc \_notMem ; не хватило - выходим

969

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 18

tsr.asm

970 ;== int 09h ==;

971

972 07CF 2E: 89 1E 0166r mov word ptr CS:old\_int9hOffset, BX

973 07D4 2E: 8C 06 0168r mov word ptr CS:old\_int9hSegment, ES

974 07D9 B8 2509 mov AX, 2509h ; установим вектор на 09

975 07DC BA 04C5r mov DX, offset new\_int9h ; прерывание

976 07DF CD 21 int 21h

977

978 ;== int 1Ch ==;

979 07E1 B8 351C mov AX,351Ch ; получить в ES:BX вектор 1C

980 07E4 CD 21 int 21h ; прерывания

981 07E6 2E: 89 1E 016Ar mov word ptr CS:old\_int1ChOffset, BX

982 07EB 2E: 8C 06 016Cr mov word ptr CS:old\_int1ChSegment, ES

983 07F0 B8 251C mov AX, 251Ch ; установим вектор на 1C

984 07F3 BA 05A4r mov DX, offset new\_int1Ch ; прерывание

985 07F6 CD 21 int 21h

986

987 ;== int 2Fh ==;

988 07F8 B8 352F mov AX,352Fh ; получить в ES:BX вектор 1C

989 07FB CD 21 int 21h ; прерывания

990 07FD 2E: 89 1E 016Er mov word ptr CS:old\_int2FhOffset, BX

991 0802 2E: 8C 06 0170r mov word ptr CS:old\_int2FhSegment, ES

992 0807 B8 252F mov AX, 252Fh ; установим вектор на 2F

993 080A BA 05D3r mov DX, offset new\_int2Fh ; прерывание

994 080D CD 21 int 21h

995

996 080F E8 FC11 call changeFx

997 0812 BA 0392r mov DX, offset installedMsg ; выводим что все ок

998 0815 B4 09 mov AH, 9

999 0817 CD 21 int 21h

1000 0819 BA 0774r mov DX, offset \_initTSR ; остаемся в памяти резидентом

1001 081C CD 27 int 27h ; и выходим

1002 ; конец основной программы

1003 081E \_remove: ; выгрузка программы из памяти

1004 081E B4 FF mov AH, 0FFh

1005 0820 B0 01 mov AL, 1

1006 0822 CD 2F int 2Fh

1007 0824 EB 12 90 jmp \_exit

1008 0827 \_alreadyInstalled:

1009 0827 B4 09 mov AH, 09h

1010 0829 BA 03A5r lea DX, alreadyInstalledMsg

1011 082C CD 21 int 21h

1012 082E EB 08 90 jmp \_exit

1013 0831 \_notMem: ; не хватает памяти, чтобы остаться резидентом

1014 0831 BA 03BBr mov DX, offset noMemMsg

1015 0834 B4 09 mov AH, 9

1016 0836 CD 21 int 21h

1017 0838 \_exit: ; выход

1018 0838 CD 20 int 20h

1019

1020 ;=== Процедура проверки параметров ком. строки ===;

1021 ;===

1022 083A commandParamsParser proc

1023 083A 0E push CS

1024 083B 07 pop ES

1025 083C C6 06 0172r 00 mov unloadTSR, 0

1026 0841 C6 06 0173r 00 mov notLoadTSR, 0

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 19

tsr.asm

1027

1028 0846 BE 0080 mov SI, 80h ;SI=смещение командной строки.

1029 0849 AC lodsb ;Получим кол-во символов.

1030 084A 0A C0 or AL, AL ;Если 0 символов введено,

1031 084C 74 4A jz \_exitHelp ;то все в порядке.

1032

1033 084E \_nextChar:

1034

1035 084E 46 inc SI ;Теперь SI указывает на первый символ +

1036 строки.

1037

1038 084F 80 3C 0D cmp [SI], BYTE ptr 13

1039 0852 74 44 je \_exitHelp

1040

1041

1042 0854 AD lodsw ;Получаем два символа

1043 0855 3D 3F2F cmp AX, '?/' ;Это '/?' (данные расположены в +

1044 обратном порядк, т.е. AL:AH вместо AH:AL)

1045 0858 74 0D je \_question

1046 085A 3D 752F cmp AX, 'u/'

1047 085D 74 1F je \_finishTSR

1048 ;;; для большой "U"

1049 085F 3D 552F cmp AX, 'U/'

1050 0862 74 1A je \_finishTSR

1051 ;cmp AH, '/'

1052 ;je \_errorParam

1053

1054 0864 EB 32 90 jmp \_exitHelp

1055

1056 0867 \_question:

1057 ; вывод строки помощи

1058 0867 B4 03 mov AH,03

1059 0869 CD 10 int 10h

1060 086B BD 01E4r lea BP, helpMsg

1061 086E B9 0141 mov CX, helpMsg\_length

1062 0871 B3 07 mov BL, 0111b

1063 0873 B8 1301 mov AX, 1301h

1064 0876 CD 10 int 10h

1065 ; конец вывода строки помощи

1066 0878 F6 16 0173r not notLoadTSR ;флаг того, что необходимо не загружать резидент

1067 087C EB D0 jmp \_nextChar

1068

1069 ;@ === Удаление резидента из памяти ===

1070 ;@ Если по варианту необходимо выгружать резидент по параметру '/u' коммандной строки,

1071 ;@ нужно использовать следующий код, в остальных случаях необходимо закомменитровать

1072 ;@ этот код, кроме названия метки! (по желанию можно избавиться и от метки, но +

1073 аккуратно просмотреть использование)

1074 087E \_finishTSR:

1075 087E F6 16 0172r not unloadTSR ;флаг того, что необходимо выгузить резидент

1076 0882 EB CA jmp \_nextChar

1077

1078 0884 EB 12 90 jmp \_exitHelp

1079

1080 0887 \_errorParam:

1081 ;вывод строки

1082 0887 B4 03 mov AH,03

1083 0889 CD 10 int 10h

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 20

tsr.asm

1084 088B BD 0325r lea BP, CS:errorParamMsg

1085 088E B9 0025 mov CX, errorParamMsg\_length

1086 0891 B3 07 mov BL, 0111b

1087 0893 B8 1301 mov AX, 1301h

1088 0896 CD 10 int 10h

1089 ;конец вывода строки

1090 0898 \_exitHelp:

1091 0898 C3 ret

1092 0899 commandParamsParser endp

1093

1094 0899 code ends

1095 end \_start

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 21

Symbol Table

Symbol Name Type Value

??DATE Text "05/14/23"

??FILENAME Text "tsr "

??TIME Text "20:11:41"

??VERSION Number 030A

@CPU Text 0101H

@CURSEG Text CODE

@FILENAME Text TSR

@WORDSIZE Text 2

ALREADYINSTALLEDMSG Byte CODE:03A5

CHANGEFONT Near CODE:075A

CHANGEFX Near CODE:0423

CHARTOCURSIVEINDEX Byte CODE:0155

COMMANDPARAMSPARSER Near CODE:083A

COUNTER Word CODE:0174

CURSIVEENABLED Byte CODE:0144

CURSIVESYMBOL Byte CODE:0145

ERRORPARAMMSG Byte CODE:0325

ERRORPARAMMSG\_LENGTH Number 0025

F1\_TXT Byte CODE:041B

F2\_TXT Byte CODE:041D

F3\_TXT Byte CODE:041F

F4\_TXT Byte CODE:0421

FX\_LENGTH Number 0002

HELPMSG Byte CODE:01E4

HELPMSG\_LENGTH Number 0141

IGNOREDCHARS Byte CODE:0103

IGNOREDLENGTH Number 0034

IGNOREENABLED Byte CODE:0137

INSTALLEDMSG Byte CODE:0392

LINE1\_LENGTH Number 0024

LINE2\_LENGTH Number 0024

LINE3\_LENGTH Number 0024

NEW\_INT1CH Far CODE:05A4

NEW\_INT2FH Near CODE:05D3

NEW\_INT9H Far CODE:04C5

NOMEMMSG Byte CODE:03BB

NOREMOVEMSG Byte CODE:03FE

NOREMOVEMSG\_LENGTH Number 001D

NOTINSTALLEDMSG Byte CODE:03CF

NOTLOADTSR Byte CODE:0173

OLD\_INT1CHOFFSET Word CODE:016A

OLD\_INT1CHSEGMENT Word CODE:016C

OLD\_INT2FHOFFSET Word CODE:016E

OLD\_INT2FHSEGMENT Word CODE:0170

OLD\_INT9HOFFSET Word CODE:0166

OLD\_INT9HSEGMENT Word CODE:0168

PRINTDELAY Number 0007

PRINTPOS Word CODE:0176

PRINTSIGNATURE Near CODE:0660

REMOVEDMSG Byte CODE:03ED

REMOVEDMSG\_LENGTH Number 0011

SAVEDSYMBOL Byte CODE:0156

SAVEFONT Near CODE:0767

SETCURSIVE Near CODE:070D

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 22

Symbol Table

SIGNATURELINE1 Byte CODE:0178

SIGNATURELINE2 Byte CODE:019C

SIGNATURELINE3 Byte CODE:01C0

SIGNATUREPRINTINGENABLED Byte CODE:0143

TABLEBOTTOM Byte CODE:036E

TABLEBOTTOM\_LENGTH Number 0024

TABLETOP Byte CODE:034A

TABLETOP\_LENGTH Number 0024

TRANSLATEENABLED Byte CODE:0142

TRANSLATEFROM Byte CODE:0138

TRANSLATELENGTH Number 0005

TRANSLATETO Byte CODE:013D

TRUE Number 00FF

UNLOADTSR Byte CODE:0172

\_2FH\_EXIT Near CODE:065B

\_2FH\_STD Near CODE:05E3

\_ACTUALPRINT Near CODE:069E

\_ALREADYINSTALLED Near CODE:0827

\_ALREADY\_INSTALLED Near CODE:05E8

\_BLOCK Near CODE:0571

\_CHECKF1 Near CODE:0432

\_CHECKF2 Near CODE:0456

\_CHECKF3 Near CODE:0477

\_CHECKF4 Near CODE:0498

\_CHECK\_IGNORED Near CODE:0565

\_CHECK\_TRANSLATE Near CODE:0579

\_CHECK\_TRANSLATE\_LOOP Near CODE:0586

\_DONTPRINT Near CODE:05CC

\_ERRORPARAM Near CODE:0887

\_EXIT Near CODE:0838

\_EXITHELP Near CODE:0898

\_EXITSETCURSIVE Near CODE:0757

\_EXIT\_TMP Near CODE:07BF

\_F1 Near CODE:050E

\_F2 Near CODE:051F

\_F3 Near CODE:052D

\_F9 Near CODE:0500

\_FINISHTSR Near CODE:087E

\_GO Near CODE:0555

\_GREENF1 Near CODE:044F

\_GREENF2 Near CODE:0473

\_GREENF3 Near CODE:0494

\_GREENF4 Near CODE:04B5

\_INITTSR Near CODE:0774

\_LETSPRINT Near CODE:05BF

\_NEXTCHAR Near CODE:084E

\_NOTMEM Near CODE:0831

\_NOTREMOVE Near CODE:0636

\_NOTREMOVINGNOW Near CODE:07AA

\_NOTTOPRINT Near CODE:05D1

\_OUTFX Near CODE:04B9

\_PRINTBOTTOM Near CODE:0697

\_PRINTCENTER Near CODE:0690

\_PRINTTOP Near CODE:0689

\_QUESTION Near CODE:0867

\_QUIT Near CODE:059B

\_REDF1 Near CODE:0448

Turbo Assembler Version 3.1 05/14/23 20:11:41 Page 23

Symbol Table

\_REDF2 Near CODE:046C

\_REDF3 Near CODE:048D

\_REDF4 Near CODE:04AE

\_REMOVE Near CODE:081E

\_REMOVINGONPARAMETER Near CODE:0796

\_RESTORESYMBOL Near CODE:0748

\_SHIFTTABLE Near CODE:071F

\_START Near CODE:0100

\_TEST\_FX Near CODE:04FE

\_TMP Near CODE:07C2

\_TRANSLATE Near CODE:0592

\_TRANSLATE\_OR\_IGNORE Near CODE:053B

\_UNINSTALL Near CODE:05EB

\_UNLOADED Near CODE:064A

Groups & Segments Bit Size Align Combine Class

CODE 16 0899 Para none CODE

# **2. Файл unloader.lst**

Turbo Assembler Version 3.1 05/14/23 17:29:07 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 05/14/23 17:29:07 Page 2

Symbol Table

Symbol Name Type Value

??DATE Text "05/14/23"

??FILENAME Text "unloader"

??TIME Text "17:29:07"

??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