Московский государственный технический университет

им. Н.Э. Баумана

|  |  |
| --- | --- |
| УТВЕРЖДАЮ: |  |
| Большаков С.А. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г. |

Курсовая работа по курсу «Системное программирование»

Исходный текст программного продукта

(вид документа)

писчая бумага

(вид носителя)

21

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛИ: |  |
| студенты группы ИУ5-45б  Коновалов И. Н. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Большаков С.А. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г. |

Москва – 2023

Содержание

[1. Файл tsr.lst 3](#_Toc778704)

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

1. **Файл tsr2.lst**

Turbo Assembler Version 3.1 05/19/23 14:03:31 Page 1

tsr2.asm

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

2 ; tsr.asm

3 ;

4 ; Сборка:

5 ; tasm.exe /l tsr.asm

6 ; tlink /t /x tsr.obj

7 ;

8 ; Авторы:

9 ; МГТУ им. Н. Э. Баумана, ИУ5-45Б, 2023 г

10 ; Коновалов И.Н.

11 ;

12 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

13

14 0000 code segment 'code'

15 assume CS:code, DS:code

16 org 100h

17 0100 \_start:

18

19 0100 E9 0680 jmp \_initTSR ; на начало программы

20 ;@Если по варианту нужно, чтобы происходила замена символов

21 ; данные

22 0103 A0 A1 A2 A3 A4 A5 F1+ ignoredChars DB +

23 A6 A7 A8 A9 AA AB AC+ 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя' ; список игнорируемых символов

24 AD AE AF E0 E1 E2 E3+

25 E4 E5 E6 E7 E8 E9 EA+

26 EB EC ED EE EF

27 0124 80 81 82 83 84 85 F0+ replaceWith DB +

28 86 87 88 89 8A 8B 8C+ 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЬЭЮЯ' ; список заменяемых символов

29 8D 8E 8F 90 91 92 93+

30 94 95 96 97 98 99 9A+

31 9C 9D 9E 9F

32 =0041 ignoredLength equ $-ignoredChars +

33 ; длина строки ignoredChars

34 0144 00 ignoreEnabled DB 0 +

35 ; флаг функции игнорирования ввода

36 0145 4B 56 59 4A 47 translateFrom DB 'KVYJG' +

37 ; символы для замены

38 014A 8B 8C 8D 8E 8F translateTo DB 'ЛМНОП' +

39 ; символы на которые будет идти замена

40 =0005 translateLength equ $-translateTo +

41 ; длина строки trasnlateFrom

42 014F 00 translateEnabled DB 0 +

43 ; флаг функции перевода

44

45 0150 00 signaturePrintingEnabled DB 0 +

46 ; флаг функции вывода информации об авторе

47 0151 00 cursiveEnabled DB 0 +

48 ; флаг перевода символа в курсив

49

50 0152 3F cursiveSymbol DB 00111111b +

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

52 0153 21 DB 00100001b

53 0154 22 DB 00100010b

54 0155 24 DB 00100100b

55 0156 78 DB 01111000b

56 0157 40 DB 01000000b

57 0158 40 DB 01000000b

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 2

tsr2.ASM

58 0159 70 DB 01110000b

59 015A 48 DB 01001000b

60 015B 48 DB 01001000b

61 015C 44 DB 01000100b

62 015D 44 DB 01000100b

63 015E 42 DB 01000010b

64 015F 82 DB 10000010b

65 0160 84 DB 10000100b

66 0161 F8 DB 11111000b

67

68 0162 82 charToCursiveIndex DB 'В' +

69 ;@ символ для замены

70 0163 10\*(FF) savedSymbol DB 16 dup(0FFh) +

71 ; переменная для хранения старого символа

72

73 =00FF true equ 0FFh +

74 ; константа истинности

75 0173 ???? old\_int9hOffset DW ? +

76 ; адрес старого обработчика int 9h

77 0175 ???? old\_int9hSegment DW ? +

78 ; сегмент старого обработчика int 9h

79 0177 ???? old\_int1ChOffset DW ? +

80 ; адрес старого обработчика int 1Ch

81 0179 ???? old\_int1ChSegment DW ? +

82 ; сегмент старого обработчика int 1Ch

83 017B ???? old\_int2FhOffset DW ? +

84 ; адрес старого обработчика int 2Fh

85 017D ???? old\_int2FhSegment DW ? +

86 ; сегмент старого обработчика int 2Fh

87

88 017F 00 notLoadTSR DB 0 +

89 ; 1 - не загружать

90 0180 0000 counter DW 0

91 =0002 printDelay equ 2 +

92 ;@ задержка перед выводом "подписи" в секундах

93

94

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

96 (1я строка).

97 0182 B3 8A AE A6 A8 A5 A2+ signatureLine1 DB 179, 'Кожиев Таймураз Казбекович +

98 20 92 A0 A9 AC E3 E0+ ', 179

99 A0 A7 20 8A A0 A7 A1+

100 A5 AA AE A2 A8 E7 20+

101 20 20 20 20 20 20 20+

102 20 20 20 20 20 20 20+

103 20 20 20 20 20 20 20+

104 20 20 B3

105 =0034 Line1\_length equ $-signatureLine1

106 01B6 B3 88 93 35 2D 34 35+ signatureLine2 DB 179, 'ИУ5-45 +

107 20 20 20 20 20 20 20+ ', 179

108 20 20 20 20 20 20 20+

109 20 20 20 20 20 20 20+

110 20 20 20 20 20 20 20+

111 20 20 20 20 20 20 20+

112 20 20 20 20 20 20 20+

113 20 20 B3

114 =0034 Line2\_length equ $-signatureLine2

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 3

tsr2.ASM

115 01EA B3 82 A0 E0 A8 A0 AD+ signatureLine3 DB 179, 'Вариант #7 +

116 E2 20 23 37 20 20 20+ ', 179

117 20 20 20 20 20 20 20+

118 20 20 20 20 20 20 20+

119 20 20 20 20 20 20 20+

120 20 20 20 20 20 20 20+

121 20 20 20 20 20 20 20+

122 20 20 B3

123 =0034 Line3\_length equ $-signatureLine3

124 021E 3E 74 73 72 2E 63 6F+ helpMsg DB '>tsr.com [/?]', 10, 13

125 6D 20 5B 2F 3F 5D 0A+

126 0D

127 022D 20 5B 2F 3F 5D 20 2D+ DB ' [/?] - вывод данной справки', 10, 13

128 20 A2 EB A2 AE A4 20+

129 A4 A0 AD AD AE A9 20+

130 E1 AF E0 A0 A2 AA A8+

131 0A 0D

132 024B 20 20 46 33 20 20 2D+ DB ' F3 - вывод ФИО и группы по таймеру внизу экрана', 10, 13

133 20 A2 EB A2 AE A4 20+

134 94 88 8E 20 A8 20 A3+

135 E0 E3 AF AF EB 20 AF+

136 AE 20 E2 A0 A9 AC A5+

137 E0 E3 20 A2 AD A8 A7+

138 E3 20 ED AA E0 A0 AD+

139 A0 0A 0D

140 027F 20 20 46 34 20 20 2D+ DB ' F4 - включение и отключения курсивного вывода русского символа +

141 20 A2 AA AB EE E7 A5+ В', 10, 13

142 AD A8 A5 20 A8 20 AE+

143 E2 AA AB EE E7 A5 AD+

144 A8 EF 20 AA E3 E0 E1+

145 A8 A2 AD AE A3 AE 20+

146 A2 EB A2 AE A4 A0 20+

147 E0 E3 E1 E1 AA AE A3+

148 AE 20 E1 A8 AC A2 AE+

149 AB A0 20 82 0A 0D

150 02C4 20 20 46 35 20 20 2D+ DB ' F5 - включение и отключение частичной русификации клавиатуры +

151 20 A2 AA AB EE E7 A5+ (KVYJG -> ЛМНОП)', 10, 13

152 AD A8 A5 20 A8 20 AE+

153 E2 AA AB EE E7 A5 AD+

154 A8 A5 20 E7 A0 E1 E2+

155 A8 E7 AD AE A9 20 E0+

156 E3 E1 A8 E4 A8 AA A0+

157 E6 A8 A8 20 AA AB A0+

158 A2 A8 A0 E2 E3 E0 EB+

159 28 4B 56 59 4A 47 20+

160 2D 3E 20 8B 8C 8D 8E+

161 8F 29 0A 0D

162 0315 20 20 46 36 20 20 2D+ DB ' F6 - включение и отключение режима перевода строчных на +

163 20 A2 AA AB EE E7 A5+ прописные', 10, 13

164 AD A8 A5 20 A8 20 AE+

165 E2 AA AB EE E7 A5 AD+

166 A8 A5 20 E0 A5 A6 A8+

167 AC A0 20 AF A5 E0 A5+

168 A2 AE A4 A0 20 E1 E2+

169 E0 AE E7 AD EB E5 20+

170 AD A0 20 AF E0 AE AF+

171 A8 E1 AD EB A5 0A 0D

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 4

tsr2.ASM

172

173 =013D helpMsg\_length equ $-helpMsg

174 035B 8E E8 A8 A1 AA A0 20+ errorParamMsg DB 'Ошибка параметров коммандной +

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

176 E0 AE A2 20 AA AE AC+

177 AC A0 AD A4 AD AE A9+

178 20 E1 E2 E0 AE AA A8+

179 0A 0D

180 =0025 errorParamMsg\_length equ $-errorParamMsg

181

182 0380 DA 32\*(C4) BF tableTop DB 218, Line1\_length-2 dup+

183 (196), 191

184 =0034 tableTop\_length equ $-tableTop

185 03B4 C0 32\*(C4) D9 tableBottom DB 192, Line1\_length-2 dup (196), +

186 217

187 =0034 tableBottom\_length equ $-tableBottom

188

189 ; сообщения

190 03E8 90 A5 A7 A8 A4 A5 AD+ installedMsg DB 'Резидент загружен!$'

191 E2 20 A7 A0 A3 E0 E3+

192 A6 A5 AD 21 24

193 03FB 90 A5 A7 A8 A4 A5 AD+ alreadyInstalledMsg DB 'Резидент уже загружен$'

194 E2 20 E3 A6 A5 20 A7+

195 A0 A3 E0 E3 A6 A5 AD+

196 24

197 0411 8D A5 A4 AE E1 E2 A0+ noMemMsg DB 'Недостаточно памяти$'

198 E2 AE E7 AD AE 20 AF+

199 A0 AC EF E2 A8 24

200 0425 8D A5 20 E3 A4 A0 AB+ notInstalledMsg DB 'Не удалось загрузить резидент$'

201 AE E1 EC 20 A7 A0 A3+

202 E0 E3 A7 A8 E2 EC 20+

203 E0 A5 A7 A8 A4 A5 AD+

204 E2 24

205

206 0443 90 A5 A7 A8 A4 A5 AD+ removedMsg DB 'Резидент выгружен'

207 E2 20 A2 EB A3 E0 E3+

208 A6 A5 AD

209 =0011 removedMsg\_length equ $-removedMsg

210

211 0454 8D A5 20 E3 A4 A0 AB+ noRemoveMsg DB 'Не удалось выгрузить резидент'

212 AE E1 EC 20 A2 EB A3+

213 E0 E3 A7 A8 E2 EC 20+

214 E0 A5 A7 A8 A4 A5 AD+

215 E2

216 =001D noRemoveMsg\_length equ $-noRemoveMsg

217

218 0471 46 33 f1\_txt DB 'F3'

219 0473 46 34 f2\_txt DB 'F4'

220 0475 46 35 f3\_txt DB 'F5'

221 0477 46 36 f4\_txt DB 'F6'

222 =0002 fx\_length equ $-f4\_txt

223

224 0479 changeFx proc

225 0479 50 push AX

226 047A 53 push BX

227 047B 51 push CX

228 047C 52 push DX

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 5

tsr2.ASM

229 047D 55 push BP

230 047E 06 push ES

231 047F 33 DB xor BX, BX

232

233 0481 B4 03 mov AH, 03h

234 0483 CD 10 int 10h

235 0485 52 push DX

236

237 0486 0E push CS

238 0487 07 pop ES

239

240 0488 \_checkF1:

241 0488 BD 0471r lea BP, f1\_txt

242 048B B9 0002 mov CX, fx\_length

243 048E B7 00 mov BH, 0

244 0490 B6 00 mov DH, 0

245 0492 B2 4E mov DL, 78

246 0494 B8 1301 mov AX, 1301h

247

248 0497 80 3E 0150r FF cmp signaturePrintingEnabled, true

249 049C 74 07 je \_greenF1

250

251 049E \_redF1:

252 049E B3 4F mov BL, 01001111b ; red

253 04A0 CD 10 int 10h

254 04A2 EB 08 90 jmp \_checkF2

255

256 04A5 \_greenF1:

257 04A5 BD 0471r lea BP, f1\_txt

258 04A8 B3 2F mov BL, 00101111b ; green

259 04AA CD 10 int 10h

260

261 04AC \_checkF2:

262 04AC BD 0473r lea BP, f2\_txt

263 04AF B9 0002 mov CX, fx\_length

264 04B2 B7 00 mov BH, 0

265 04B4 B6 01 mov DH, 1

266 04B6 B2 4E mov DL, 78

267 04B8 B8 1301 mov AX, 1301h

268

269 04BB 80 3E 0151r FF cmp cursiveEnabled, true

270 04C0 74 07 je \_greenF2

271

272 04C2 \_redF2:

273 04C2 B3 4F mov BL, 01001111b ; red

274 04C4 CD 10 int 10h

275 04C6 EB 05 90 jmp \_checkF3

276

277 04C9 \_greenF2:

278 04C9 B3 2F mov BL, 00101111b ; green

279 04CB CD 10 int 10h

280

281 04CD \_checkF3:

282 04CD BD 0475r lea BP, f3\_txt

283 04D0 B9 0002 mov CX, fx\_length

284 04D3 B7 00 mov BH, 0

285 04D5 B6 02 mov DH, 2

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 6

tsr2.ASM

286 04D7 B2 4E mov DL, 78

287 04D9 B8 1301 mov AX, 1301h

288

289 04DC 80 3E 014Fr FF cmp translateEnabled, true

290 04E1 74 07 je \_greenF3

291

292 04E3 \_redF3:

293 04E3 B3 4F mov BL, 01001111b ; red

294 04E5 CD 10 int 10h

295 04E7 EB 05 90 jmp \_checkF4

296

297 04EA \_greenF3:

298 04EA B3 2F mov BL, 00101111b ; green

299 04EC CD 10 int 10h

300

301 04EE \_checkF4:

302 04EE BD 0477r lea BP, f4\_txt

303 04F1 B9 0002 mov CX, fx\_length

304 04F4 B7 00 mov BH, 0

305 04F6 B6 03 mov DH, 3

306 04F8 B2 4E mov DL, 78

307 04FA B8 1301 mov AX, 1301h

308

309 04FD 80 3E 0144r FF cmp ignoreEnabled, true

310 0502 74 07 je \_greenF4

311

312 0504 \_redF4:

313 0504 B3 4F mov BL, 01001111b ; red

314 0506 CD 10 int 10h

315 0508 EB 05 90 jmp \_outFx

316

317 050B \_greenF4:

318 050B B3 2F mov BL, 00101111b ; green

319 050D CD 10 int 10h

320

321 050F \_outFx:

322 050F 5A pop DX

323 0510 B4 02 mov AH, 02h

324 0512 CD 10 int 10h

325

326 0514 07 pop ES

327 0515 5D pop BP

328 0516 5A pop DX

329 0517 59 pop CX

330 0518 5B pop BX

331 0519 58 pop AX

332 051A C3 ret

333 051B changeFx endp

334

335 ;новый обработчик

336 051B new\_int9h proc far

337 ; сохраняем значения всех, изменяемых регистров в стэке

338 051B 56 push SI

339 051C 50 push AX

340 051D 53 push BX

341 051E 51 push CX

342 051F 52 push DX

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 7

tsr2.ASM

343 0520 06 push ES

344 0521 1E push DS

345 ; синхронизируем CS и DS

346 0522 0E push CS

347 0523 1F pop DS

348

349 0524 B8 0040 mov AX, 40h ; 40h-сегмент,где хранятся флаги сост-я клавиатуры, кольц. +

350 буфер ввода

351 0527 8E C0 mov ES, AX

352 0529 E4 60 in AL, 60h ; записываем в AL скан-код нажатой клавиши

353

354

355 ;@ далее - код для всех вариантов

356

357 ;проверка F3-F6

358 052B \_test\_Fx:

359 052B 2C 3A sub AL, 58 ; в AL теперь номер функциональной клавиши

360 052D \_F1:

361 052D 3C 03 cmp AL, 3 ; F1

362 052F 75 0A jne \_F2

363 0531 F6 16 0150r not signaturePrintingEnabled

364 0535 E8 FF41 call changeFx

365 0538 EB 2E 90 jmp \_translate\_or\_ignore

366 053B \_F2:

367 053B 3C 04 cmp AL, 4 ; F2

368 053D 75 0D jne \_F3

369 053F F6 16 0151r not cursiveEnabled

370 0543 E8 FF33 call changeFx

371 0546 E8 01D3 call setCursive ; перевод символа в курсив и обратно в зависимости от +

372 флага cursiveEnabled

373 0549 EB 1D 90 jmp \_translate\_or\_ignore

374 054C \_F3:

375 054C 3C 05 cmp AL, 5 ; F3

376 054E 75 0A jne \_F4

377 0550 F6 16 014Fr not translateEnabled

378 0554 E8 FF22 call changeFx

379 0557 EB 0F 90 jmp \_translate\_or\_ignore

380 055A \_F4:

381 055A 3C 06 cmp AL, 6 ; F4

382 055C 75 0A jne \_translate\_or\_ignore

383 055E F6 16 0144r not ignoreEnabled

384 0562 E8 FF14 call changeFx

385 0565 EB 01 90 jmp \_translate\_or\_ignore

386

387 ;игнорирование и перевод

388 0568 \_translate\_or\_ignore:

389

390 0568 9C pushf

391 0569 2E: FF 1E 0173r call dword ptr CS:[old\_int9hOffset] ; вызываем стандартный обработчик прерывания

392 056E B8 0040 mov AX, 40h ; 40h-сегмент,где хранятся флаги сост-я клавы,кольц. +

393 буфер ввода

394 0571 8E C0 mov ES, AX

395 0573 26: 8B 1E 001C mov BX, ES:[1Ch] ; адрес хвоста

396 0578 4B dec BX ; сместимся назад к последнему

397 0579 4B dec BX ; введённому символу

398 057A 83 FB 1E cmp BX, 1Eh ; не вышли ли мы за пределы буфера?

399 057D 73 03 jae \_go

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 8

tsr2.ASM

400 057F BB 003C mov BX, 3Ch ; хвост вышел за пределы буфера, значит последний введённый +

401 символ

402 ; находится в конце буфера

403

404 0582 \_go:

405 0582 26: 8B 17 mov DX, ES:[BX] ; в DX 0 введённый символ

406 ;включен ли режим блокировки ввода?

407 0585 80 3E 0144r FF cmp ignoreEnabled, true

408 058A 75 21 jne \_check\_translate

409

410 ; да, включен

411 058C BE 0000 mov SI, 0

412 058F B9 0041 mov CX, ignoredLength ;кол-во игнорируемых символов

413

414 ; проверяем, присутствует ли текущий символ в списке игнорируемых

415 0592 \_check\_ignored:

416 0592 3A 94 0103r cmp DL,ignoredChars[SI]

417 0596 74 06 je \_block

418 0598 46 inc SI

419 0599 E2 F7 loop \_check\_ignored

420 059B EB 10 90 jmp \_check\_translate

421

422 ; блокируем

423 059E \_block:

424 ;@mov ES:[1Ch], BX ;блокировка ввода символа

425 059E 26: 89 07 mov ES:[BX], AX

426 05A1 33 C0 xor AX, AX

427 05A3 8A 84 0124r mov AL, replaceWith[SI]

428 05A7 26: 89 07 mov ES:[BX], AX ; замена символа

429 05AA EB 23 90 jmp \_quit

430

431 05AD \_check\_translate:

432 ; включен ли режим перевода?

433 05AD 80 3E 014Fr FF cmp translateEnabled, true

434 05B2 75 1B jne \_quit

435

436 ; да, включен

437 05B4 BE 0000 mov SI, 0

438 05B7 B9 0005 mov CX, translateLength ; кол-во символов для перевода

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

440 05BA \_check\_translate\_loop:

441 05BA 3A 94 0145r cmp DL, translateFrom[SI]

442 05BE 74 06 je \_translate

443 05C0 46 inc SI

444 05C1 E2 F7 loop \_check\_translate\_loop

445 05C3 EB 0A 90 jmp \_quit

446

447 ; переводим

448 05C6 \_translate:

449 05C6 33 C0 xor AX, AX

450 05C8 8A 84 014Ar mov AL, translateTo[SI]

451 05CC 26: 89 07 mov ES:[BX], AX ; замена символа

452

453 05CF \_quit:

454 ; восстанавливаем все регистры

455 05CF 1F pop DS

456 05D0 07 pop ES

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 9

tsr2.ASM

457 05D1 5A pop DX

458 05D2 59 pop CX

459 05D3 5B pop BX

460 05D4 58 pop AX

461 05D5 5E pop SI

462 05D6 CF iret

463 05D7 new\_int9h endp

464

465 ;=== Обработчик прерывания int 1Ch ===;

466 ;=== Вызывается каждые 55 мс ===;

467 05D7 new\_int1Ch proc far

468 05D7 50 push AX

469 05D8 0E push CS

470 05D9 1F pop DS

471

472 05DA 9C pushf

473 05DB 2E: FF 1E 0177r call dword ptr CS:[old\_int1ChOffset]

474

475 05E0 80 3E 0150r FF cmp signaturePrintingEnabled, true ; если нажата управляющая клавиша (в данном случае +

476 F1)

477 05E5 75 1C jne \_notToPrint

478

479 05E7 83 3E 0180r 25 cmp counter, printDelay\*1000/55 + 1 ; если кол-во "тактов" эквивалентно +

480 %printDelay% секундам

481 05EC 74 03 je \_letsPrint

482

483 05EE EB 0E 90 jmp \_dontPrint

484

485 05F1 \_letsPrint:

486 05F1 F6 16 0150r not signaturePrintingEnabled

487 05F5 C7 06 0180r 0000 mov counter, 0

488 05FB E8 0094 call printSignature

489

490 05FE \_dontPrint:

491 05FE 83 06 0180r 01 add counter, 1

492

493 0603 \_notToPrint:

494

495 0603 58 pop AX

496

497 0604 CF iret

498 0605 new\_int1Ch endp

499

500 ;=== Обработчик прерывания int 2Fh ===;

501 ;=== Служит для:

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

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

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

505 ;===

506 0605 new\_int2Fh proc

507 0605 80 FC FF cmp AH, 0FFh ;наша функция?

508 0608 75 0B jne \_2Fh\_std ;нет - на старый обработчик

509 060A 3C 00 cmp AL, 0 ;подфункция проверки, загружен ли резидент в память?

510 060C 74 0C je \_already\_installed

511 060E 3C 01 cmp AL, 1 ;подфункция выгрузки из памяти?

512 0610 74 0B je \_uninstall

513 0612 EB 01 90 jmp \_2Fh\_std ;нет - на старый обработчик

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 10

tsr2.ASM

514

515 0615 \_2Fh\_std:

516 0615 2E: FF 2E 017Br jmp dword ptr CS:[old\_int2FhOffset] ;вызов старого обработчика

517

518 061A \_already\_installed:

519 061A B4 69 mov AH, 'i' ;вернём 'i', если резидент загружен в память

520 061C CF iret

521

522 061D \_uninstall:

523 061D 1E push DS

524 061E 06 push ES

525 061F 52 push DX

526 0620 53 push BX

527

528 0621 33 DB xor BX, BX

529

530 ; CS = ES, для доступа к переменным

531 0623 0E push CS

532 0624 07 pop ES

533

534 0625 B8 2509 mov AX, 2509h

535 0628 26: 8B 16 0173r mov DX, ES:old\_int9hOffset ; возвращаем вектор прерывания

536 062D 26: 8E 1E 0175r mov DS, ES:old\_int9hSegment ; на место

537 0632 CD 21 int 21h

538

539 0634 B8 251C mov AX, 251Ch

540 0637 26: 8B 16 0177r mov DX, ES:old\_int1ChOffset ; возвращаем вектор прерывания

541 063C 26: 8E 1E 0179r mov DS, ES:old\_int1ChSegment ; на место

542 0641 CD 21 int 21h

543

544 0643 B8 252F mov AX, 252Fh

545 0646 26: 8B 16 017Br mov DX, ES:old\_int2FhOffset ; возвращаем вектор прерывания

546 064B 26: 8E 1E 017Dr mov DS, ES:old\_int2FhSegment ; на место

547 0650 CD 21 int 21h

548

549 0652 2E: 8E 06 002C mov ES, CS:2Ch ; загрузим в ES адрес окружения

550 0657 B4 49 mov AH, 49h ; выгрузим из памяти окружение

551 0659 CD 21 int 21h

552 065B 72 0B jc \_notRemove

553

554 065D 0E push CS

555 065E 07 pop ES ;в ES - адрес резидентной программы

556 065F B4 49 mov AH, 49h ;выгрузим из памяти резидент

557 0661 CD 21 int 21h

558 0663 72 03 jc \_notRemove

559 0665 EB 15 90 jmp \_unloaded

560

561 0668 \_notRemove: ; не удалось выполнить выгрузку

562 ; вывод сообщения о неудачной выгрузке

563 0668 B4 03 mov AH, 03h ; получаем позицию курсора

564 066A CD 10 int 10h

565 066C BD 0454r lea BP, noRemoveMsg

566 066F B9 001D mov CX, noRemoveMsg\_length

567 0672 B3 07 mov BL, 0111b

568 0674 B8 1301 mov AX, 1301h

569 0677 CD 10 int 10h

570 0679 EB 12 90 jmp \_2Fh\_exit

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 11

tsr2.ASM

571

572 067C \_unloaded: ; выгрузка прошла успешно

573 ; вывод сообщения об удачной выгрузке

574 067C B4 03 mov AH, 03h ; получаем позицию курсора

575 067E CD 10 int 10h

576 0680 BD 0443r lea BP, removedMsg

577 0683 B9 0011 mov CX, removedMsg\_length

578 0686 B3 07 mov BL, 0111b

579 0688 B8 1301 mov AX, 1301h

580 068B CD 10 int 10h

581

582 068D \_2Fh\_exit:

583 068D 5B pop BX

584 068E 5A pop DX

585 068F 07 pop ES

586 0690 1F pop DS

587 0691 CF iret

588 0692 new\_int2Fh endp

589

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

591 ;=== Настраивается значениями переменных в начале исходника

592 ;===

593 0692 printSignature proc

594 0692 50 push AX

595 0693 52 push DX

596 0694 51 push CX

597 0695 53 push BX

598 0696 06 push ES

599 0697 54 push SP

600 0698 55 push BP

601 0699 56 push SI

602 069A 57 push DI

603

604 069B 33 C0 xor AX, AX

605 069D 33 DB xor BX, BX

606 069F 33 D2 xor DX, DX

607

608 06A1 B4 03 mov AH, 03h ;чтение текущей позиции курсора

609 06A3 CD 10 int 10h

610 06A5 52 push DX ;помещаем информацию о +

611 положении курсора в стек

612

613

614 ;по варианту вывод по низу

615 06A6 \_printBottom:

616 06A6 B6 13 mov DH, 19

617 06A8 B2 0F mov DL, 15

618 06AA EB 01 90 jmp \_actualPrint

619

620 06AD \_actualPrint:

621 06AD B4 0F mov AH, 0Fh ;чтение текущего видеорежима. в+

622 BH - текущая страница

623 06AF CD 10 int 10h

624

625 06B1 0E push CS

626 06B2 07 pop ES ;указываем ES на CS

627

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 12

tsr2.ASM

628 ;вывод 'верхушки' таблицы

629 06B3 52 push DX

630 06B4 BD 0380r lea BP, tableTop ;помещаем в BP указатель на +

631 выводимую строку

632 06B7 B9 0034 mov CX, tableTop\_length ;в CX - длина строки

633 06BA B3 07 mov BL, 0111b ;цвет выводимого текста ref: +

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

635 06BC B8 1301 mov AX, 1301h ;AH=13h - номер ф-ии, AL=01h - +

636 курсор перемещается при выводе каждого из символов строки

637 06BF CD 10 int 10h

638 06C1 5A pop DX

639 06C2 FE C6 inc DH

640

641

642 ;вывод первой линии

643 06C4 52 push DX

644 06C5 BD 0182r lea BP, signatureLine1

645 06C8 B9 0034 mov CX, Line1\_length

646 06CB B3 07 mov BL, 0111b

647 06CD B8 1301 mov AX, 1301h

648 06D0 CD 10 int 10h

649 06D2 5A pop DX

650 06D3 FE C6 inc DH

651

652 ;вывод второй линии

653 06D5 52 push DX

654 06D6 BD 01B6r lea BP, signatureLine2

655 06D9 B9 0034 mov CX, Line2\_length

656 06DC B3 07 mov BL, 0111b

657 06DE B8 1301 mov AX, 1301h

658 06E1 CD 10 int 10h

659 06E3 5A pop DX

660 06E4 FE C6 inc DH

661

662 ;вывод третьей линии

663 06E6 52 push DX

664 06E7 BD 01EAr lea BP, signatureLine3

665 06EA B9 0034 mov CX, Line3\_length

666 06ED B3 07 mov BL, 0111b

667 06EF B8 1301 mov AX, 1301h

668 06F2 CD 10 int 10h

669 06F4 5A pop DX

670 06F5 FE C6 inc DH

671

672 ;вывод 'низа' таблицы

673 06F7 52 push DX

674 06F8 BD 03B4r lea BP, tableBottom

675 06FB B9 0034 mov CX, tableBottom\_length

676 06FE B3 07 mov BL, 0111b

677 0700 B8 1301 mov AX, 1301h

678 0703 CD 10 int 10h

679 0705 5A pop DX

680 0706 FE C6 inc DH

681

682 0708 33 DB xor BX, BX

683 070A 5A pop DX ;восстанавливаем из стека +

684 прежнее положение курсора

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 13

tsr2.ASM

685 070B B4 02 mov AH, 02h ;меняем положение курсора на +

686 первоначальное

687 070D CD 10 int 10h

688 070F E8 FD67 call changeFx

689

690 0712 5F pop DI

691 0713 5E pop SI

692 0714 5D pop BP

693 0715 5C pop SP

694 0716 07 pop ES

695 0717 5B pop BX

696 0718 59 pop CX

697 0719 5A pop DX

698 071A 58 pop AX

699

700 071B C3 ret

701 071C printSignature endp

702

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

704 на обычное и наоброт

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

706 071C setCursive proc

707 071C 06 push ES ; сохраняем регистры

708 071D 50 push AX

709 071E 0E push CS

710 071F 07 pop ES

711

712 0720 80 3E 0151r FF cmp cursiveEnabled, true

713 0725 75 30 jne \_restoreSymbol

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

715 ; предварительно сохраняя старый символ в savedSymbol

716

717 0727 E8 004C call saveFont

718 072A 8A 0E 0162r mov CL, charToCursiveIndex

719 072E \_shifTtable:

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

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

722 072E 83 C5 10 add BP, 16

723 0731 E2 FB loop \_shiftTable

724

725 ; пpи savefont смещается pегистp ES

726 ; поэтомy пpиходится делать такие махинации, чтобы

727 ; записать полyченный элемент в savedSymbol

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

729 0733 1E push DS

730 0734 58 pop AX

731 0735 06 push ES

732 0736 1F pop DS

733 0737 50 push AX

734 0738 07 pop ES

735 0739 50 push AX

736

737 073A 8B F5 mov SI, BP

738 073C BF 0163r lea DI, savedSymbol

739 ; сохpаняем в пеpеменнyю savedSymbol

740 ; таблицy нyжного символа

741 073F B9 0010 mov CX, 16

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 14

tsr2.ASM

742 ; movsb из DS:SI в ES:DI

743 0742 F3> A4 rep movsb

744 ; исходные позиции сегментов возвpащены

745 0744 1F pop DS ; восстановление DS

746

747 ; заменим написание символа на кypсив

748 0745 B9 0001 mov CX, 1

749 0748 B6 00 mov DH, 0

750 074A 8A 16 0162r mov DL, charToCursiveIndex

751 074E BD 0152r lea BP, cursiveSymbol

752 0751 E8 0015 call changeFont

753 0754 EB 10 90 jmp \_exitSetCursive

754

755 0757 \_restoreSymbol:

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

757

758 0757 B9 0001 mov CX, 1

759 075A B6 00 mov DH, 0

760 075C 8A 16 0162r mov DL, charToCursiveIndex

761 0760 BD 0163r lea bp, savedSymbol

762 0763 E8 0003 call changeFont

763

764 0766 \_exitSetCursive:

765 0766 58 pop AX

766 0767 07 pop ES

767 0768 C3 ret

768 0769 setCursive endp

769

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

771 ;===

772 ; \*\*\* входные данные

773 ; DL = номер символа для замены

774 ; CX = Кол-во символов заменяемых изображений символов

775 ; (начиная с символа указанного в DX)

776 ; ES:bp = адрес таблицы

777 ;

778 ; \*\*\* описание работы процедуры

779 ; Происходит вызов int 10h (видеосервис)

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

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

782 ; символа для текущего шрифта

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

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

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

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

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

788 ;

789 ; \*\*\* результат

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

791 ; на предложенное пользователем.

792 ; Изменению подвергнутся все символы, находящиеся на экране,

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

794

795 0769 changeFont proc

796 0769 50 push AX

797 076A 53 push BX

798 076B B8 1100 mov AX, 1100h

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 15

tsr2.ASM

799 076E BB 1000 mov BX, 1000h

800 0771 CD 10 int 10h

801 0773 58 pop AX

802 0774 5B pop BX

803 0775 C3 ret

804 0776 changeFont endp

805

806 ;=== Функция сохранения нормального начертания символа

807 ;===

808 ; \*\*\* входные данные

809 ; BH - тип возвращаемой символьной таблицы

810 ; 0 - таблица из int 1fh

811 ; 1 - таблица из int 44h

812 ; 2-5 - таблица из 8x14, 8x8, 8x8 (top), 9x14

813 ; 6 - 8x16

814 ;

815 ; \*\*\* описание работы процедуры

816 ; Происходит вызов int 10h (видеосервис)

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

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

819 ;

820 ; \*\*\* результат

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

822 ; в CX находится байт на символ

823 ; в DL количество экранных строк

824 ; ВАЖНО! Происходит сдвиг регистра ES

825 ; ( ES становится равным C000h )

826

827 0776 saveFont proc

828 0776 50 push AX

829 0777 53 push BX

830 0778 B8 1130 mov AX, 1130h

831 077B BB 0600 mov BX, 0600h

832 077E CD 10 int 10h

833 0780 58 pop AX

834 0781 5B pop BX

835 0782 C3 ret

836 0783 saveFont endp

837

838

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

840 ;===

841 0783 \_initTSR: ; старт резидента

842 0783 B4 03 mov AH, 03h

843 0785 CD 10 int 10h

844 0787 52 push DX

845 0788 B4 00 mov AH,00h ; установка видеорежима (83h текст +

846 80x25 16/8 CGA,EGA b800 Comp,RGB,Enhanced), без очистки экрана

847 078A B0 83 mov AL,83h

848 078C CD 10 int 10h

849 078E 5A pop DX

850 078F B4 02 mov AH, 02h

851 0791 CD 10 int 10h

852

853

854 0793 E8 008C call commandParamsParser

855 0796 B8 3509 mov AX,3509h ; получить в ES:BX вектор 09

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 16

tsr2.ASM

856 0799 CD 21 int 21h ; прерывания

857

858

859 079B 80 3E 017Fr FF cmp notLoadTSR, true ; если была выведена справка

860 07A0 74 0E je \_exit\_tmp ; просто выходим

861

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

863 комментируем 5 строк ниже

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

865 07A2 B4 FF mov AH, 0FFh

866 07A4 B0 00 mov AL, 0

867 07A6 CD 2F int 2Fh

868 07A8 80 FC 69 cmp AH, 'i' ; проверка того, загружена ли уже программа

869 07AB 74 62 je \_alreadyInstalled

870

871 07AD EB 04 90 jmp \_tmp

872

873 07B0 \_exit\_tmp:

874 07B0 EB 6E 90 jmp \_exit

875

876 07B3 \_tmp:

877 07B3 06 push ES

878 07B4 A1 002C mov AX, DS:[2Ch] ; psp

879 07B7 8E C0 mov ES, AX

880 07B9 B4 49 mov AH, 49h ; хватит памяти чтоб остаться

881 07BB CD 21 int 21h ; резидентом?

882 07BD 07 pop ES

883 07BE 72 59 jc \_notMem ; не хватило - выходим

884

885 ;== int 09h ==;

886

887 07C0 2E: 89 1E 0173r mov word ptr CS:old\_int9hOffset, BX

888 07C5 2E: 8C 06 0175r mov word ptr CS:old\_int9hSegment, ES

889 07CA B8 2509 mov AX, 2509h ; установим вектор на 09

890 07CD BA 051Br mov DX, offset new\_int9h ; прерывание

891 07D0 CD 21 int 21h

892

893 ;== int 1Ch ==;

894 07D2 B8 351C mov AX,351Ch ; получить в ES:BX вектор 1C

895 07D5 CD 21 int 21h ; прерывания

896 07D7 2E: 89 1E 0177r mov word ptr CS:old\_int1ChOffset, BX

897 07DC 2E: 8C 06 0179r mov word ptr CS:old\_int1ChSegment, ES

898 07E1 B8 251C mov AX, 251Ch ; установим вектор на 1C

899 07E4 BA 05D7r mov DX, offset new\_int1Ch ; прерывание

900 07E7 CD 21 int 21h

901

902 ;== int 2Fh ==;

903 07E9 B8 352F mov AX,352Fh ; получить в ES:BX вектор 1C

904 07EC CD 21 int 21h ; прерывания

905 07EE 2E: 89 1E 017Br mov word ptr CS:old\_int2FhOffset, BX

906 07F3 2E: 8C 06 017Dr mov word ptr CS:old\_int2FhSegment, ES

907 07F8 B8 252F mov AX, 252Fh ; установим вектор на 2F

908 07FB BA 0605r mov DX, offset new\_int2Fh ; прерывание

909 07FE CD 21 int 21h

910

911 0800 E8 FC76 call changeFx

912 0803 BA 03E8r mov DX, offset installedMsg ; выводим что все ок

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 17

tsr2.ASM

913 0806 B4 09 mov AH, 9

914 0808 CD 21 int 21h

915 080A BA 0783r mov DX, offset \_initTSR ; остаемся в памяти резидентом

916 080D CD 27 int 27h ; и выходим

917 ; конец основной программы

918 080F \_alreadyInstalled:

919 080F B4 09 mov AH, 09h

920 0811 BA 03FBr lea DX, alreadyInstalledMsg

921 0814 CD 21 int 21h

922 0816 EB 08 90 jmp \_exit

923 0819 \_notMem: ; не хватает памяти, чтобы остаться резидентом

924 0819 BA 0411r mov DX, offset noMemMsg

925 081C B4 09 mov AH, 9

926 081E CD 21 int 21h

927 0820 \_exit: ; выход

928 0820 CD 20 int 20h

929

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

931 ;===

932 0822 commandParamsParser proc

933 0822 0E push CS

934 0823 07 pop ES

935 0824 C6 06 017Fr 00 mov notLoadTSR, 0

936

937 0829 BE 0080 mov SI, 80h ;SI=смещение командной строки.

938 082C AC lodsb ;Получим кол-во символов.

939 082D 0A C0 or AL, AL ;Если 0 символов введено,

940 082F 74 3A jz \_exitHelp ;то все в порядке.

941

942 0831 \_nextChar:

943

944 0831 46 inc SI ;Теперь SI указывает на первый символ +

945 строки.

946

947 0832 80 3C 0D cmp [SI], BYTE ptr 13

948 0835 74 34 je \_exitHelp

949

950

951 0837 AD lodsw ;Получаем два символа

952 0838 3D 3F2F cmp AX, '?/' ;Это '/?' (данные расположены в +

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

954 083B 74 03 je \_question

955

956 ;cmp AH, '/'

957 ;je \_errorParam

958

959 083D EB 2C 90 jmp \_exitHelp

960

961 0840 \_question:

962 ; вывод строки помощи

963 0840 B4 03 mov AH,03

964 0842 CD 10 int 10h

965 0844 BD 021Er lea BP, helpMsg

966 0847 B9 013D mov CX, helpMsg\_length

967 084A B3 07 mov BL, 0111b

968 084C B8 1301 mov AX, 1301h

969 084F CD 10 int 10h

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 18

tsr2.ASM

970 ; конец вывода строки помощи

971 0851 F6 16 017Fr not notLoadTSR ;флаг того, что необходимо не загружать резидент

972 0855 EB DA jmp \_nextChar

973

974 0857 EB 12 90 jmp \_exitHelp

975

976 085A \_errorParam:

977 ;вывод строки

978 085A B4 03 mov AH,03

979 085C CD 10 int 10h

980 085E BD 035Br lea BP, CS:errorParamMsg

981 0861 B9 0025 mov CX, errorParamMsg\_length

982 0864 B3 07 mov BL, 0111b

983 0866 B8 1301 mov AX, 1301h

984 0869 CD 10 int 10h

985 ;конец вывода строки

986 086B \_exitHelp:

987 086B C3 ret

988 086C commandParamsParser endp

989

990 086C code ends

991 end \_start

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 19

Symbol Table

Symbol Name Type Value

??DATE Text "04/29/21"

??FILENAME Text "tsr2 "

??TIME Text "22:19:48"

??VERSION Number 030A

@CPU Text 0101H

@CURSEG Text CODE

@FILENAME Text TSR2

@WORDSIZE Text 2

ALREADYINSTALLEDMSG Byte CODE:03FB

CHANGEFONT Near CODE:0769

CHANGEFX Near CODE:0479

CHARTOCURSIVEINDEX Byte CODE:0162

COMMANDPARAMSPARSER Near CODE:0822

COUNTER Word CODE:0180

CURSIVEENABLED Byte CODE:0151

CURSIVESYMBOL Byte CODE:0152

ERRORPARAMMSG Byte CODE:035B

ERRORPARAMMSG\_LENGTH Number 0025

F1\_TXT Byte CODE:0471

F2\_TXT Byte CODE:0473

F3\_TXT Byte CODE:0475

F4\_TXT Byte CODE:0477

FX\_LENGTH Number 0002

HELPMSG Byte CODE:021E

HELPMSG\_LENGTH Number 013D

IGNOREDCHARS Byte CODE:0103

IGNOREDLENGTH Number 0041

IGNOREENABLED Byte CODE:0144

INSTALLEDMSG Byte CODE:03E8

LINE1\_LENGTH Number 0034

LINE2\_LENGTH Number 0034

LINE3\_LENGTH Number 0034

NEW\_INT1CH Far CODE:05D7

NEW\_INT2FH Near CODE:0605

NEW\_INT9H Far CODE:051B

NOMEMMSG Byte CODE:0411

NOREMOVEMSG Byte CODE:0454

NOREMOVEMSG\_LENGTH Number 001D

NOTINSTALLEDMSG Byte CODE:0425

NOTLOADTSR Byte CODE:017F

OLD\_INT1CHOFFSET Word CODE:0177

OLD\_INT1CHSEGMENT Word CODE:0179

OLD\_INT2FHOFFSET Word CODE:017B

OLD\_INT2FHSEGMENT Word CODE:017D

OLD\_INT9HOFFSET Word CODE:0173

OLD\_INT9HSEGMENT Word CODE:0175

PRINTDELAY Number 0002

PRINTSIGNATURE Near CODE:0692

REMOVEDMSG Byte CODE:0443

REMOVEDMSG\_LENGTH Number 0011

REPLACEWITH Byte CODE:0124

SAVEDSYMBOL Byte CODE:0163

SAVEFONT Near CODE:0776

SETCURSIVE Near CODE:071C

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 20

Symbol Table

SIGNATURELINE1 Byte CODE:0182

SIGNATURELINE2 Byte CODE:01B6

SIGNATURELINE3 Byte CODE:01EA

SIGNATUREPRINTINGENABLED Byte CODE:0150

TABLEBOTTOM Byte CODE:03B4

TABLEBOTTOM\_LENGTH Number 0034

TABLETOP Byte CODE:0380

TABLETOP\_LENGTH Number 0034

TRANSLATEENABLED Byte CODE:014F

TRANSLATEFROM Byte CODE:0145

TRANSLATELENGTH Number 0005

TRANSLATETO Byte CODE:014A

TRUE Number 00FF

\_2FH\_EXIT Near CODE:068D

\_2FH\_STD Near CODE:0615

\_ACTUALPRINT Near CODE:06AD

\_ALREADYINSTALLED Near CODE:080F

\_ALREADY\_INSTALLED Near CODE:061A

\_BLOCK Near CODE:059E

\_CHECKF1 Near CODE:0488

\_CHECKF2 Near CODE:04AC

\_CHECKF3 Near CODE:04CD

\_CHECKF4 Near CODE:04EE

\_CHECK\_IGNORED Near CODE:0592

\_CHECK\_TRANSLATE Near CODE:05AD

\_CHECK\_TRANSLATE\_LOOP Near CODE:05BA

\_DONTPRINT Near CODE:05FE

\_ERRORPARAM Near CODE:085A

\_EXIT Near CODE:0820

\_EXITHELP Near CODE:086B

\_EXITSETCURSIVE Near CODE:0766

\_EXIT\_TMP Near CODE:07B0

\_F1 Near CODE:052D

\_F2 Near CODE:053B

\_F3 Near CODE:054C

\_F4 Near CODE:055A

\_GO Near CODE:0582

\_GREENF1 Near CODE:04A5

\_GREENF2 Near CODE:04C9

\_GREENF3 Near CODE:04EA

\_GREENF4 Near CODE:050B

\_INITTSR Near CODE:0783

\_LETSPRINT Near CODE:05F1

\_NEXTCHAR Near CODE:0831

\_NOTMEM Near CODE:0819

\_NOTREMOVE Near CODE:0668

\_NOTTOPRINT Near CODE:0603

\_OUTFX Near CODE:050F

\_PRINTBOTTOM Near CODE:06A6

\_QUESTION Near CODE:0840

\_QUIT Near CODE:05CF

\_REDF1 Near CODE:049E

\_REDF2 Near CODE:04C2

\_REDF3 Near CODE:04E3

\_REDF4 Near CODE:0504

\_RESTORESYMBOL Near CODE:0757

\_SHIFTTABLE Near CODE:072E

Turbo Assembler Version 3.1 04/29/21 22:19:48 Page 21

Symbol Table

\_START Near CODE:0100

\_TEST\_FX Near CODE:052B

\_TMP Near CODE:07B3

\_TRANSLATE Near CODE:05C6

\_TRANSLATE\_OR\_IGNORE Near CODE:0568

\_UNINSTALL Near CODE:061D

\_UNLOADED Near CODE:067C

Groups & Segments Bit Size Align Combine Class

CODE 16 086C Para none CODE

1. **Файл unloader.lst**

Turbo Assembler Version 3.1 05/19/23 14:32:48 Page 1

unldtsr.ASM

1

2

3 0000 code segment 'code'

4 assume CS:code, DS:code

5 org 100h

6 0100 \_start:

7

8 0100 B4 FF mov AH, 0FFh

9 0102 B0 01 mov AL, 1

10 0104 CD 2F int 2Fh

11 0106 CD 20 int 20h

12

13 0108 code ends

14 end \_start

Turbo Assembler Version 3.1 05/19/23 14:32:48 Page 2

Symbol Table

Symbol Name Type Value

??DATE Text "05/19/23"

??FILENAME Text "unldtsr "

??TIME Text "14:28:23"

??VERSION Number 030A

@CPU Text 0101H

@CURSEG Text CODE

@FILENAME Text UNLDTSR

@WORDSIZE Text 2

\_START Near CODE:0100

Groups & Segments Bit Size Align Combine Class

CODE 16 0108 Para none CODE