Московский государственный технический университет

им. Н.Э. Баумана

|  |  |
| --- | --- |
| УТВЕРЖДАЮ: |  |
| Большаков С.А. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2013 г. |

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

Вариант №0

Исходный текст программного продукта

(вид документа)

писчая бумага

(вид носителя)

20

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛИ: |  |
| студенты группы ИУ5-44 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Латкин И.И., Леонтьев А.В., Назаров К.В. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2013 г. |

Москва – 2013

**Содержание**

[1. Файл tsr.lst 3](#_Toc354941476)

[2. Файл unloader.lst 19](#_Toc354941477)

1. **Файл tsr.lst**

Turbo Assembler Version 3.1

tsr.asm

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

2 ; main.asm

3 ;

4 ; Сборка:

5 ; tasm.exe /l main.asm

6 ; tlink /t /x main.obj

7 ;

8 ; Примечания:

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

10 ;

11 ; Авторы:

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

13 ; Леонтьев А.В.

14 ; Латкин И.И.

15 ; Назаров К.В.

16 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

17

18 0000 code segment 'code'

19 assume CS:code, DS:code

20 org 100h

21 0100 \_start:

22

23 0100 E9 05BB jmp \_initTSR ; на начало программы

24

25 ; данные

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

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

28 6F 70 71 72 73 74 75+

29 76 77 78 79 7A

30 =001A ignoredLength equ $-ignoredChar +

31 ; длина строки ignoredChars

32 011D 00 ignoreEnabled DB 0 +

33 ; флаг функции игнорирования ввода

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

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

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

37 ;@ символы на которые будет идти замена

38 =0005 translateLength equ $-translateTo +

39 ; длина строки trasnlateFrom

40 0128 00 translateEnabled DB 0 +

41 ; флаг функции перевода

42

43 0129 00 signaturePrintingEnabled DB 0 +

44 ; флаг функции вывода информации об авторе

45 012A 00 cursiveEnabled DB 0 +

46 ; флаг перевода символа в курсив

47

48 012B 00 cursiveSymbol DB 00000000b +

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

50 012C 00 DB 00000000b

51 012D 00 DB 00000000b

52 012E 3E DB 00111110b

53 012F 3F DB 00111111b

54 0130 33 DB 00110011b

55 0131 66 DB 01100110b

56 0132 66 DB 01100110b

57 0133 7C DB 01111100b

58 0134 C6 DB 11000110b

59 0135 C6 DB 11000110b

60 0136 C6 DB 11000110b

61 0137 FC DB 11111100b

62 0138 00 DB 00000000b

63 0139 00 DB 00000000b

64 013A 00 DB 00000000b

65

66 013B 82 charToCursiveIndex DB 'В' +

67 ;@ символ для замены

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

69 ; переменная для хранения старого символа

70

71 =00FF true equ 0FFh +

72 ; константа истинности

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

74 ; адрес старого обработчика int 9h

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

76 ; сегмент старого обработчика int 9h

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

78 ; адрес старого обработчика int 1Ch

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

80 ; сегмент старого обработчика int 1Ch

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

82 ; адрес старого обработчика int 2Fh

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

84 ; сегмент старого обработчика int 2Fh

85

86 0158 00 unloadTSR DB 0 +

87 ; 1 - выгрузить резидент

88 0159 00 notLoadTSR DB 0 +

89 ; 1 - не загружать

90 015A 0000 counter DW 0

91 =0002 printDelay equ 2 +

92 ;@ задержка перед выводом "подписи" в секундах

93 015C 0001 printPos DW 1 +

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

95

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

97 (1я строка).

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

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

100 20 80 AB A5 AA E1 A5+

101 A9 20 8B A5 AE AD E2+

102 EC A5 A2 2C 20 8A AE+

103 AD E1 E2 A0 AD E2 A8+

104 AD 20 8D A0 A7 A0 E0+

105 AE A2 B3

106 =0034 Line1\_length equ $-signatureLine1

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

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

109 20 20 20 20 20 20 20+

110 20 20 20 20 20 20 20+

111 20 20 20 20 20 20 20+

112 20 20 20 20 20 20 20+

113 20 20 20 20 20 20 20+

114 20 20 B3

115 =0034 Line2\_length equ $-signatureLine2

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

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

118 20 20 20 20 20 20 20+

119 20 20 20 20 20 20 20+

120 20 20 20 20 20 20 20+

121 20 20 20 20 20 20 20+

122 20 20 20 20 20 20 20+

123 20 20 B3

124 =0034 Line3\_length equ $-signatureLine3

125 01FA 6D 61 69 6E 2E 63 6F+ helpMsg DB 'main.com [/?] [/u]', 10, 13

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

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

128 020E 5B 2F 3F 5D 20 20 20+ DB '[/?] вывод данной +

129 20 A2 EB A2 AE A4 20+ справки', 10, 13

130 A4 A0 AD AD AE A9 20+

131 E1 AF E0 A0 A2 AA A8+

132 0A 0D

133 022C 5B 2F 75 5D 20 20 20+ DB '[/u] выгрузка +

134 20 A2 EB A3 E0 E3 A7+ резидента из памяти', 10, 13

135 AA A0 20 E0 A5 A7 A8+

136 A4 A5 AD E2 A0 20 A8+

137 A7 20 AF A0 AC EF E2+

138 A8 0A 0D

139 =0058 helpMsg\_length equ $-helpMsg

140 0252 8E E8 A8 A1 AA A0 20+ errorParamMsg DB 'Ошибка параметров коммандной +

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

142 E0 AE A2 20 AA AE AC+

143 AC A0 AD A4 AD AE A9+

144 20 E1 E2 E0 AE AA A8+

145 0A 0D

146 =0025 errorParamMsg\_length equ $-errorParamMsg

147

148 0277 DA 32\*(C4) BF tableTop DB 218, Line1\_length-2 dup+

149 (196), 191

150 =0034 tableTop\_length equ $-tableTop

151 02AB C0 32\*(C4) D9 tableBottom DB 192, Line1\_length-2 dup (196), +

152 217

153 =0034 tableBottom\_length equ $-tableBottom

154

155 ; сообщения

156 02DF 90 A5 A7 A8 A4 A5 AD+ installedMsg DB 'Резидент загружен!$'

157 E2 20 A7 A0 A3 E0 E3+

158 A6 A5 AD 21 24

159 02F2 90 A5 A7 A8 A4 A5 AD+ alreadyInstalledMsg DB 'Резидент уже загружен$'

160 E2 20 E3 A6 A5 20 A7+

161 A0 A3 E0 E3 A6 A5 AD+

162 24

163 0308 8D A5 A4 AE E1 E2 A0+ noMemMsg DB 'Недостаточно памяти$'

164 E2 AE E7 AD AE 20 AF+

165 A0 AC EF E2 A8 24

166 031C 8D A5 20 E3 A4 A0 AB+ notInstalledMsg DB 'Не удалось загрузить резидент$'

167 AE E1 EC 20 A7 A0 A3+

168 E0 E3 A7 A8 E2 EC 20+

169 E0 A5 A7 A8 A4 A5 AD+

170 E2 24

171

172 033A 90 A5 A7 A8 A4 A5 AD+ removedMsg DB 'Резидент выгружен'

173 E2 20 A2 EB A3 E0 E3+

174 A6 A5 AD

175 =0011 removedMsg\_length equ $-removedMsg

176

177 034B 8D A5 20 E3 A4 A0 AB+ noRemoveMsg DB 'Не удалось выгрузить резидент'

178 AE E1 EC 20 A2 EB A3+

179 E0 E3 A7 A8 E2 EC 20+

180 E0 A5 A7 A8 A4 A5 AD+

181 E2

182 =001D noRemoveMsg\_length equ $-noRemoveMsg

183

184 0368 46 31 f1\_txt DB 'F1'

185 036A 46 32 f2\_txt DB 'F2'

186 036C 46 33 f3\_txt DB 'F3'

187 036E 46 34 f4\_txt DB 'F4'

188 =0002 fx\_length equ $-f4\_txt

189

190 0370 changeFx proc

191 0370 50 push AX

192 0371 53 push BX

193 0372 51 push CX

194 0373 52 push DX

195 0374 55 push BP

196 0375 06 push ES

197 0376 33 DB xor BX, BX

198

199 0378 B4 03 mov AH, 03h

200 037A CD 10 int 10h

201 037C 52 push DX

202

203 037D 0E push CS

204 037E 07 pop ES

205

206 037F \_checkF1:

207 037F BD 0368r lea BP, f1\_txt

208 0382 B9 0002 mov CX, fx\_length

209 0385 B7 00 mov BH, 0

210 0387 B6 00 mov DH, 0

211 0389 B2 4E mov DL, 78

212 038B B8 1301 mov AX, 1301h

213

214 038E 80 3E 0129r FF cmp signaturePrintingEnabled, true

215 0393 74 07 je \_greenF1

216

217 0395 \_redF1:

218 0395 B3 4F mov BL, 01001111b ; red

219 0397 CD 10 int 10h

220 0399 EB 08 90 jmp \_checkF2

221

222 039C \_greenF1:

223 039C BD 0368r lea BP, f1\_txt

224 039F B3 2F mov BL, 00101111b ; green

225 03A1 CD 10 int 10h

226

227 03A3 \_checkF2:

228 03A3 BD 036Ar lea BP, f2\_txt

229 03A6 B9 0002 mov CX, fx\_length

230 03A9 B7 00 mov BH, 0

231 03AB B6 01 mov DH, 1

232 03AD B2 4E mov DL, 78

233 03AF B8 1301 mov AX, 1301h

234

235 03B2 80 3E 012Ar FF cmp cursiveEnabled, true

236 03B7 74 07 je \_greenF2

237

238 03B9 \_redF2:

239 03B9 B3 4F mov BL, 01001111b ; red

240 03BB CD 10 int 10h

241 03BD EB 05 90 jmp \_checkF3

242

243 03C0 \_greenF2:

244 03C0 B3 2F mov BL, 00101111b ; green

245 03C2 CD 10 int 10h

246

247 03C4 \_checkF3:

248 03C4 BD 036Cr lea BP, f3\_txt

249 03C7 B9 0002 mov CX, fx\_length

250 03CA B7 00 mov BH, 0

251 03CC B6 02 mov DH, 2

252 03CE B2 4E mov DL, 78

253 03D0 B8 1301 mov AX, 1301h

254

255 03D3 80 3E 0128r FF cmp translateEnabled, true

256 03D8 74 07 je \_greenF3

257

258 03DA \_redF3:

259 03DA B3 4F mov BL, 01001111b ; red

260 03DC CD 10 int 10h

261 03DE EB 05 90 jmp \_checkF4

262

263 03E1 \_greenF3:

264 03E1 B3 2F mov BL, 00101111b ; green

265 03E3 CD 10 int 10h

266

267 03E5 \_checkF4:

268 03E5 BD 036Er lea BP, f4\_txt

269 03E8 B9 0002 mov CX, fx\_length

270 03EB B7 00 mov BH, 0

271 03ED B6 03 mov DH, 3

272 03EF B2 4E mov DL, 78

273 03F1 B8 1301 mov AX, 1301h

274

275 03F4 80 3E 011Dr FF cmp ignoreEnabled, true

276 03F9 74 07 je \_greenF4

277

278 03FB \_redF4:

279 03FB B3 4F mov BL, 01001111b ; red

280 03FD CD 10 int 10h

281 03FF EB 05 90 jmp \_outFx

282

283 0402 \_greenF4:

284 0402 B3 2F mov BL, 00101111b ; green

285 0404 CD 10 int 10h

286

287 0406 \_outFx:

288 0406 5A pop DX

289 0407 B4 02 mov AH, 02h

290 0409 CD 10 int 10h

291

292 040B 07 pop ES

293 040C 5D pop BP

294 040D 5A pop DX

295 040E 59 pop CX

296 040F 5B pop BX

297 0410 58 pop AX

298 0411 C3 ret

299 0412 changeFx endp

300

301 ;новый обработчик

302 0412 new\_int9h proc far

303 ; сохраняем значения всех, изменяемых регистров в стэке

304 0412 56 push SI

305 0413 50 push AX

306 0414 53 push BX

307 0415 51 push CX

308 0416 52 push DX

309 0417 06 push ES

310 0418 1E push DS

311 ; синхронизируем CS и DS

312 0419 0E push CS

313 041A 1F pop DS

314

315 041B B8 0040 mov AX, 40h ; 40h-сегмент,где хранятся флаги сост-я клавиатуры, кольц. +

316 буфер ввода

317 041E 8E C0 mov ES, AX

318 0420 E4 60 in AL, 60h ; записываем в AL скан-код нажатой клавиши

319

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

321 0422 3C 16 cmp AL, 22 ; была нажата клавиша U?

322 0424 75 24 jne \_test\_Fx

323 0426 26: 8A 26 0017 mov AH, ES:[17h] ; флаги клавиатуры

324 042B 80 E4 0F and AH, 00001111b

325 042E 80 FC 04 cmp AH, 00000100b ; был ли нажат ctrl?

326 0431 75 17 jne \_test\_Fx

327 ; выгрузка

328 0433 B4 FF mov AH, 0FFh

329 0435 B0 01 mov AL, 01h

330 0437 CD 2F int 2Fh

331 ; завершаем обработку нажатия

332

333 0439 E4 61 in AL, 61h ;контроллер состояния клавиатуры

334 043B 0C 80 or AL, 10000000b ;пометим, что клавишу нажали

335 043D E6 61 out 61h, AL

336 043F 24 7F and AL, 01111111b ;пометим, что клавишу отпустили

337 0441 E6 61 out 61h, AL

338 0443 B0 20 mov AL, 20h

339 0445 E6 20 out 20h, AL ;отправим в контроллер прерываний признак конца +

340 прерывания

341

342 ; выходим

343 0447 E9 009D jmp \_quit

344

345 ;@ далее - код для всех вариантов

346

347 ;проверка F1-F4

348 044A \_test\_Fx:

349 044A 2C 3A sub AL, 58 ; в AL теперь номер функциональной клавиши

350 044C \_F1:

351 044C 3C 01 cmp AL, 1 ; F1

352 044E 75 0A jne \_F2

353 0450 F6 16 0129r not signaturePrintingEnabled

354 0454 E8 FF19 call changeFx

355 0457 EB 2E 90 jmp \_translate\_or\_ignore

356 045A \_F2:

357 045A 3C 02 cmp AL, 2 ; F2

358 045C 75 0D jne \_F3

359 045E F6 16 012Ar not cursiveEnabled

360 0462 E8 FF0B call changeFx

361 0465 E8 01EF call setCursive ; перевод символа в курсив и обратно в зависимости от +

362 флага cursiveEnabled

363 0468 EB 1D 90 jmp \_translate\_or\_ignore

364 046B \_F3:

365 046B 3C 03 cmp AL, 3 ; F3

366 046D 75 0A jne \_F4

367 046F F6 16 0128r not translateEnabled

368 0473 E8 FEFA call changeFx

369 0476 EB 0F 90 jmp \_translate\_or\_ignore

370 0479 \_F4:

371 0479 3C 04 cmp AL, 4 ; F4

372 047B 75 0A jne \_translate\_or\_ignore

373 047D F6 16 011Dr not ignoreEnabled

374 0481 E8 FEEC call changeFx

375 0484 EB 01 90 jmp \_translate\_or\_ignore

376

377 ;игнорирование и перевод

378 0487 \_translate\_or\_ignore:

379

380 0487 9C pushf

381 0488 2E: FF 1E 014Cr call dword ptr CS:[old\_int9hOffset] ; вызываем стандартный обработчик прерывания

382 048D B8 0040 mov AX, 40h ; 40h-сегмент,где хранятся флаги сост-я клавы,кольц. +

383 буфер ввода

384 0490 8E C0 mov ES, AX

385 0492 26: 8B 1E 001C mov BX, ES:[1Ch] ; адрес хвоста

386 0497 4B dec BX ; сместимся назад к последнему

387 0498 4B dec BX ; введённому символу

388 0499 83 FB 1E cmp BX, 1Eh ; не вышли ли мы за пределы буфера?

389 049C 73 03 jae \_go

390 049E BB 003C mov BX, 3Ch ; хвост вышел за пределы буфера, значит последний введённый +

391 символ

392 ; находится в конце буфера

393

394 04A1 \_go:

395 04A1 26: 8B 17 mov DX, ES:[BX] ; в DX 0 введённый символ

396 ;включен ли режим блокировки ввода?

397 04A4 80 3E 011Dr FF cmp ignoreEnabled, true

398 04A9 75 1A jne \_check\_translate

399

400 ; да, включен

401 04AB BE 0000 mov SI, 0

402 04AE B9 001A mov CX, ignoredLength ;кол-во игнорируемых символов

403

404 ; проверяем, присутствует ли текущий символ в списке игнорируемых

405 04B1 \_check\_ignored:

406 04B1 3A 94 0103r cmp DL,ignoredChars[SI]

407 04B5 74 06 je \_block

408 04B7 46 inc SI

409 04B8 E2 F7 loop \_check\_ignored

410 04BA EB 09 90 jmp \_check\_translate

411

412 ; блокируем

413 04BD \_block:

414 04BD 26: 89 1E 001C mov ES:[1Ch], BX ;блокировка ввода символа

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

416 ;@ а заменять одни символы другими,

417 ;@ замените строку выше строкой

418 ;@ mov ES:[BX], AX

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

420 на звёздочки

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

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

423 ;@ и раскомментировать строки ниже:

424 ;@ xor AX, AX

425 ;@ mov AL, replaceWith[SI]

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

427 04C2 EB 23 90 jmp \_quit

428

429 04C5 \_check\_translate:

430 ; включен ли режим перевода?

431 04C5 80 3E 0128r FF cmp translateEnabled, true

432 04CA 75 1B jne \_quit

433

434 ; да, включен

435 04CC BE 0000 mov SI, 0

436 04CF B9 0005 mov CX, translateLength ; кол-во символов для перевода

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

438 04D2 \_check\_translate\_loop:

439 04D2 3A 94 011Er cmp DL, translateFrom[SI]

440 04D6 74 06 je \_translate

441 04D8 46 inc SI

442 04D9 E2 F7 loop \_check\_translate\_loop

443 04DB EB 0A 90 jmp \_quit

444

445 ; переводим

446 04DE \_translate:

447 04DE 33 C0 xor AX, AX

448 04E0 8A 84 0123r mov AL, translateTo[SI]

449 04E4 26: 89 07 mov ES:[BX], AX ; замена символа

450

451 04E7 \_quit:

452 ; восстанавливаем все регистры

453 04E7 1F pop DS

454 04E8 07 pop ES

455 04E9 5A pop DX

456 04EA 59 pop CX

457 04EB 5B pop BX

458 04EC 58 pop AX

459 04ED 5E pop SI

460 04EE CF iret

461 04EF new\_int9h endp

462

463 ;=== Обработчик прерывания int 1Ch ===;

464 ;=== Вызывается каждые 55 мс ===;

465 04EF new\_int1Ch proc far

466 04EF 50 push AX

467 04F0 0E push CS

468 04F1 1F pop DS

469

470 04F2 9C pushf

471 04F3 2E: FF 1E 0150r call dword ptr CS:[old\_int1ChOffset]

472

473 04F8 80 3E 0129r FF cmp signaturePrintingEnabled, true ; если нажата управляющая клавиша (в данном случае +

474 F1)

475 04FD 75 1C jne \_notToPrint

476

477 04FF 83 3E 015Ar 25 cmp counter, printDelay\*1000/55 + 1 ; если кол-во "тактов" эквивалентно +

478 %printDelay% секундам

479 0504 74 03 je \_letsPrint

480

481 0506 EB 0E 90 jmp \_dontPrint

482

483 0509 \_letsPrint:

484 0509 F6 16 0129r not signaturePrintingEnabled

485 050D C7 06 015Ar 0000 mov counter, 0

486 0513 E8 0094 call printSignature

487

488 0516 \_dontPrint:

489 0516 83 06 015Ar 01 add counter, 1

490

491 051B \_notToPrint:

492

493 051B 58 pop AX

494

495 051C CF iret

496 051D new\_int1Ch endp

497

498 ;=== Обработчик прерывания int 2Fh ===;

499 ;=== Служит для:

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

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

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

503 ;===

504 051D new\_int2Fh proc

505 051D 80 FC FF cmp AH, 0FFh ;наша функция?

506 0520 75 0B jne \_2Fh\_std ;нет - на старый обработчик

507 0522 3C 00 cmp AL, 0 ;подфункция проверки, загружен ли резидент в память?

508 0524 74 0C je \_already\_installed

509 0526 3C 01 cmp AL, 1 ;подфункция выгрузки из памяти?

510 0528 74 0B je \_uninstall

511 052A EB 01 90 jmp \_2Fh\_std ;нет - на старый обработчик

512

513 052D \_2Fh\_std:

514 052D 2E: FF 2E 0154r jmp dword ptr CS:[old\_int2FhOffset] ;вызов старого обработчика

515

516 0532 \_already\_installed:

517 0532 B4 69 mov AH, 'i' ;вернём 'i', если резидент загружен в память

518 0534 CF iret

519

520 0535 \_uninstall:

521 0535 1E push DS

522 0536 06 push ES

523 0537 52 push DX

524 0538 53 push BX

525

526 0539 33 DB xor BX, BX

527

528 ; CS = ES, для доступа к переменным

529 053B 0E push CS

530 053C 07 pop ES

531

532 053D B8 2509 mov AX, 2509h

533 0540 26: 8B 16 014Cr mov DX, ES:old\_int9hOffset ; возвращаем вектор прерывания

534 0545 26: 8E 1E 014Er mov DS, ES:old\_int9hSegment ; на место

535 054A CD 21 int 21h

536

537 054C B8 251C mov AX, 251Ch

538 054F 26: 8B 16 0150r mov DX, ES:old\_int1ChOffset ; возвращаем вектор прерывания

539 0554 26: 8E 1E 0152r mov DS, ES:old\_int1ChSegment ; на место

540 0559 CD 21 int 21h

541

542 055B B8 252F mov AX, 252Fh

543 055E 26: 8B 16 0154r mov DX, ES:old\_int2FhOffset ; возвращаем вектор прерывания

544 0563 26: 8E 1E 0156r mov DS, ES:old\_int2FhSegment ; на место

545 0568 CD 21 int 21h

546

547 056A 2E: 8E 06 002C mov ES, CS:2Ch ; загрузим в ES адрес окружения

548 056F B4 49 mov AH, 49h ; выгрузим из памяти окружение

549 0571 CD 21 int 21h

550 0573 72 0B jc \_notRemove

551

552 0575 0E push CS

553 0576 07 pop ES ;в ES - адрес резидентной программы

554 0577 B4 49 mov AH, 49h ;выгрузим из памяти резидент

555 0579 CD 21 int 21h

556 057B 72 03 jc \_notRemove

557 057D EB 15 90 jmp \_unloaded

558

559 0580 \_notRemove: ; не удалось выполнить выгрузку

560 ; вывод сообщения о неудачной выгрузке

561 0580 B4 03 mov AH, 03h ; получаем позицию курсора

562 0582 CD 10 int 10h

563 0584 BD 034Br lea BP, noRemoveMsg

564 0587 B9 001D mov CX, noRemoveMsg\_length

565 058A B3 07 mov BL, 0111b

566 058C B8 1301 mov AX, 1301h

567 058F CD 10 int 10h

568 0591 EB 12 90 jmp \_2Fh\_exit

569

570 0594 \_unloaded: ; выгрузка прошла успешно

571 ; вывод сообщения об удачной выгрузке

572 0594 B4 03 mov AH, 03h ; получаем позицию курсора

573 0596 CD 10 int 10h

574 0598 BD 033Ar lea BP, removedMsg

575 059B B9 0011 mov CX, removedMsg\_length

576 059E B3 07 mov BL, 0111b

577 05A0 B8 1301 mov AX, 1301h

578 05A3 CD 10 int 10h

579

580 05A5 \_2Fh\_exit:

581 05A5 5B pop BX

582 05A6 5A pop DX

583 05A7 07 pop ES

584 05A8 1F pop DS

585 05A9 CF iret

586 05AA new\_int2Fh endp

587

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

589 ;=== Настраивается значениями переменных в начале исходника

590 ;===

591 05AA printSignature proc

592 05AA 50 push AX

593 05AB 52 push DX

594 05AC 51 push CX

595 05AD 53 push BX

596 05AE 06 push ES

597 05AF 54 push SP

598 05B0 55 push BP

599 05B1 56 push SI

600 05B2 57 push DI

601

602 05B3 33 C0 xor AX, AX

603 05B5 33 DB xor BX, BX

604 05B7 33 D2 xor DX, DX

605

606 05B9 B4 03 mov AH, 03h ;чтение текущей позиции курсора

607 05BB CD 10 int 10h

608 05BD 52 push DX ;помещаем информацию о +

609 положении курсора в стек

610

611 05BE 83 3E 015Cr 00 cmp printPos, 0

612 05C3 74 0E je \_printTop

613

614 05C5 83 3E 015Cr 01 cmp printPos, 1

615 05CA 74 0E je \_printCenter

616

617 05CC 83 3E 015Cr 02 cmp printPos, 2

618 05D1 74 0E je \_printBottom

619

620 ;все числа подобраны на глаз...

621 05D3 \_printTop:

622 05D3 B6 00 mov DH, 0

623 05D5 B2 0F mov DL, 15

624 05D7 EB 0F 90 jmp \_actualPrint

625

626 05DA \_printCenter:

627 05DA B6 09 mov DH, 9

628 05DC B2 0F mov DL, 15

629 05DE EB 08 90 jmp \_actualPrint

630

631 05E1 \_printBottom:

632 05E1 B6 13 mov DH, 19

633 05E3 B2 0F mov DL, 15

634 05E5 EB 01 90 jmp \_actualPrint

635

636 05E8 \_actualPrint:

637 05E8 B4 0F mov AH, 0Fh ;чтение текущего видеорежима. в+

638 BH - текущая страница

639 05EA CD 10 int 10h

640

641 05EC 0E push CS

642 05ED 07 pop ES ;указываем ES на CS

643

644 ;вывод 'верхушки' таблицы

645 05EE 52 push DX

646 05EF BD 0277r lea BP, tableTop ;помещаем в BP указатель на +

647 выводимую строку

648 05F2 B9 0034 mov CX, tableTop\_length ;в CX - длина строки

649 05F5 B3 07 mov BL, 0111b ;цвет выводимого текста ref: +

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

651 05F7 B8 1301 mov AX, 1301h ;AH=13h - номер ф-ии, AL=01h - +

652 курсор перемещается при выводе каждого из символов строки

653 05FA CD 10 int 10h

654 05FC 5A pop DX

655 05FD FE C6 inc DH

656

657

658 ;вывод первой линии

659 05FF 52 push DX

660 0600 BD 015Er lea BP, signatureLine1

661 0603 B9 0034 mov CX, Line1\_length

662 0606 B3 07 mov BL, 0111b

663 0608 B8 1301 mov AX, 1301h

664 060B CD 10 int 10h

665 060D 5A pop DX

666 060E FE C6 inc DH

667

668 ;вывод второй линии

669 0610 52 push DX

670 0611 BD 0192r lea BP, signatureLine2

671 0614 B9 0034 mov CX, Line2\_length

672 0617 B3 07 mov BL, 0111b

673 0619 B8 1301 mov AX, 1301h

674 061C CD 10 int 10h

675 061E 5A pop DX

676 061F FE C6 inc DH

677

678 ;вывод третьей линии

679 0621 52 push DX

680 0622 BD 01C6r lea BP, signatureLine3

681 0625 B9 0034 mov CX, Line3\_length

682 0628 B3 07 mov BL, 0111b

683 062A B8 1301 mov AX, 1301h

684 062D CD 10 int 10h

685 062F 5A pop DX

686 0630 FE C6 inc DH

687

688 ;вывод 'низа' таблицы

689 0632 52 push DX

690 0633 BD 02ABr lea BP, tableBottom

691 0636 B9 0034 mov CX, tableBottom\_length

692 0639 B3 07 mov BL, 0111b

693 063B B8 1301 mov AX, 1301h

694 063E CD 10 int 10h

695 0640 5A pop DX

696 0641 FE C6 inc DH

697

698 0643 33 DB xor BX, BX

699 0645 5A pop DX ;восстанавливаем из стека +

700 прежнее положение курсора

701 0646 B4 02 mov AH, 02h ;меняем положение курсора на +

702 первоначальное

703 0648 CD 10 int 10h

704 064A E8 FD23 call changeFx

705

706 064D 5F pop DI

707 064E 5E pop SI

708 064F 5D pop BP

709 0650 5C pop SP

710 0651 07 pop ES

711 0652 5B pop BX

712 0653 59 pop CX

713 0654 5A pop DX

714 0655 58 pop AX

715

716 0656 C3 ret

717 0657 printSignature endp

718

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

720 на обычное и наоброт

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

722 0657 setCursive proc

723 0657 06 push ES ; сохраняем регистры

724 0658 50 push AX

725 0659 0E push CS

726 065A 07 pop ES

727

728 065B 80 3E 012Ar FF cmp cursiveEnabled, true

729 0660 75 30 jne \_restoreSymbol

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

731 ; предварительно сохраняя старый символ в savedSymbol

732

733 0662 E8 004C call saveFont

734 0665 8A 0E 013Br mov CL, charToCursiveIndex

735 0669 \_shifTtable:

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

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

738 0669 83 C5 10 add BP, 16

739 066C E2 FB loop \_shiftTable

740

741 ; пpи savefont смещается pегистp ES

742 ; поэтомy пpиходится делать такие махинации, чтобы

743 ; записать полyченный элемент в savedSymbol

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

745 066E 1E push DS

746 066F 58 pop AX

747 0670 06 push ES

748 0671 1F pop DS

749 0672 50 push AX

750 0673 07 pop ES

751 0674 50 push AX

752

753 0675 8B F5 mov SI, BP

754 0677 BF 013Cr lea DI, savedSymbol

755 ; сохpаняем в пеpеменнyю savedSymbol

756 ; таблицy нyжного символа

757 067A B9 0010 mov CX, 16

758 ; movsb из DS:SI в ES:DI

759 067D F3> A4 rep movsb

760 ; исходные позиции сегментов возвpащены

761 067F 1F pop DS ; восстановление DS

762

763 ; заменим написание символа на кypсив

764 0680 B9 0001 mov CX, 1

765 0683 B6 00 mov DH, 0

766 0685 8A 16 013Br mov DL, charToCursiveIndex

767 0689 BD 012Br lea BP, cursiveSymbol

768 068C E8 0015 call changeFont

769 068F EB 10 90 jmp \_exitSetCursive

770

771 0692 \_restoreSymbol:

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

773

774 0692 B9 0001 mov CX, 1

775 0695 B6 00 mov DH, 0

776 0697 8A 16 013Br mov DL, charToCursiveIndex

777 069B BD 013Cr lea bp, savedSymbol

778 069E E8 0003 call changeFont

779

780 06A1 \_exitSetCursive:

781 06A1 58 pop AX

782 06A2 07 pop ES

783 06A3 C3 ret

784 06A4 setCursive endp

785

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

787 ;===

788 ; \*\*\* входные данные

789 ; DL = номер символа для замены

790 ; CX = Кол-во символов заменяемых изображений символов

791 ; (начиная с символа указанного в DX)

792 ; ES:bp = адрес таблицы

793 ;

794 ; \*\*\* описание работы процедуры

795 ; Происходит вызов int 10h (видеосервис)

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

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

798 ; символа для текущего шрифта

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

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

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

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

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

804 ;

805 ; \*\*\* результат

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

807 ; на предложенное пользователем.

808 ; Изменению подвергнутся все символы, находящиеся на экране,

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

810

811 06A4 changeFont proc

812 06A4 50 push AX

813 06A5 53 push BX

814 06A6 B8 1100 mov AX, 1100h

815 06A9 BB 1000 mov BX, 1000h

816 06AC CD 10 int 10h

817 06AE 58 pop AX

818 06AF 5B pop BX

819 06B0 C3 ret

820 06B1 changeFont endp

821

822 ;=== Функция сохранения нормального начертания символа

823 ;===

824 ; \*\*\* входные данные

825 ; BH - тип возвращаемой символьной таблицы

826 ; 0 - таблица из int 1fh

827 ; 1 - таблица из int 44h

828 ; 2-5 - таблица из 8x14, 8x8, 8x8 (top), 9x14

829 ; 6 - 8x16

830 ;

831 ; \*\*\* описание работы процедуры

832 ; Происходит вызов int 10h (видеосервис)

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

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

835 ;

836 ; \*\*\* результат

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

838 ; в CX находится байт на символ

839 ; в DL количество экранных строк

840 ; ВАЖНО! Происходит сдвиг регистра ES

841 ; ( ES становится равным C000h )

842

843 06B1 saveFont proc

844 06B1 50 push AX

845 06B2 53 push BX

846 06B3 B8 1130 mov AX, 1130h

847 06B6 BB 0600 mov BX, 0600h

848 06B9 CD 10 int 10h

849 06BB 58 pop AX

850 06BC 5B pop BX

851 06BD C3 ret

852 06BE saveFont endp

853

854

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

856 ;===

857 06BE \_initTSR: ; старт резидента

858 06BE B4 03 mov AH, 03h

859 06C0 CD 10 int 10h

860 06C2 52 push DX

861 06C3 B4 00 mov AH,00h ; установка видеорежима (83h текст +

862 80x25 16/8 CGA,EGA b800 Comp,RGB,Enhanced), без очистки экрана

863 06C5 B0 83 mov AL,83h

864 06C7 CD 10 int 10h

865 06C9 5A pop DX

866 06CA B4 02 mov AH, 02h

867 06CC CD 10 int 10h

868

869

870 06CE E8 00B3 call commandParamsParser

871 06D1 B8 3509 mov AX,3509h ; получить в ES:BX вектор 09

872 06D4 CD 21 int 21h ; прерывания

873

874 ;@ === Удаление резидента из памяти ===

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

876 ;@ нужно закомментировать следующие 3 строки, а также

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

878 06D6 80 3E 0158r FF cmp unloadTSR, true

879 06DB 74 03 je \_removingOnParameter

880 06DD EB 15 90 jmp \_notRemovingNow

881

882 06E0 \_removingOnParameter:

883 06E0 B4 FF mov AH, 0FFh

884 06E2 B0 00 mov AL, 0

885 06E4 CD 2F int 2Fh

886 06E6 80 FC 69 cmp AH, 'i' ; проверка того, загружена ли уже программа

887 06E9 74 7D je \_remove

888 06EB B4 09 mov AH, 09h ;@ для выгрузки резидента по повторному+

889 запуску закомментировать эту строку

890 06ED BA 031Cr lea DX, notInstalledMsg ;@ для выгрузки резидента по повторному запуску +

891 закомментировать эту строку

892 06F0 CD 21 int 21h ;@ для выгрузки резидента по повторному+

893 запуску закомментировать эту строку

894 06F2 CD 20 int 20h ;@ для выгрузки резидента по повторному+

895 запуску закомментировать эту строку

896

897 06F4 \_notRemovingNow:

898

899 06F4 80 3E 0159r FF cmp notLoadTSR, true ; если была выведена справка

900 06F9 74 0E je \_exit\_tmp ; просто выходим

901

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

903 комментируем 5 строк ниже

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

905 06FB B4 FF mov AH, 0FFh

906 06FD B0 00 mov AL, 0

907 06FF CD 2F int 2Fh

908 0701 80 FC 69 cmp AH, 'i' ; проверка того, загружена ли уже программа

909 0704 74 6B je \_alreadyInstalled

910

911 0706 EB 04 90 jmp \_tmp

912

913 0709 \_exit\_tmp:

914 0709 EB 77 90 jmp \_exit

915

916 070C \_tmp:

917 070C 06 push ES

918 070D A1 002C mov AX, DS:[2Ch] ; psp

919 0710 8E C0 mov ES, AX

920 0712 B4 49 mov AH, 49h ; хватит памяти чтоб остаться

921 0714 CD 21 int 21h ; резидентом?

922 0716 07 pop ES

923 0717 72 62 jc \_notMem ; не хватило - выходим

924

925 ;== int 09h ==;

926

927 0719 2E: 89 1E 014Cr mov word ptr CS:old\_int9hOffset, BX

928 071E 2E: 8C 06 014Er mov word ptr CS:old\_int9hSegment, ES

929 0723 B8 2509 mov AX, 2509h ; установим вектор на 09

930 0726 BA 0412r mov DX, offset new\_int9h ; прерывание

931 0729 CD 21 int 21h

932

933 ;== int 1Ch ==;

934 072B B8 351C mov AX,351Ch ; получить в ES:BX вектор 1C

935 072E CD 21 int 21h ; прерывания

936 0730 2E: 89 1E 0150r mov word ptr CS:old\_int1ChOffset, BX

937 0735 2E: 8C 06 0152r mov word ptr CS:old\_int1ChSegment, ES

938 073A B8 251C mov AX, 251Ch ; установим вектор на 1C

939 073D BA 04EFr mov DX, offset new\_int1Ch ; прерывание

940 0740 CD 21 int 21h

941

942 ;== int 2Fh ==;

943 0742 B8 352F mov AX,352Fh ; получить в ES:BX вектор 1C

944 0745 CD 21 int 21h ; прерывания

945 0747 2E: 89 1E 0154r mov word ptr CS:old\_int2FhOffset, BX

946 074C 2E: 8C 06 0156r mov word ptr CS:old\_int2FhSegment, ES

947 0751 B8 252F mov AX, 252Fh ; установим вектор на 2F

948 0754 BA 051Dr mov DX, offset new\_int2Fh ; прерывание

949 0757 CD 21 int 21h

950

951 0759 E8 FC14 call changeFx

952 075C BA 02DFr mov DX, offset installedMsg ; выводим что все ок

953 075F B4 09 mov AH, 9

954 0761 CD 21 int 21h

955 0763 BA 06BEr mov DX, offset \_initTSR ; остаемся в памяти резидентом

956 0766 CD 27 int 27h ; и выходим

957 ; конец основной программы

958 0768 \_remove: ; выгрузка программы из памяти

959 0768 B4 FF mov AH, 0FFh

960 076A B0 01 mov AL, 1

961 076C CD 2F int 2Fh

962 076E EB 12 90 jmp \_exit

963 0771 \_alreadyInstalled:

964 0771 B4 09 mov AH, 09h

965 0773 BA 02F2r lea DX, alreadyInstalledMsg

966 0776 CD 21 int 21h

967 0778 EB 08 90 jmp \_exit

968 077B \_notMem: ; не хватает памяти, чтобы остаться резидентом

969 077B BA 0308r mov DX, offset noMemMsg

970 077E B4 09 mov AH, 9

971 0780 CD 21 int 21h

972 0782 \_exit: ; выход

973 0782 CD 20 int 20h

974

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

976 ;===

977 0784 commandParamsParser proc

978 0784 0E push CS

979 0785 07 pop ES

980 0786 C6 06 0158r 00 mov unloadTSR, 0

981 078B C6 06 0159r 00 mov notLoadTSR, 0

982

983 0790 BE 0080 mov SI, 80h ;SI=смещение командной строки.

984 0793 AC lodsb ;Получим кол-во символов.

985 0794 0A C0 or AL, AL ;Если 0 символов введено,

986 0796 74 45 jz \_exitHelp ;то все в порядке.

987

988 0798 \_nextChar:

989

990 0798 46 inc SI ;Теперь SI указывает на первый символ +

991 строки.

992

993 0799 80 3C 0D cmp [SI], BYTE ptr 13

994 079C 74 3F je \_exitHelp

995

996

997 079E AD lodsw ;Получаем два символа

998 079F 3D 3F2F cmp AX, '?/' ;Это '/?' (данные расположены в +

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

1000 07A2 74 08 je \_question

1001 07A4 3D 752F cmp AX, 'u/'

1002 07A7 74 1A je \_finishTSR

1003

1004 ;cmp AL, '/'

1005 ;je \_errorParam

1006

1007 07A9 EB 32 90 jmp \_exitHelp

1008

1009 07AC \_question:

1010 ; вывод строки помощи

1011 07AC B4 03 mov AH,03

1012 07AE CD 10 int 10h

1013 07B0 BD 01FAr lea BP, helpMsg

1014 07B3 B9 0058 mov CX, helpMsg\_length

1015 07B6 B3 07 mov BL, 0111b

1016 07B8 B8 1301 mov AX, 1301h

1017 07BB CD 10 int 10h

1018 ; конец вывода строки помощи

1019 07BD F6 16 0159r not notLoadTSR ;флаг того, что необходимо не загружать резидент

1020 07C1 EB D5 jmp \_nextChar

1021

1022 ;@ === Удаление резидента из памяти ===

1023 ;@ Если по варианту необходимо выгружать резидент по параметру '/u' коммандной строки,

1024 ;@ нужно использовать следующий код, в остальных случаях необходимо закомменитровать

1025 ;@ этот код, кроме названия метки! (по желанию можно избавиться и от метки, но +

1026 аккуратно просмотреть использование)

1027 07C3 \_finishTSR:

1028 07C3 F6 16 0158r not unloadTSR ;флаг того, что необходимо выгузить резидент

1029 07C7 EB CF jmp \_nextChar

1030

1031 07C9 EB 12 90 jmp \_exitHelp

1032

1033 07CC \_errorParam:

1034 ;вывод строки

1035 07CC B4 03 mov AH,03

1036 07CE CD 10 int 10h

1037 07D0 BD 0252r lea BP, CS:errorParamMsg

1038 07D3 B9 0025 mov CX, errorParamMsg\_length

1039 07D6 B3 07 mov BL, 0111b

1040 07D8 B8 1301 mov AX, 1301h

1041 07DB CD 10 int 10h

1042 ;конец вывода строки

1043 07DD \_exitHelp:

1044 07DD C3 ret

1045 07DE commandParamsParser endp

1046

1047 07DE code ends

1048 end \_start

Symbol Table

Symbol Name Type Value

??DATE Text "04/28/13"

??FILENAME Text "tsr "

??TIME Text "18:37:04"

??VERSION Number 030A

@CPU Text 0101H

@CURSEG Text CODE

@FILENAME Text TSR

@WORDSIZE Text 2

ALREADYINSTALLEDMSG Byte CODE:02F2

CHANGEFONT Near CODE:06A4

CHANGEFX Near CODE:0370

CHARTOCURSIVEINDEX Byte CODE:013B

COMMANDPARAMSHANDLER Near CODE:0784

COUNTER Word CODE:015A

CURSIVEENABLED Byte CODE:012A

CURSIVESYMBOL Byte CODE:012B

ERRORPARAMMSG Byte CODE:0252

ERRORPARAMMSG\_LENGTH Number 0025

F1\_TXT Byte CODE:0368

F2\_TXT Byte CODE:036A

F3\_TXT Byte CODE:036C

F4\_TXT Byte CODE:036E

FX\_LENGTH Number 0002

HELPMSG Byte CODE:01FA

HELPMSG\_LENGTH Number 0058

IGNOREDCHARS Byte CODE:0103

IGNOREDLENGTH Number 001A

IGNOREENABLED Byte CODE:011D

INSTALLEDMSG Byte CODE:02DF

LINE1\_LENGTH Number 0034

LINE2\_LENGTH Number 0034

LINE3\_LENGTH Number 0034

NEW\_INT1CH Far CODE:04EF

NEW\_INT2FH Near CODE:051D

NEW\_INT9H Far CODE:0412

NOMEMMSG Byte CODE:0308

NOREMOVEMSG Byte CODE:034B

NOREMOVEMSG\_LENGTH Number 001D

NOTINSTALLEDMSG Byte CODE:031C

NOTLOADTSR Byte CODE:0159

OLD\_INT1CHOFFSET Word CODE:0150

OLD\_INT1CHSEGMENT Word CODE:0152

OLD\_INT2FHOFFSET Word CODE:0154

OLD\_INT2FHSEGMENT Word CODE:0156

OLD\_INT9HOFFSET Word CODE:014C

OLD\_INT9HSEGMENT Word CODE:014E

PRINTDELAY Number 0002

PRINTPOS Word CODE:015C

PRINTSIGNATURE Near CODE:05AA

REMOVEDMSG Byte CODE:033A

REMOVEDMSG\_LENGTH Number 0011

SAVEDSYMBOL Byte CODE:013C

SAVEFONT Near CODE:06B1

SETCURSIVE Near CODE:0657

SIGNATURELINE1 Byte CODE:015E

SIGNATURELINE2 Byte CODE:0192

SIGNATURELINE3 Byte CODE:01C6

SIGNATUREPRINTINGENABLED Byte CODE:0129

TABLEBOTTOM Byte CODE:02AB

TABLEBOTTOM\_LENGTH Number 0034

TABLETOP Byte CODE:0277

TABLETOP\_LENGTH Number 0034

TRANSLATEENABLED Byte CODE:0128

TRANSLATEFROM Byte CODE:011E

TRANSLATELENGTH Number 0005

TRANSLATETO Byte CODE:0123

TRUE Number 00FF

UNLOADTSR Byte CODE:0158

\_2FH\_EXIT Near CODE:05A5

\_2FH\_STD Near CODE:052D

\_ACTUALPRINT Near CODE:05E8

\_ALREADYINSTALLED Near CODE:0771

\_ALREADY\_INSTALLED Near CODE:0532

\_BLOCK Near CODE:04BD

\_CHECKF1 Near CODE:037F

\_CHECKF2 Near CODE:03A3

\_CHECKF3 Near CODE:03C4

\_CHECKF4 Near CODE:03E5

\_CHECK\_IGNORED Near CODE:04B1

\_CHECK\_TRANSLATE Near CODE:04C5

\_CHECK\_TRANSLATE\_LOOP Near CODE:04D2

\_DONTPRINT Near CODE:0516

\_ERRORPARAM Near CODE:07CC

\_EXIT Near CODE:0782

\_EXITHELP Near CODE:07DD

\_EXITSETCURSIVE Near CODE:06A1

\_EXIT\_TMP Near CODE:0709

\_F1 Near CODE:044C

\_F2 Near CODE:045A

\_F3 Near CODE:046B

\_F4 Near CODE:0479

\_FINISHTSR Near CODE:07C3

\_GO Near CODE:04A1

\_GREENF1 Near CODE:039C

\_GREENF2 Near CODE:03C0

\_GREENF3 Near CODE:03E1

\_GREENF4 Near CODE:0402

\_INITTSR Near CODE:06BE

\_LETSPRINT Near CODE:0509

\_NEXTCHAR Near CODE:0798

\_NOTMEM Near CODE:077B

\_NOTREMOVE Near CODE:0580

\_NOTREMOVINGNOW Near CODE:06F4

\_NOTTOPRINT Near CODE:051B

\_OUTFX Near CODE:0406

\_PRINTBOTTOM Near CODE:05E1

\_PRINTCENTER Near CODE:05DA

\_PRINTTOP Near CODE:05D3

\_QUESTION Near CODE:07AC

\_QUIT Near CODE:04E7

\_REDF1 Near CODE:0395

\_REDF2 Near CODE:03B9

\_REDF3 Near CODE:03DA

\_REDF4 Near CODE:03FB

\_REMOVE Near CODE:0768

\_REMOVINGONPARAMETER Near CODE:06E0

\_RESTORESYMBOL Near CODE:0692

\_SHIFTTABLE Near CODE:0669

\_START Near CODE:0100

\_TEST\_FX Near CODE:044A

\_TMP Near CODE:070C

\_TRANSLATE Near CODE:04DE

\_TRANSLATE\_OR\_IGNORE Near CODE:0487

\_UNINSTALL Near CODE:0535

\_UNLOADED Near CODE:0594

Groups & Segments Bit Size Align Combine Class

CODE 16 07DE Para none CODE

1. **Файл unloader.lst**

Turbo Assembler Version 3.1

unloader.asm

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

2 ; unloader.asm

3 ;

4 ; Сборка:

5 ; tasm.exe /l unloader.asm

6 ; tlink /t /x unloader.obj

7 ;

8 ; Программа для выгрузки TSR из памяти

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

10

11 0000 code segment 'code'

12 assume CS:code, DS:code

13 org 100h

14 0100 \_start:

15

16 0100 B4 FF mov AH, 0FFh

17 0102 B0 01 mov AL, 1

18 0104 CD 2F int 2Fh ; наше прерывание

19 0106 CD 20 int 20h ; выходим

20

21 0108 code ends

22 end \_start

Symbol Table

Symbol Name Type Value

??DATE Text "04/28/13"

??FILENAME Text "unloader"

??TIME Text "19:37:59"

??VERSION Number 030A

@CPU Text 0101H

@CURSEG Text CODE

@FILENAME Text UNLOADER

@WORDSIZE Text 2

\_START Near CODE:0100

Groups & Segments Bit Size Align Combine Class

CODE 16 0108 Para none CODE