

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Операционные системы»
Тема: Исследование структур загрузочных модулей

Студент гр. 9382

Дерюгин Д.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование различий в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

Задание.

Шаг 1. Напишите текст исходного .COM модуля, который определяет тип РС и версию системы. Это довольно простая задача и для тех, кто уже имеет опыт программирования на ассемблере, это будет небольшой разминкой. Для тех, кто раньше не сталкивался с программированием на ассемблере, это неплохая задача для первого опыта.

За основу возьмите шаблон, приведенный в разделе «Основные сведения». Необходимые сведения о том, как извлечь требуемую информацию, представлены в следующем разделе.

Ассемблерная программа должна читать содержимое предпоследнего байта ROM BIOS, по таблице, сравнивая коды, определять тип РС и выводить строку с названием модели. Если код не совпадает ни с одним значением, то двоичный код переводиться в символьную строку, содержащую запись шестнадцатеричного числа и выводиться на экран в виде соответствующего сообщения.

Затем определяется версия системы. Ассемблерная программа должна по значениям регистров AL и AH формировать текстовую строку в формате xx.yy, где xx - номер основной версии, а yy - номер модификации в десятичной системе счисления, формировать строки с серийным номером OEM и серийным номером пользователя. Полученные строки выводятся на экран.

Отладьте полученный исходный модуль.

Результатом выполнения этого шага будет «хороший» .COM модуль, а также необходимо построить «плохой» .EXE, полученный из исходного текста для .COM модуля

Шаг 2. Напишите текст исходного .EXE модуля, который выполняет те же функции, что и модуль в Шаге 1 и постройте и отладьте его. Таким образом, будет получен «хороший» .EXE.

Шаг 3. Сравните исходные тексты для .COM и .EXE модулей. Ответьте на контрольные вопросы «Отличия исходных текстов COM и EXE программ».

Шаг 4. Запустите FAR и откройте (F3/F4) файл загрузочного модуля .COM файл «плохого» .EXE в шестнадцатеричном виде. Затем откройте (F3/F4) файл загрузочного модуля «хорошего» .EXE и сравните его с предыдущими файлами.

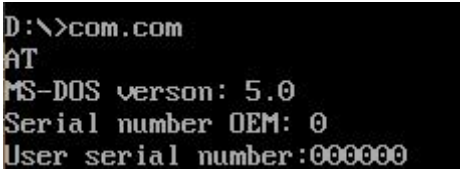
Ответьте на контрольные вопросы «Отличия форматов файлов COM и EXE модулей».

Шаг 5. Откройте отладчик TD.EXE и загрузите .COM. Ответьте на контрольные вопросы «Загрузка COM модуля в основную память». Представьте в отчете план загрузки модуля .COM в основную память.

Шаг 6. Откройте отладчик TD.EXE и загрузите «хороший» .EXE. Ответьте на контрольные вопросы «Загрузка «хорошего» EXE модуля в основную память».

Шаг 7. Оформление отчета в соответствии с требованиями. В отчете необходимо привести скриншоты. Для файлов их вид в шестнадцатеричном виде, для загрузочных модулей – в отладчике.

Выполнение работы.



```
D:\>com.com
AT
MS-DOS version: 5.0
Serial number OEM: 0
User serial number:000000
```

Рис. 1 Результат работы “хорошего” .com модуля

```
D:\>com.exe

0+PC
5 0
0+PC
0
000000
0+PC
0+PC
D:\>
```

Рис. 2 Результат работы «плохого» .exe модуля

```
D:\>exe.exe
AT
MS-DOS version: 5.0
Serial number OEM: 0
User serial number:000000
D:\>1
```

Рис.3 Результат работы «хорошего» .exe модуля

A...	Hexadecimal (1 Byte)	Text (As
0000	4D 5A D4 00 03 00 00 00 20 00 00 00 FF FF 00 00	M Z .
0010	00 00 5B E7 00 01 00 00 1E 00 00 00 01 00 00 00	. . [
0020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
00A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
00B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
00C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
00D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
00E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
00F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
01A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
01B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
01C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
01D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
01E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
01F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0230	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0260	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0270	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0280	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0290	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
02A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
02B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
02C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
02D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
02E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
02F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	. . .
0300	E9 C5 01 50 43 0D 0A 24 43 2F 58 54 0D 0A 24 41	. . . F
0310	54 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 33 30	T . . \$
0320	0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 35 30 20	. . \$ F
0330	6F 72 20 36 30 0D 0A 24 50 53 32 20 6D 6F 64 65	o r 6
0340	6C 20 38 30 0D 0A 24 50 D0 A1 6A 72 0D 0A 24 50 1	8 0
0350	43 20 43 6F 6E 76 65 72 74 69 62 6C 65 0D 0A 24	C C c
0360	4D 53 2D 44 4F 53 20 76 65 72 73 6F 6E 3A 20 20	M S - I
0370	2E 20 20 0D 0A 24 53 65 72 69 61 6C 20 6E 75 6D	.
0380	62 65 72 20 4F 45 4D 3A 20 20 0D 0A 24 55 73 65	b e r
0390	72 20 73 65 72 69 61 6C 20 6E 75 6D 62 65 72 3A	r s e
03A0	20 20 20 20 20 0D 0A 24 20 0D 0A 24 24 0F 3C 09	
03B0	76 02 04 07 04 30 C3 51 8A E0 E8 EF FF 86 C4 B1	v . . .
03C0	04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 E9 FF 88 25
03D0	4F 88 05 4F 8A C7 E8 DE FF 88 25 4F 88 05 5B C3	O . . C
03E0	51 52 56 32 E4 33 D2 B9 0A 00 F7 F1 80 CA 30 88	Q R V 2
03F0	14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 0C 30 88	. N 3 .
0400	04 5E 5A 59 C3 B4 09 CD 21 C3 B8 00 F0 8E C0 26	. ^ Z Y
0410	A0 FE FF 3C FF 74 2F 3C FE 74 32 3C FB 74 2E 3C	. . . <
0420	FC 74 31 3C FA 74 34 3C FA 74 37 3C F8 74 3A 3C	. t 1 <
0430	FD 74 3D 3C F9 74 40 E8 7D FF BE A8 01 83 C6 01	. t = <
0440	89 04 BA A8 01 C3 BA 03 01 E8 B9 FF C3 BA 08 01
0450	E8 B2 FF C3 BA 0F 01 E8 AB FF C3 BA 5B 02 E8 A4
0460	FF C3 BA 62 02 E8 9D FF C3 BA 69 02 E8 96 FF C3	. . . h
0470	BA 47 01 E8 8F FF C3 BA 4F 01 E8 88 FF C3 B4 30	. G . .
0480	CD 21 50 BE 60 01 83 C6 0F E8 54 FF 58 83 C6 02	. ! P .
0490	8A C4 E8 4B FF BA 60 01 E8 6A FF BE 76 01 83 C6	. . . K
04A0	13 8A C7 E8 3A FF BA 76 01 E8 59 FF BF 8D 01 83
04B0	C7 18 8B C1 E8 11 FF 8A C3 E8 FB FE 83 EF 02 89
04C0	05 BA 8D 01 E8 3E FF C3 E8 3F FF E8 B0 FF 32 C0

Рис.4 Файл загрузочного модуля «плохого» .EXE в 16-ном виде

A...	Hexadecimal (1 Byte)	Text (ASCII)
0000	E9 C5 01 50 43 0D 0A 24 43 2F 58 54 0D 0A 24 41	• • •
0010	54 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 33 30	T • •
0020	0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 35 30 20	• • \$
0030	6F 72 20 36 30 0D 0A 24 50 53 32 20 6D 6F 64 65	o r
0040	6C 20 38 30 0D 0A 24 50 D0 A1 6A 72 0D 0A 24 50	l 8
0050	43 20 43 6F 6E 76 65 72 74 69 62 6C 65 0D 0A 24	C C
0060	4D 53 2D 44 4F 53 20 76 65 72 73 6F 6E 3A 20 20	M S -
0070	2E 20 20 0D 0A 24 53 65 72 69 61 6C 20 6E 75 6D	. •
0080	62 65 72 20 4F 45 4D 3A 20 20 0D 0A 24 55 73 65	b e r
0090	72 20 73 65 72 69 61 6C 20 6E 75 6D 62 65 72 3A	r s e
00A0	20 20 20 20 20 0D 0A 24 20 0D 0A 24 24 0F 3C 09	
00B0	76 02 04 07 04 30 C3 51 8A E0 E8 EF FF 86 C4 B1	v • •
00C0	04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 E9 FF 88 25	• • •
00D0	4F 88 05 4F 8A C7 E8 DE FF 88 25 4F 88 05 5B C3	O • •
00E0	51 52 56 32 E4 33 D2 B9 0A 00 F7 F1 80 CA 30 88	Q R V
00F0	14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 0C 30 88	• N 3
0100	04 5E 5A 59 C3 B4 09 CD 21 C3 B8 00 F0 8E C0 26	• ^ Z
0110	A0 FE FF 3C FF 74 2F 3C FE 74 32 3C FB 74 2E 3C	• • •
0120	FC 74 31 3C FA 74 34 3C FA 74 37 3C F8 74 3A 3C	• t 1
0130	FD 74 3D 3C F9 74 40 E8 7D FF BE A8 01 83 C6 01	• t =
0140	89 04 BA A8 01 C3 BA 03 01 E8 B9 FF C3 BA 08 01	• • •
0150	E8 B2 FF C3 BA 0F 01 E8 AB FF C3 BA 5B 02 E8 A4	• • •
0160	FF C3 BA 62 02 E8 9D FF C3 BA 69 02 E8 96 FF C3	• • •
0170	BA 47 01 E8 8F FF C3 BA 4F 01 E8 88 FF C3 B4 30	• G •
0180	CD 21 50 BE 60 01 83 C6 0F E8 54 FF 58 83 C6 02	• ! P
0190	8A C4 E8 4B FF BA 60 01 E8 6A FF BE 76 01 83 C6	• • •
01A0	13 8A C7 E8 3A FF BA 76 01 E8 59 FF BF 8D 01 83	• • •
01B0	C7 18 8B C1 E8 11 FF 8A C3 E8 FB FE 83 EF 02 89	• • •
01C0	05 BA 8D 01 E8 3E FF C3 E8 3F FF E8 B0 FF 32 C0	• • •
01D0	B4 4C CD 21	• L •

Рис.5 Файл загрузочного модуля .COM в 16-ном виде


```

0000 4D 5A B0 00 03 00 01 00 20 00 00 00 FF FF 00 00 M Z *
0010 C8 00 C7 FE 1C 01 18 00 1E 00 00 00 01 00 20 01 *
0020 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
01A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
01B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
01C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
01D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
01E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
01F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
0290 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
02A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *

02B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
02C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
02D0 50 43 0D 0A 24 43 2F 58 54 0D 0A 24 41 54 0D 0A P C *
02E0 24 50 53 32 20 6D 6F 64 65 6C 20 33 30 0D 0A 24 $ P S 2
02F0 50 53 32 20 6D 6F 64 65 6C 20 35 30 20 6F 72 20 P S 2
0300 36 30 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 38 6 0 *
0310 30 0D 0A 24 50 D0 A1 6A 72 0D 0A 24 50 43 20 43 0 *
0320 6F 6E 76 65 72 74 69 62 6C 65 0D 0A 24 4D 53 2D o n v e
0330 44 4F 53 20 76 65 72 73 6F 6E 3A 20 20 2E 20 20 D O S
0340 0D 0A 24 53 65 72 69 61 6C 20 6E 75 6D 62 65 72 *
0350 20 4F 45 4D 3A 20 20 0D 0A 24 55 73 65 72 20 73 O E M
0360 65 72 69 61 6C 20 6E 75 6D 62 65 72 3A 20 20 20 e r i a
0370 20 20 0D 0A 24 20 0D 0A 24 00 00 00 00 00 00 00 *
0380 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8 EF $ *
0390 FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 *
03A0 E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25 4F *
03B0 88 05 5B C3 51 52 56 32 E4 33 D2 B9 0A 00 F7 F1 *
03C0 80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 *
03D0 04 0C 30 88 04 5E 5A 59 C3 B4 09 CD 21 C3 B8 00 *
03E0 F0 8E C0 26 A0 FE FF 3C FF 74 2F 3C FE 74 32 3C *
03F0 FB 74 2E 3C FC 74 31 3C FA 74 34 3C FA 74 37 3C *
0400 F8 74 3A 3C FD 74 3D 3C F9 74 40 E8 7D FF BE A5 *
0410 00 83 C6 01 89 04 BA A5 00 C3 BA 00 00 E8 B9 FF *
0420 C3 BA 05 00 E8 B2 FF C3 BA 0C 00 E8 AB FF C3 BA *
0430 AF 00 E8 A4 FF C3 BA B6 00 E8 9D FF C3 BA BD 00 *
0440 E8 96 FF C3 BA 44 00 E8 8F FF C3 BA 4C 00 E8 88 *
0450 FF C3 B4 30 CD 21 50 BE 5D 00 83 C6 0F E8 54 FF *
0460 58 83 C6 02 8A C4 E8 4B FF BA 5D 00 E8 6A FF BE X *
0470 73 00 83 C6 13 8A C7 E8 3A FF BA 73 00 E8 59 FF s *
0480 BF 8A 00 83 C7 18 8B C1 E8 11 FF 8A C3 E8 FB FE *
0490 83 EF 02 89 05 BA 8A 00 E8 3E FF C3 2B C0 50 B8 *
04A0 0D 00 8E D8 E8 37 FF E8 A8 FF 32 C0 B4 4C CD 21 *

```

Рис.6 Файл загрузочного модуля «хорошего» .EXE в 16-ном виде

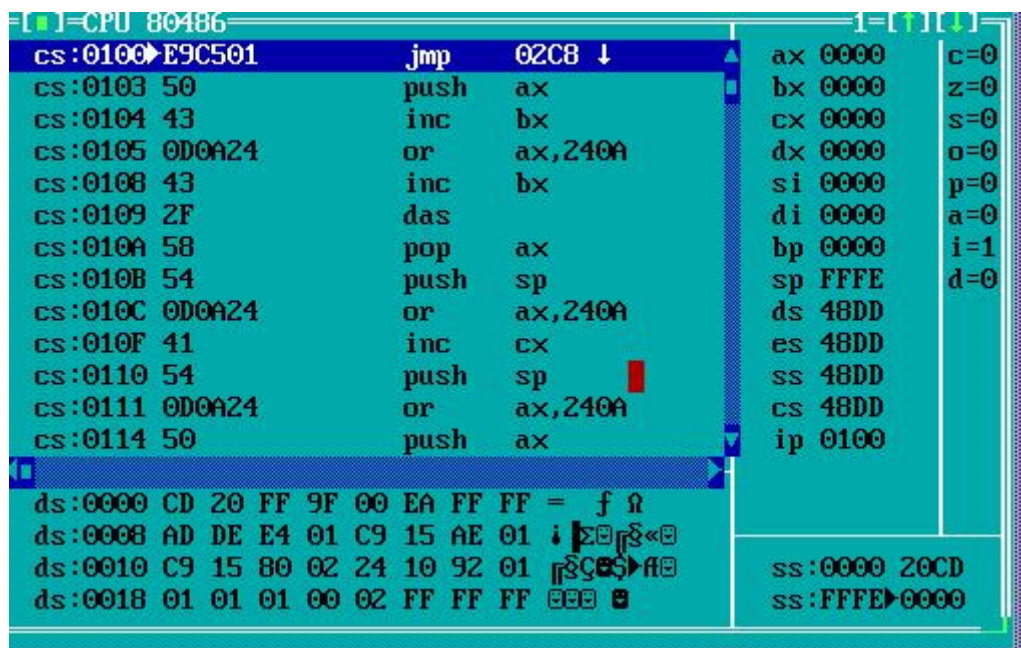


Рис. 7 Отладчик TD.EXE для .COM

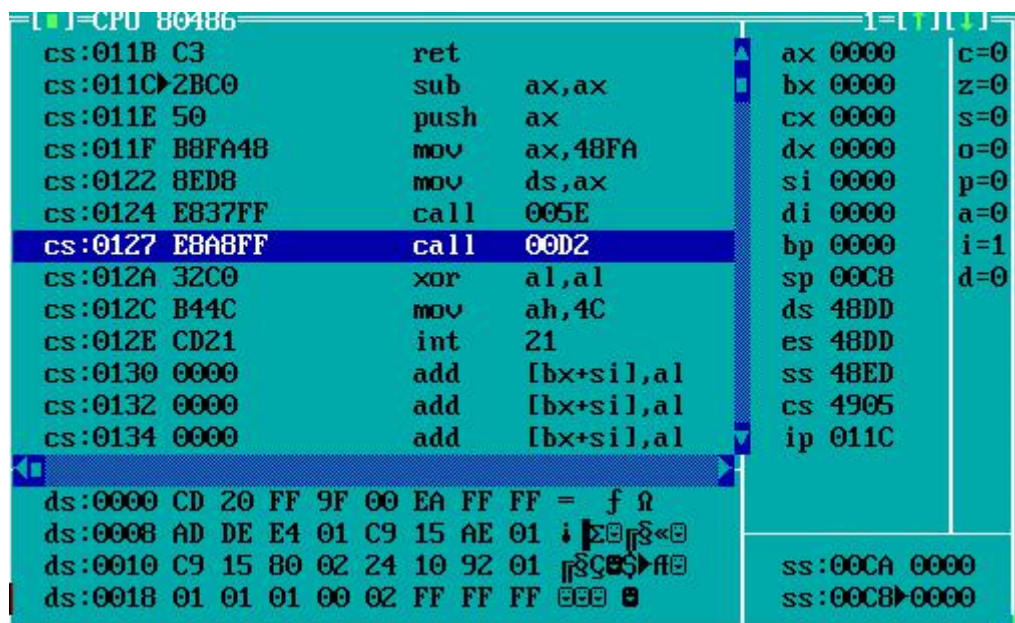


Рис. 8 Отладчик TD.EXE для .EXE

Ответы на вопросы.

1. Сколько сегментов должна содержать COM-программа?

COM-программа должна содержать один сегмент.

2. EXE-программа от 1 и более сегментов.

3. Какие директивы должны быть обязательно в тексте COM-программы?

org 100h – обязательная директива для COM-программы, подгружается PSP размером в 100h, а также assume для задания соответствия между сегментами и сегментными регистрами

4. Все ли форматы команд можно использовать в COM-программе?

Нельзя использовать команды, где используются перемещаемые сегменты

Отличия форматов файлов .COM и .EXE программ:

1. Какова структура файла .COM? С какого адреса располагается код?

Файл .COM состоит из одного сегмента и содержит как данные, так и сам код программы. Код начинается с адреса 0h.

2. Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с адреса 0?

В «плохом» EXE все находится в одном сегменте. Код располагается с адреса 300h. С адреса ноль располагается заголовок и таблица настроек.

3. Какова структура «хорошего» EXE? Чем он отличается от файла «плохого» EXE?

В «хорошем» exe код и данные хранятся в отдельных сегментах, а в «плохом» они не разделяются на сегменты. «плохой» exe не учитывает стек и смещается на 100h. В итоге у «плохого» exe код начинается с 300h, а в «хорошем» exe стек располагается в отдельном сегменте и код начинается с 2D0h

Загрузка COM модуля в основную память:

1. Какой формат загрузки модуля COM? С какого адреса располагается код?

Первые 100h уходят на PSP. Затем инициализируются сегментные регистры, которые указывают на начало PSP. Код начинается с адреса 100h, сразу после PSP.

2. Что располагается с адреса 0?

PSP

3. Какие значения имеют сегментные регистры? На какие области памяти они указывают?

Регистры имеют значения 48DD, указывают на начало PSP.

4. Как определяется стек? Какую область памяти он занимает? Какие адреса?

Стек определяется автоматически, располагается с 0h до fffeh.

Загрузка «хорошего» EXE модуля в основную память:

1. Как загружается “хороший” EXE? Какие значения имеют сегментные регистры?

Первые 100h присваиваются PSP. Потом инициализируются сегментные регистры. DS и ES указывают на 48DD, SS на 48ED, CS на 4905.

2. На что указывают регистры DS и ES?

Регистры DS и ES указывают на начало PSP.

3. Как определяется стек?

При помощи директивы .STACK, задается размер стека. Регистр SS указывает на начало сегмента стека, а SP на его конец.

4. Как определяется точка входа?

Точка входа определяется при помощи директивы END.

Выводы.

В ходе данной лабораторной работы были изучены структуры модулей типов COM и EXE.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
AStack SEGMENT STACK
    DB 1000 DUP(?)
AStack ENDS

DATA SEGMENT
;IS_INTERRUPT_LOAD dw, 0
;IS_UN_LOAD dw, 0
LOADED dw 0
INTERRUPT_LOADED db 'Interrupt was already loaded' , 0dh, 0ah, '$'
UN_LOADED db '/un loaded', 0dh, 0ah, '$'
RESET_INTERRUPT db 'Reset interruption', 0dh, 0ah, '$'
COMPLITE_LOADING db 'loading complete', 0dh, 0ah, '$'
NOT_INTERRUPT db 'Interruption is not loaded', 0dh, 0ah, '$'
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:AStack

ROUT PROC FAR
    jmp start

    KEEP_CS DW 0; для хранения сегмента
    KEEP_IP DW 0; и мещения прерывания
    KEEP_SP DW 0
    KEEP_SS DW 0
    KEEP_PSP DW 0
    SIGNATURE DW 1234h
    int9_vect dd 0
    REQ_KEY db 21h; f
    new_stack dw 100h dup (?)

    start:

    mov KEEP_SP, sp
    mov KEEP_SS, ss
    mov sp, offset new_stack
    add sp, 100
    push ax
    mov ax, seg new_stack
    mov ss, ax
```



```

pop ax

push ax
push cx
push ds
push es

in al, 60h
cmp al, REQ_KEY
je do_req

pop es
pop ds
pop cx
pop ax
mov ss, KEEP_SS
mov sp, KEEP_SP
jmp cs:[int9_vect]

do_req:
push ax
in al, 61h
mov ah, al
or al, 80h
out 61h, al
xchg ah, al
out 61h, al
mov al, 20h
out 20h, al
pop ax

print_0:
mov ah, 05h
mov cl, '0'
mov ch, 00h
int 16h
or al, al
jnz skip
jmp end_int

skip:
mov ax, 0040h

```

```

    mov es, ax
    mov ax, es:[1ah]
    mov es:[1ch], ax
    jmp print_0

end_int:
    pop es
    pop ds
    pop cx
    pop ax
    mov ss, KEEP_SS
    mov sp, KEEP_SP
    iret

ROUT endp

print proc near
    mov     AH,     09h
    int     21h

    ret
print ENDP

IS_INERRUPTION_SET proc near
    push bx
    push es
    push si

    mov ah, 35h; функция получения вектора
    mov al, 09h; номер вектора
    int 21h

    mov si, offset SIGNATURE
    sub si, offset ROUT
    cmp es:[bx+si], 1234h
    jne return

    mov dx, offset INTERRUPT_LOADED
    call print

    mov LOADED, 1

```

```

    jmp end_ret

return:
mov KEEP_IP, bx; запоминание смещения
mov KEEP_CS, es; и сегмента

end_ret:
pop si
pop es
pop bx
ret

IS_INERRUPTION_SET endp

IS_UN_SET proc near

    mov al, es:[81h + 1]
    cmp al, '/'
    jne ending
    mov al, es:[81h + 2]
    cmp al, 'u'
    jne ending
    mov al, es:[81h + 3]
    cmp al, 'n'
    jne ending

    cmp LOADED, 1
    je int_and_un

    mov dx, offset NOT_INTERRUPTION
    call print
    mov LOADED, 2
    jmp ending

int_and_un:
mov LOADED, 10
mov dx, offset UN_LOADED
call print

ending:
ret

```

```
IS_UN_SET ENDP
```

```
FREE_MEMORY proc near
```

```
    mov ah, 35h
    mov al, 09h
    int 21h
    xor ax, ax

    cli
    push ds
    mov dx, es:KEEP_IP
    mov ax, es:KEEP_CS
    mov ds, ax
    mov ah, 25h
    mov al, 09h
    int 21h; восстанавливаем вектор
    pop ds
    sti

    mov ax, es:KEEP_PSP
    mov es, ax
    push es
    mov ax, es:[2ch]; адрес среды
    mov es, ax

    mov ah, 49h
    int 21h; освобождение среды

    pop es
    mov ah, 49h
    int 21h

    mov dx, offset RESET_INTERRUPT
    call print
```

```
ret
```



```
FREE_MEMORY ENDP
```

```
SET_INTERRUPTION proc near
```

```
    mov ah, 35h
    mov al, 09h
    int 21h
    mov KEEP_CS, es
    mov KEEP_IP, bx
    mov word ptr int9_vect[02h], es
    mov word ptr int9_vect, bx
```

```
    push ds
    push ax
    push dx
    mov dx, offset ROUT
    mov ax, seg ROUT
    mov ds, ax
    mov ah, 25h
    mov al, 09h
    int 21h
```

```
    pop dx
    pop ax
    pop ds
```

```
ret
```

```
SET_INTERRUPTION ENDP
```

```
SAVE_MEMORY proc near
```

```
    push ax
    push bx
    push dx
    push cx
```

```
    mov dx, offset COMPLITE_LOADING
    call print
```

```
    mov DX, offset LAST
    mov cl, 4h
    shr dx, cl
    inc dx
```

```

    mov ax,cs
    sub ax, KEEP_PSP
    add dx,ax
    xor ax,ax
    mov ah,31h
    int 21h

    pop cx
    pop dx
    pop bx
    pop ax

ret
SAVE_MEMORY ENDP

Main proc far

    mov ax, DATA
    mov ds, ax
    mov KEEP_PSP, es

    call IS_INTERRUPTION_SET
    call IS_UN_SET

    cmp LOADED, 10
    je free_mem
    cmp LOADED, 2
    je finish
    cmp LOADED, 1
    je finish
    call SET_INTERRUPTION
    call SAVE_MEMORY
    jmp finish

free_mem:
    call FREE_MEMORY

```

```
finish:
mov ah, 4ch
int 21h
LAST:
```

```
Main endp
```

```
CODE ENDS
```

```
END Main
```