

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

**ОТЧЕТ**  
**по практической работе № 1**

**по дисциплине «Операционные системы»**

**Тема: Исследование структур загрузочных модулей**

Студент гр. 8383

Степанов В.Д.

Преподаватель

Губкин А.Ф.

Санкт-Петербург

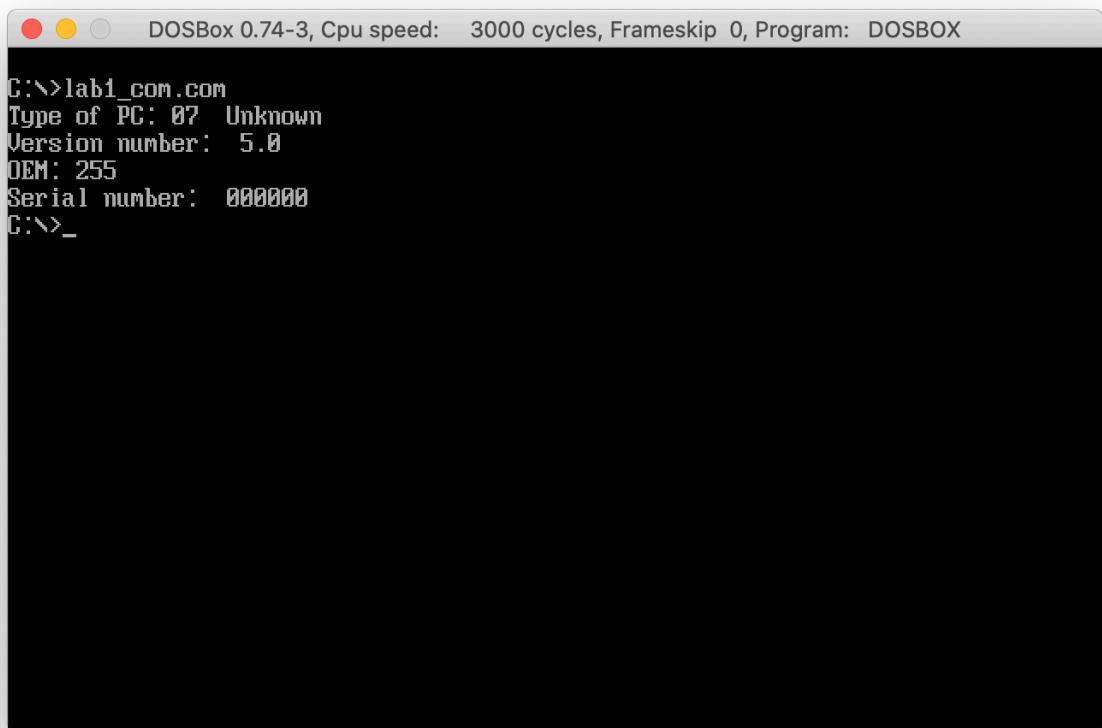
2020

## **Цель работы.**

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

## **Выполнение работы.**

Был написан текст исходного **.COM** модуля, который определяет тип РС и версию системы. Текст кода представлен в приложении А. Из написанного текста был получен «хороший» **.COM** модуль и «плохой» **.EXE** модуль. Работа модулей представлена на рисунке 1 и рисунке 2 соответственно.



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX  
C:\>lab1\_com.com  
Type of PC: 07 Unknown  
Version number: 5.0  
OEM: 255  
Serial number: 000000  
C:\>\_

Рисунок 1 – Работа «хорошего» **.COM** модуля

Был написан текст для «хорошего» **.EXE** модуля, который выполняет те же функции, что и «хороший» **.COM** модуль. Работа модуля представлена на рисунке 3.

```
C:\>lab1_com.exe

Type of PC: 07
Type of PC: 07
Type of PC:
Type of PC: 5.0
Type of PC:
Type of PC: 255
Type of PC: 000000
Type of PC:
C:\>
```

Рисунок 2 – Работа «плохого» .EXE модуля

```
C:\>lab1_exe.exe
Type of PC: 07 Unknown
Version number: 5.0
OEM: 255
Serial number: 000000
C:\>
```

Рисунок 3 – Работа «плохого» .EXE модуля

## Сравнение текстов .COM и .EXE модулей

- 1) .COM-программа содержит один сегмент не превышающий 64К.
- 2) .EXE-программа может содержать множество сегментов.

3) ORG 100h, которая устанавливает в начале программы значение счетчика адресов равное 100h, т.к. операционная система при загрузке программы размещает в ее первые 100h байт префикс программного сегмента.

4) Нет. Например нельзя задавать сегмент стека, так как для СОМ-программы стек выделяется операционной системой в конце сегмента программы.

В приложении FAR были открыты все созданные модули в шестнадцатеричном виде. Содержимое файлов приведено на рисунках 4, 5 и 6.

0000000000:	4D 5A 22 01 03 00 00 00 00	20 00 00 00 FF FF 00 00 MZ'@W	
0000000010:	00 00 00 00 01 00 00 00 00	3E 00 00 00 01 00 FB 50 00 > @ M	
0000000020:	6A 72 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 jr	
0000000030:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000040:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000050:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000060:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000070:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000080:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000090:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000A0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000B0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000C0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000D0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000E0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000F0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000100:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000110:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000120:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000130:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000140:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000150:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000160:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000170:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000180:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000190:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000001A0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000001B0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000001C0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000001D0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000001E0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000001F0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000200:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000210:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000220:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000230:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000240:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000250:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000260:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000270:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000280:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000290:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000002A0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000002B0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000002C0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000002D0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000002E0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000002F0:	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000300:	E9 13 92 54 79 70 65 20	6F 66 20 50 43 3A 20 24 иНоФtpe of PC: \$	
0000000310:	50 43 24 50 43 2F 58 54	PC5PC-X1501SPS2	
0000000320:	6D 6F 54 65 6C 20 33 30	24 50 53 20 30 6D 6F 54	
0000000330:	65 6C 20 35 30 20 6D 72	model 1 808PCj+SPC	
0000000340:	6D 6F 64 65 6C 20 38 30	24 50 43 66 72 24 50 43	
0000000350:	20 63 6F 6E 26 65 22 74	61 62 6C 65 24 20 55 6E	
0000000360:	6B 6E 6F 77 6E 24 20 20	convertables Un	
0000000370:	69 6F 6E 20 6E 75 6D 62	knowns\$ \$JQlvers	
0000000380:	20 20 24 00 00 4F 45 4D	65 72 30 20 24 20 20 2E	
0000000390:	20 24 00 00 53 65 72 69	ion number: \$ .	
00000003A0:	72 30 20 24 20 20 20 20	\$JQ OEM: \$ .	
00000003B0:	24 50 52 80 03 01 E8 26	\$PRSerial numbe	
00000003C0:	FE FF B0 07 3C FF 24 34	r: \$PRewEmu # n?A&	
00000003D0:	3C FC 24 34 3C FA 24 36	3C FC 74 36 3C FB 74 32 ил" <at4mt6wt8wts:	
00000003E0:	3C FD 24 3C 3C F9 24 3E	3C FC 74 38 3C FB 74 38 <т4т6т8т8wt:	
00000003F0:	B8 66 01 E8 39 00 B8 5D	BF 66 01 E8 6D 00 89 05 <зт<тwt>Yf@n 2%	
0000000400:	08 00 73 F1 3C 00 74 04	01 EB 2E 90 B8 10 01 EB ef@9 e10...7e@0@	
0000000410:	01 EB 16 90 B8 29 01 EB	08 19 01 EB 1C 98 B8 1C <?e!@?>?e!@?>e- <td></td>	
0000000420:	04 01 EB 04 90 B8 4E	10 90 B8 3C 01 EB 08 90 Q...?e?Q...?e?Q...?e- <td></td>	
0000000430:	2B C0 04 09 CD 21 58 C3	01 EB 08 00 58 C3 50 eI@?#?N@?X@?Z@?P	
0000000440:	4F 88 05 4F 88 C7 E8 12	53 80 FC E8 1D 00 88 25 +?@DHXGT?ыыи" ?%	
0000000450:	24 0F 3C 09 76 02 04 07	08 00 25 4F 88 05 SB C3 0?@?Э?#? ?z?@?@P	
0000000460:	FF 86 C4 B1 04 D2 E8 E8	04 30 C3 51 8A E0 E8 EF \$K*o@*+@#G?@ann	
0000000470:	33 D2 09 00 F7 F1 80	E6 FF 59 C3 51 52 32 E4 я+Д+@ИникаУTQR2d	
0000000480:	C0 30 88 14 4E 33 D2 3D	0C 30 88 04 58 C3 50 0 sc< t#9@?ZVTP	
0000000490:	53 51 52 2B C0 B4 30 CD	21 BA 64 01 E8 90 FF 50 SQR+A?@H@?ж@?@?P	
00000004A0:	BE 7D 01 46 E8 C5 FF 58	8A C4 83 C6 03 E8 BC FF ?>@и@?Х@?Д?Ж@?з@?	
00000004B0:	B8 7D 01 E8 79 FF B8 83	01 E8 73 FF BE B1 81 83 е>@и@?@?@?@?@?@?	
00000004C0:	C6 02 B8 C7 E8 A5 FF B8	8B 01 E8 62 FF B8 92 01 Ж@?З@?я@?С@?и@?@?	
00000004D0:	E8 5C FF BF A4 01 B3 C7	06 B8 C1 E8 5A FF B8 C3 и@?@?@?З@?Б@?З@?Г	
00000004E0:	E8 78 FF B3 EP 02 89 05	B4 04 01 E8 41 FF 5A 59 и@?@?@?@?@?@?@?@?	
00000004F0:	5D 58 C3 50 53 2B DB B3	B6 04 00 F6 F3 B1 D6 B0 SQR+A?@H@?ж@?@?P	
0000000500:	C2 30 2B C0 B4 02 CD 21	8A D6 80 C2 30 2B C0 B4 В@?@?@?@?@?@?@?@?	
0000000510:	02 CD 21 5B 58 C3 E8 98	FE E8 73 FF 33 C0 B4 4C @?@?@?@?@?@?@?@?	

Рисунок 4 – Содержание «плохого» .EXE модуля

000000000000: E9 48 02 54 79 70 65 20	6F 66 20 50 43 3A 20 24	иНетуре of PC: \$
0000000010: 50 43 24 50 43 2F 58 54	24 41 54 24 50 53 32 20	PC\$PC\$XT\$AT\$PS2
0000000020: 6D 6F 64 65 6C 20 33 30	24 50 53 32 20 6D 6F 64	model 30\$PS2 mod
0000000030: 65 6C 20 35 30 20 6F 72	20 36 30 24 50 53 32 20	el 50 or 60\$PS2
0000000040: 6D 6F 64 65 6C 20 38 30	24 50 43 6A 72 24 50 43	model 80\$PCjr\$PC
0000000050: 20 63 6F 6E 76 65 72 74	61 62 6C 65 24 20 55 6E	convertable\$ Un
0000000060: 6B 6E 6F 77 6E 24 20 20	20 24 0D 0A 56 65 72 73	known\$ \$РЮers
0000000070: 69 6F 6E 20 6E 75 6D 62	65 72 3A 20 24 20 20 2E	ion number: \$ -
0000000080: 20 20 24 0D 0A 4F 45 4D	3A 20 24 20 20 20 20 20	\$РЮEM: \$ -
0000000090: 20 24 0D 0A 53 65 72 69	61 6C 20 6E 75 6D 62 65	\$РЮSerial numbe
00000000A0: 72 3A 20 24 20 20 20 20	20 20 20 20 20 20 20 20	r: \$
00000000B0: 24 50 52 BA 03 01 E8 76	00 B8 00 F0 8E C0 26 A0	\$PRe\$@и ё р?А&
00000000C0: FE FF B0 07 3C FF 74 34	3C FE 74 36 3C FB 74 32	юя\$•<ят4<ют6<ют2
00000000D0: 3C FC 74 34 3C FA 74 36	3C FC 74 38 3C F8 74 3A	<ят4<ят6<ят8<ят:
00000000E0: 3C FD 74 3C 3C F9 74 3E	BF 66 01 E8 A2 00 89 05	<ят<ят>иf@иү >&
00000000F0: BA 66 01 E8 39 00 BA 5D	01 EB 2E 90 BA 10 01 EB	еf@и9 ё ]@л. ?е►@л
0000000100: 28 90 BA 13 01 EB 22 90	BA 19 01 EB 1C 90 BA 1C	<?е!!@л"?е4@л-?е-
0000000110: 01 EB 16 90 BA 29 01 EB	10 90 BA 3C 01 EB 0A 90	@л-?е>@л!>е<@л@?
0000000120: BA 49 01 EB 04 90 BA 4E	01 E8 03 00 5A 58 C3 50	еI@л♦?еN@и ZXGP
0000000130: 2B C0 B4 09 CD 21 58 C3	50 53 2B DB B3 10 B4 00	+A?оH!ХГРС+И?►?
0000000140: F6 F3 8B D0 8B D0 80 C2	30 2B C0 80 FA 3A 72 03	цу<Р<Р>В0+A?ь:г▼
0000000150: 80 C2 07 B4 02 CD 21 8A	D6 80 C2 30 2B C0 80 FA	?В•?оH!-?Ц?В0+A?ь
0000000160: 3A 72 03 80 C2 07 B4 02	CD 21 5B 58 C3 53 8A FC	:г▼?В•?оH! [ХГР\$Ь
0000000170: E8 1D 00 88 25 4F 88 05	4F 8A C7 E8 12 00 88 25	и+ ?зО?Ф?Зи‡ ?%
0000000180: 4F 88 05 5B C3 24 0F 3C	09 76 02 04 07 04 30 C3	О?&[Г\$*к@оv♦♦*ОГ
0000000190: 51 8A E0 E8 EF FF 86 C4	B1 04 D2 E8 E6 FF 59	Q?аипя+Д+ТиижяУ
00000001A0: C3 51 52 32 E4 33 D2 B9	0A 00 F7 F1 80 CA 30 88	ГQR2л3TH@ чс?Ю?
00000001B0: 14 4E 33 D2 3D 0A 00 73	F1 3C 00 74 04 0C 30 88	ЧN3T=© sc< t♦?О?
00000001C0: 04 5A 59 C3 50 53 51 52	2B C0 B4 30 CD 21 BA 6A	♦ZYGPSQR+A?оH!ej
00000001D0: 01 E8 5B FF 50 BE 7D 01	46 E8 C5 FF 58 8A C4 83	Ои[яР?]@РиЕяХ?Л?
00000001E0: C6 03 E8 BC FF BA 7D 01	E8 44 FF BA 83 01 E8 3E	Ж@и?яе>ОиDяе?Ои>
00000001F0: FF BE 8B 01 83 C6 02 8A	C7 E8 A5 FF BA 8B 01 E8	я?<@?Ж@?Зи?яе<Ои
0000000200: 2D FF BA 92 01 E8 27 FF	BF A4 01 83 C7 06 8B C1	-яе'Ои'я'Г@?З@<Б
0000000210: E8 5A FF 8A C3 E8 78 FF	83 EF 02 89 05 BA A4 01	иЗя?Гихя?п@иФея
0000000220: E8 0C FF 5A 59 5B 58 C3	50 53 2B DB B3 0A B4 00	и@яZY [ХГРС+И?©?
0000000230: F6 F3 8B D0 80 C2 30 2B	C0 B4 02 CD 21 8A D6 80	цу<Р>В0+A?оH!-?Ц?
0000000240: C2 30 2B C0 B4 02 CD 21	5B 58 C3 E8 63 FE E8 73	В0+A?оH! [ХГисюис
0000000250: FF 33 C0 B4 4C CD 21		язA?ЛН!

Рисунок 5 – Содержание «хорошего» .COM модуля

### Отличия форматов файлов COM и EXE модулей

- 1) В файлах содержится только машинный код и данные программы. Код программы начинается с адреса 0h, но при загрузке происходит смещение на 100h.
- 2) У «плохого» .EXE модуля данные и код располагаются в одном сегменте, так как он был получен текста для .COM модуля. Код располагается по адресу 300h, так как перед ним начиная с адреса 0h находится таблица настроек.
- 3) Так как «плохой» .EXE был получен из текста для .COM модуля, поэтому у него данные и код располагаются в одном сегменте, в отличии от «хорошего» .EXE. Так же в «плохом» .EXE отсутсвует выделение стека.

000000000000: 4D 5A 36 00 03 00 01 00	20 00 00 00 FF FF 00 00 MZ6 ♥ ⊖ яя
00000000010: 02 00 00 00 61 01 0C 00	3E 00 00 00 01 00 FB 50 ⊖ a⊖♀ > ⊖ ыР
00000000020: 6A 72 00 00 00 00 00 00	00 00 00 00 00 00 00 00 jr
00000000030: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 65 01 ♀
00000000040: 0C 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000050: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000060: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000070: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000080: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000090: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000000A0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000000B0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000000C0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000000D0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000000E0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000000F0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000100: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000110: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000120: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000130: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000140: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000150: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000160: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000170: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000180: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000190: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000001A0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000001B0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000001C0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000001D0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000001E0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000001F0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000200: 00 01 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000210: 54 79 70 65 20 6F 66 20	50 43 3A 20 24 50 43 24 Type of PC: \$PC\$
00000000220: 50 43 2F 58 54 24 41 54	PC/XT\$AT\$PS2 mod
00000000230: 65 6C 20 33 30 24 50 53	e1 30\$PS2 model
00000000240: 35 30 20 6F 72 20 36 30	50 or 60\$PS2 mod
00000000250: 65 6C 20 38 30 24 50 43	e1 80\$PCjr\$PC co
00000000260: 6E 76 65 72 74 61 62 6C	nvertable\$ Unkno
00000000270: ?7 6E 24 20 20 20 24 0D	wn\$ \$P\$Version
00000000280: 20 6E 75 6D 62 65 72 3A	number: \$ . \$
00000000290: 0D 0A 4F 45 4D 3A 20 24	JOEM: \$ . \$
000000002A0: 0A 53 65 72 69 61 6C 20	Serial number:
000000002B0: 24 20 20 20 20 20 20 20	\$ . \$
000000002C0: 50 52 BA 00 00 EB 26 00	PRe ии ё р?А8 и
000000002D0: FF B0 07 3C FF 74 34 3C	я°«ят4 <lt>т6<lt>т2&lt;</lt></lt>
000000002E0: FC 74 34 3C FA 74 36 3C	FC 74 38 3C F8 74 3A 3C
000000002F0: FD 74 3C 3C F9 74 3E BF	63 00 E8 6D 00 89 05 BA
00000000300: 63 00 E8 39 00 BA 5A 00	эт<шт>иc им %е
00000000310: 90 BA 10 00 EB 22 90 BA	с и9 еZ л.?е? л<
00000000320: EB 16 90 BA 26 00 EB 10	?е? л"?е? л?е! л
00000000330: 46 00 EB 04 90 BA 4B 00	90 BA 39 00 EB 0A 90 BA
00000000340: C0 B4 09 CD 21 58 C3 53	л?е8 л?е? л?е
00000000350: 88 05 4F 8A C7 8E 12 00	E8 03 00 5A 58 C3 50 2B
00000000360: 0F 3C 09 26 02 04 07 04	8A FC E8 1D 00 88 25 4F
00000000370: 86 C4 B1 04 D2 E8 E8 E6	88 25 4F 88 05 5B C3 24
00000000380: D2 B9 0A 00 F7 F1 80 CA	30 C3 51 8A E0 E8 EF FF
00000000390: 00 73 F1 3C 00 74 04 0C	FF 59 C3 51 52 32 E4 33
000000003A0: 2B C0 B4 30 CD 21 BA 67	†Д+ФТиижяУГQR2д3
000000003B0: 46 E8 C7 FF 58 8A C4 83	30 88 14 4E 33 D2 3D 0A
000000003C0: E8 7B FF BA 80 00 E8 75	30 88 04 5A 59 C3 50 53
000000003D0: C7 E8 A7 FF BA 88 00 E8	+A?ОН!ег и'яР?з
000000003E0: BF A1 00 83 C7 06 8B C1	РиЗаX?Д?Ж?и?еэ
000000003F0: 83 EF 02 89 05 BA A1 00	FF BE 88 00 83 C6 02 8A
00000000400: 2B DB B3 0A B4 00 F6 F3	64 FF BA 8F 00 E8 5E FF
00000000410: 02 CD 21 8A D6 8C C2 30	88 5C FF 8A C3 E8 ?R FF
00000000420: C3 50 2B C0 B8 01 00 8E	88 43 FF 5B 58 C3 50 53
00000000430: 2B C0 B4 4C CD 21	8B D0 80 C2 30 2B C0 B4
	2B C0 B4 02 CD 21 5B 58
	D8 58 E8 93 FE E8 6E FF
	ГР+#e@ ?ШХи"юнля
	+A?ЛН!

Рисунок 6 – Содержание «хорошего» .EXE модуля

При помощи отладчика TD.EXE был загружен .COM модуль. Результат загрузки показан на рисунке 7.

### Загрузка .COM модуля в основную память

- 1) Код, данные, стек и PSP располагаются в одном сегменте. Код располагается с адреса 0100h.
- 2) С адреса 0h располагается PSP и заканчивается перед адресом 100h.

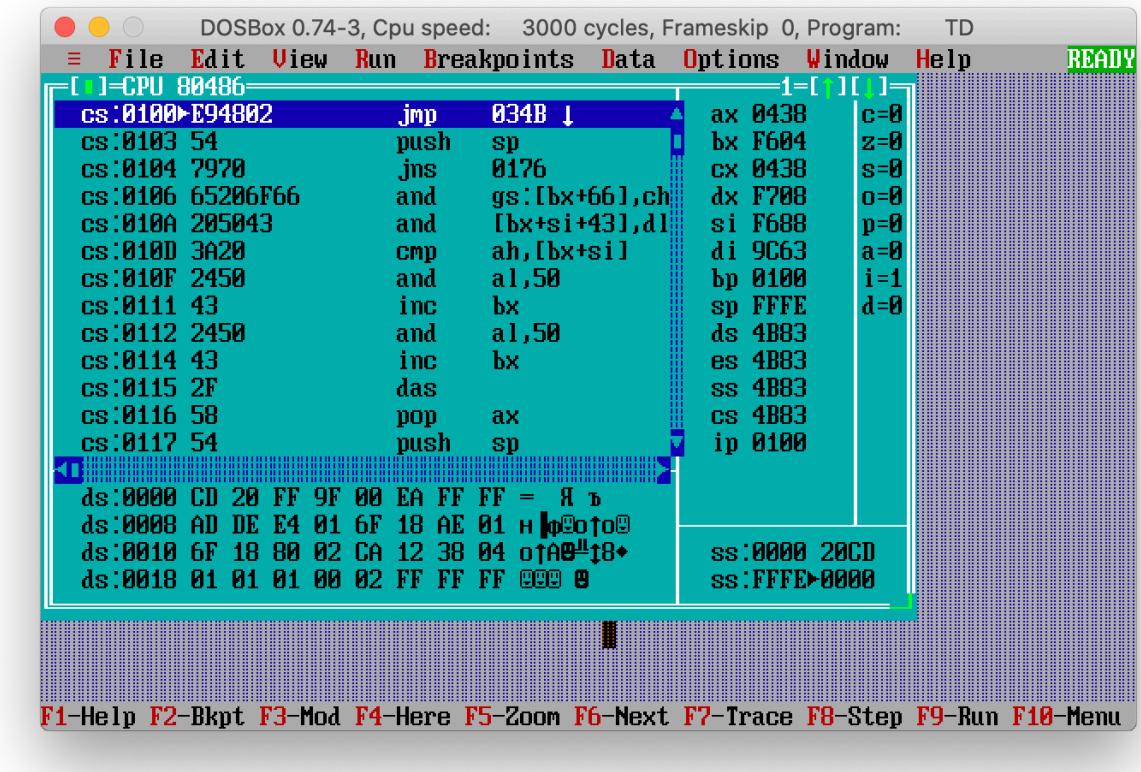


Рисунок 7 – Открытие .COM модуля в TD.EXE

- 3) Все сегментные регистры указывают на начало единственного регистра (начало PSP).
- 4) Стек занимает место, которое на занято PSP и машинным кодом. Регистр SP указывает на адрес FFFEh.

При помощи отладчика TD.EXE был загружен «хороший» .EXE модуль. Результат загрузки показан на рисунке 8.

### **Загрузка «хорошего» .EXE модуля в основную память**

- 1) Определяется свободный участок памяти, в который можно будет загрузить программу. Выделяется сегмент для PSP . CS указывает на начало кода, SP указывает на начало стека.
- 2) DS и ES указывают на начало PSP.
- 3) Стек определяется вручную через директиву ASSUME: SS:LAB1STACK. Если объявление отсутствует, то стек создается по адресу FFFEh.
- 4) При помощи директивы END.

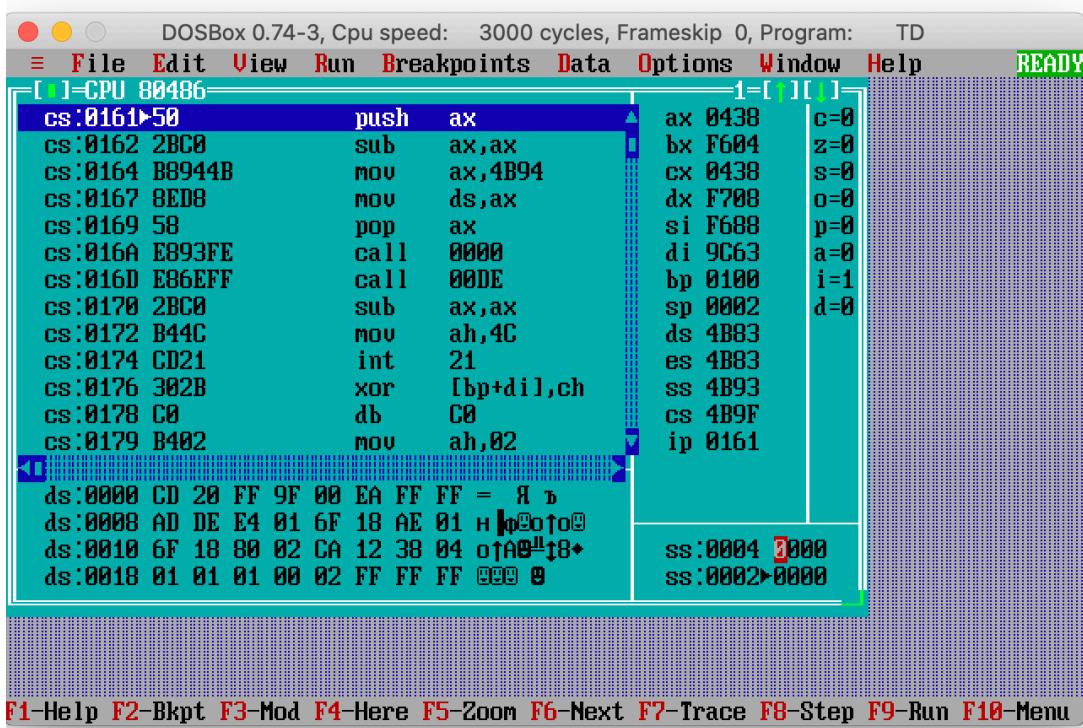


Рисунок 8 – Открытие «хорошего» .EXE модуля в TD.EXE

### Выводы.

В ходе лабораторной работы были исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов загрузки в основную память.

## ПРИЛОЖЕНИЕ А

### КОД ДЛЯ .COM МОДУЛЯ

```
LAB1 SEGMENT
    ASSUME CS:LAB1, DS:LAB1, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN

;-----
TYPE_OF_PC db "Type of PC: $"
PC db "PC$"
PC_XT db "PC/XT$"
AT db "AT$"
PS2_30 db "PS2 model 30$"
PS2_50 db "PS2 model 50 or 60$"
PS2_80 db "PS2 model 80$"
PCjr db "PCjr$"
PC_Con db "PC convertable$"
UNKNOWN_TYPE db " Unknown$"
OUT_UNKNOWN db "      $"
VER_NUM db 13, 10, "Version number: $"

OUT_VER_NUM db "  .  $"

OEM_NUM db 13, 10, "OEM: $"
OUT_OEM db "          $"
SER_NUM db 13, 10, "Serial number: $"
OUT_SER_NUM DB "                  $"
;-----

TYPE_PC PROC near

    push ax
    push dx

    mov dx, offset TYPE_OF_PC
    call PRINT

    mov ax, 0F000h
    mov es, ax
    mov al, es:[0FFEh]

    mov al, 7h
```

```

    cmp al, 0FFh
        je W_PC
    cmp al, 0FEh
        je W_PC_XT
    cmp al, 0FBh
        je W_PC_XT
    cmp al, 0FCh
        je W_AT
    cmp al, 0FAh
        je W_PS2_30
    cmp al, 0FCh
        je W_PS2_50
    cmp al, 0F8h
        je W_PS2_80
    cmp al, 0FDh
        je W_PCjr
    cmp al, 0F9h
        je W_PC_Con

```

```

    mov di, offset OUT_UNKNOWN
    call BYTE_TO_HEX
    mov [di], ax
    mov dx, offset OUT_UNKNOWN
    CALL PRINT

```

```

    mov DX, offset UNKNOWN_TYPE
    jmp ESC_PROC

```

W\_PC:

```

    mov DX, offset PC
    jmp ESC_PROC

```

W\_PC\_XT:

```

    mov DX, offset PC_XT
    jmp ESC_PROC

```

W\_AT:

```

    mov DX, offset AT
    jmp ESC_PROC

```

W\_PS2\_30:

```

        mov DX, offset PS2_30
        jmp ESC_PROC

W_PS2_50:
        mov DX, offset PS2_50
        jmp ESC_PROC

W_PS2_80:
        mov DX, offset PS2_80
        jmp ESC_PROC

W_PCjr:
        mov DX, offset PCjr
        jmp ESC_PROC

W_PC_Con:
        mov DX, offset PC_Con

ESC_PROC:
        call PRINT

pop dx
pop ax

ret
TYPE_PC ENDP

;-----

PRINT PROC near

push ax
sub ax, ax
mov ah, 9h
int 21h
pop ax

ret
PRINT ENDP

;-----


WRD_TO_HEX    PROC    near

```

```

        push      BX
        mov       BH, AH
        call     BYTE_TO_HEX
        mov       [DI], AH
        dec       DI
        mov       [DI], AL
        dec       DI
        mov       AL, BH
        call     BYTE_TO_HEX
        mov       [DI], AH
        dec       DI
        mov       [DI], AL
        pop       BX
        ret

WRD_TO_HEX ENDP

;-----

TETR_TO_HEX PROC near
        and      AL, 0Fh
        cmp      AL, 09
        jbe      NEXT
        add      AL, 07
NEXT:   add      AL, 30h
        ret

TETR_TO_HEX ENDP

;-----


BYTE_TO_HEX PROC near
        push    CX
        mov     AH, AL
        call    TETR_TO_HEX
        xchg   AL, AH
        mov     CL, 4
        shr     AL, CL
        call    TETR_TO_HEX ;â AL ñòàðøàÿ öèôðà
        pop     CX           ;â AH ìëàäøàÿ
        ret

BYTE_TO_HEX ENDP

;
```

```

BYTE_TO_DEC PROC near
    push    CX
    push    DX
    xor     AH, AH
    xor     DX, DX
    mov     CX, 10
loop_bd:  div    CX
    or     DL, 30h
    mov     [SI], DL
    dec    si
    xor     DX, DX
    cmp    AX, 10
    jae    loop_bd
    cmp    AL, 00h
    je     end_1
    or     AL, 30h
    mov     [SI], AL

end_1:   pop    DX
    pop    CX
    ret

BYTE_TO_DEC ENDP

```

;-----

```

SYSTEM_VERSION PROC

    push ax
    push bx
    push cx
    push dx

    sub ax, ax
    mov ah, 30h
    int 21h

    mov dx, offset VER_NUM
    call PRINT

    push ax
    mov si, offset OUT_VER_NUM
    inc si
    call BYTE_TO_DEC

```

```
pop ax
mov al, ah
add si, 3
call BYTE_TO_DEC

mov dx, offset OUT_VER_NUM
call print

mov DX, offset OEM_NUM
call PRINT

mov si, offset OUT_OEM
add si, 2
mov al, bh
call BYTE_TO_DEC
mov dx, OFFSET OUT_OEM
call PRINT

mov dx, offset SER_NUM
call PRINT

mov di, offset OUT_SER_NUM
add di, 6
mov ax, cx

call WRD_TO_HEX
mov al, bl
call BYTE_TO_HEX
sub di, 2
mov [di], ax
mov dx, offset OUT_SER_NUM
CALL PRINT

pop DX
pop CX
pop BX
pop AX
ret
SYSTEM_VERSION ENDP
```

```
;-----
```

```
PRINT_NUMBER_DEC PROC
```

```
    push AX
```

```
    push BX
```

```
    sub BX, BX
```

```
    mov Bl, 10
```

```
    mov AH, 0
```

```
    div Bl
```

```
    mov DX, AX
```

```
    add DL, '0'
```

```
    sub AX, AX
```

```
    mov AH, 02h
```

```
    int 21h
```

```
    mov DL, DH
```

```
    add DL, '0'
```

```
    sub AX, AX
```

```
    mov AH, 02h
```

```
    int 21h
```

```
    pop BX
```

```
    pop AX
```

```
    ret
```

```
PRINT_NUMBER_DEC ENDP
```

```
;-----
```

```
BEGIN:
```

```
    call TYPE_PC
```

```
    call SYSTEM_VERSION
```

```
    xor AX, AX
```

```
    mov AH, 4Ch
```

```
    int 21h
```

```
LAB1 ENDS
```

```
END START
```

## ПРИЛОЖЕНИЕ Б

### КОД ДЛЯ «ХОРОШЕГО».EXE МОДУЛЯ

```
LAB1STACK SEGMENT STACK
    dw 100h
LAB1STACK ENDS

DATA SEGMENT

    TYPE_OF_PC db "Type of PC: $"
    PC db "PC$"
    PC_XT db "PC/XT$"
    AT db "AT$"
    PS2_30 db "PS2 model 30$"
    PS2_50 db "PS2 model 50 or 60$"
    PS2_80 db "PS2 model 80$"
    PCjr db "PCjr$"
    PC_Con db "PC convertable$"
    UNKNOWN_TYPE db " Unknown$"
    OUT_UNKNOWN db "      $"
    VER_NUM db 13, 10, "Version number: $"

    OUT_VER_NUM db " .   $"

    OEM_NUM db 13, 10, "OEM: $"
    OUT_OEM db "        $"
    SER_NUM db 13, 10, "Serial number: $"
    OUT_SER_NUM DB "                  $""

DATA ENDS

LAB1CODE SEGMENT
ASSUME CS: LAB1CODE, DS:DATA, ES:NOTHING, SS:LAB1STACK

;-----  
  
TYPE_PC PROC near  
  
    push ax  
    push dx  
  
    mov dx, offset TYPE_OF_PC
```

```

call PRINT

mov ax, 0F000h
mov es, ax
mov al, es:[0FFEh]

mov al, 7h

cmp al, 0FFh
    je W_PC
cmp al, 0FEh
    je W_PC_XT
cmp al, 0FBh
    je W_PC_XT
cmp al, 0FCh
    je W_AT
cmp al, 0FAh
    je W_PS2_30
cmp al, 0FCCh
    je W_PS2_50
cmp al, 0F8h
    je W_PS2_80
cmp al, 0FDh
    je W_PCjr
cmp al, 0F9h
    je W_PC_Con

mov di, offset OUT_UNKNOWN
call BYTE_TO_HEX
mov [di], ax
mov dx, offset OUT_UNKNOWN
CALL PRINT

mov DX, offset UNKNOWN_TYPE
jmp ESC_PROC

W_PC:
    mov DX, offset PC
    jmp ESC_PROC

W_PC_XT:
    mov DX, offset PC_XT

```

```

        jmp ESC_PROC

W_AT:
    mov DX, offset AT
    jmp ESC_PROC

W_PS2_30:
    mov DX, offset PS2_30
    jmp ESC_PROC

W_PS2_50:
    mov DX, offset PS2_50
    jmp ESC_PROC

W_PS2_80:
    mov DX, offset PS2_80
    jmp ESC_PROC

W_PCjr:
    mov DX, offset PCjr
    jmp ESC_PROC

W_PC_Con:
    mov DX, offset PC_Con

ESC_PROC:
    call PRINT

    pop dx
    pop ax

    ret
TYPE_PC ENDP

;-----


PRINT PROC near

    push ax
    sub ax, ax
    mov ah, 9h
    int 21h
    pop ax

```

```

    ret
PRINT ENDP

;-----

WRD_TO_HEX PROC near
    push     BX
    mov      BH, AH
    call     BYTE_TO_HEX
    mov      [DI], AH
    dec      DI
    mov      [DI], AL
    dec      DI
    mov      AL, BH
    call     BYTE_TO_HEX
    mov      [DI], AH
    dec      DI
    mov      [DI], AL
    pop     BX
    ret
WRD_TO_HEX ENDP

;-----


TETR_TO_HEX PROC near
    and     AL, 0Fh
    cmp     AL, 09
    jbe     NEXT
    add     AL, 07
NEXT:   add     AL, 30h
    ret
TETR_TO_HEX ENDP

;-----


BYTE_TO_HEX PROC near
    push     CX
    mov      AH, AL
    call     TETR_TO_HEX
    xchg     AL, AH
    mov      CL, 4
    shr      AL, CL

```

```
        call      TETR_TO_HEX ;â AL ñòàðøàÿ öèôðà
        pop       CX           ;â AH ìëàäøàÿ
        ret
BYTE_TO_HEX    ENDP
```

```
;-----
```

```
BYTE_TO_DEC    PROC  near
                push   CX
                push   DX
                xor    AH, AH
                xor    DX, DX
                mov    CX, 10
loop_bd:     div    CX
                or     DL, 30h
                mov    [SI], DL
                dec    si
                xor    DX, DX
                cmp    AX, 10
                jae    loop_bd
                cmp    AL, 00h
                je     end_1
                or     AL, 30h
                mov    [SI], AL

end_1:       pop    DX
                pop    CX
                ret
BYTE_TO_DEC    ENDP
```

```
;-----
```

```
SYSTEM_VERSION PROC
                push  ax
                push  bx

                sub  ax, ax
                mov  ah, 30h
                int  21h

                mov  dx, offset VER_NUM
```

```
call PRINT

push ax
mov si, offset OUT_VER_NUM
inc si
call BYTE_TO_DEC
pop ax
mov al, ah
add si, 3
call BYTE_TO_DEC

mov dx, offset OUT_VER_NUM
call print

mov DX, offset OEM_NUM
call PRINT

mov si, offset OUT_OEM
add si, 2
mov al, bh
call BYTE_TO_DEC
mov dx, OFFSET OUT_OEM
call PRINT

mov dx, offset SER_NUM
call PRINT

mov di, offset OUT_SER_NUM
add di, 6
mov ax, cx

call WRD_TO_HEX
mov al, bl
call BYTE_TO_HEX
sub di, 2
mov [di], ax
mov dx, offset OUT_SER_NUM
CALL PRINT

pop bx
pop ax
```

```
    ret  
SYSTEM_VERSION ENDP
```

```
;-----
```

```
PRINT_NUMBER_DEC PROC
```

```
    push AX  
    push BX
```

```
    sub BX, BX  
    mov Bl, 10  
    mov AH, 0  
    div Bl  
    mov DX, AX
```

```
    add DL, '0'  
    sub AX, AX  
    mov AH, 02h  
    int 21h
```

```
    mov DL, DH
```

```
    add DL, '0'  
    sub AX, AX  
    mov AH, 02h  
    int 21h
```

```
    pop BX  
    pop AX  
    ret
```

```
PRINT_NUMBER_DEC ENDP
```

```
;-----
```

```
MAIN PROC FAR
```

```
    push AX  
    sub AX, AX  
    mov ax, data  
    mov ds, ax  
    POP AX
```

```
    call TYPE_PC  
    call SYSTEM_VERSION
```

```
sub AX, AX  
mov AH, 4Ch  
int 21h
```

```
MAIN ENDP  
LAB1CODE ENDS  
END MAIN
```