

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

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

Студент гр. 9383

\_\_\_\_\_

Орлов Д.С.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2021

## Постановка задачи

Шаг 1. Написать текст исходного .COM модуля, который определяет тип РС и версию системы. Результатом выполнения этого шага будет «хороший» .COM модуль, а также необходимо построить «плохой» .EXE, полученный из исходного текста для .COM.

Шаг 2. Написать текст исходного .EXE модуля, который выполняет те же функции, что и модуль в шаге 1 и отладить его.

Шаг 3. Сравнить исходные тексты для .COM и .EXE модулей. Ответить на вопросы «Отличия исходных текстов COM и EXE программ».

Шаг 4. Запустить FAR и открыть файл загрузочного модуля .COM и файл «плохого» .EXE в шестнадцатеричном виде. Затем открыть файл загрузочного модуля «хорошего» .EXE и сравнить его с предыдущими файлами. Ответить на контрольные вопросы «Отличия форматов файлов COM и EXE модулей».

Шаг 5. Открыть отладчик TD.EXE и загрузить .CO. Ответить на контрольные вопросы «Загрузка COM модуля в основную память». Представить в отчете план загрузки модуля .COM в основную память.

Шаг 6. Открыть отладчик TD.EXE и загрузить «хороший» .EXE. Ответить на контрольные вопросы «Загрузка «хорошего» EXE в основную память».

Шаг 7. Оформить отчет в соответствии с требованиями. Привести скриншоты. Для файлов их вид в шестнадцатеричном виде, для загрузочных модулей — в отладчике.

Таблица 1. Используемые процедуры

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа ASCII
BYTE_TO_HEX	Перевод байта в AL в два символа шестн. числа AX
BYTE_TO_WRD	Запись AL в строку DEC_NUMBER
PC_TYPE	Вывод типа ПК на экран
OS_VERSION	Вывод версии ОС на экран

## Выполнение шагов

**Шаг 1.** В процессе выполнения лабораторной работы была разработана программа для модуля .COM, исходный код которой приведен в приложении А. После отладки был получен «плохой» .EXE модуль, а из него был получен «хороший» .COM модуль.

```
I:\>com.com
Type is AT
Version: 5.0
DEM: 0
User: 000000
```

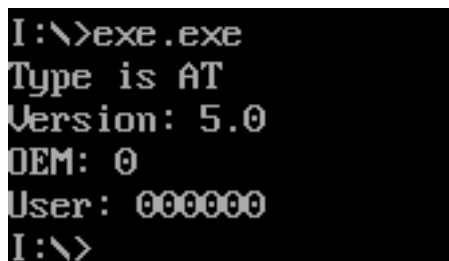
Рис. 1. Пример работы модуля .COM

Также, был создан «плохой» .EXE, который работает неверно:

```
I:\>com.exe
5 0
0
000000
is PC
00Type is PC
00Type is PC
00Type is PC
00Type
```

Рис. 2. Пример работы «плохого» .EXE

**Шаг 2.** В процессе выполнения лабораторной работы была разработана программа для «хорошего» модуля .EXE, исходный код которой приведен в приложении Б.



```
I:\>exe.exe
Type is AT
Version: 5.0
OEM: 0
User: 000000
I:\>
```

Рис. 3. Пример работы «хорошего» .EXE

**Шаг 3.** Ответим на вопросы по теме «Отличия исходных текстов .EXE и .COM программ».

1. Сколько сегментов должна содержать COM-программа?

COM-программа должна содержать только один сегмент.

2. EXE-программа?

Такая программа должна содержать не менее одного сегмента (один и более).

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

В COM программе обязательно должна быть прописана директива `org 100h`, т. к. первые 100h байт занимает PSP. Это позволит сместить все относительные адреса на 100h байт. Также необходимо прописать (`ASSUME CS:MYSEGMENT, DS:MYSEGMENT, ES:NOTHING, SS:NOTHING`), чтобы сегмент данных и сегмент кода указывали на один и тот же сегмент.

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

Нельзя использовать команды вида `<mov register, seg segname>`. Т. к. в COM-программе отсутствует таблица настроек.

#### Шаг 4. Шестнадцатеричное представление:

0000000000:	E9 EF 01 54 79 70 65 20	69 73 20 50 43 0D 0A 24	йн@Type is PC)а\$
0000000010:	54 79 70 65 20 69 73 20	50 43 2F 58 54 0D 0A 24	Type is PC/XT)а\$
0000000020:	54 79 70 65 20 69 73 20	41 54 0D 0A 24 54 79 70	Type is AT)а\$Typ
0000000030:	65 20 69 73 20 50 53 32	20 6D 6F 64 65 6C 20 33	e is PS2 model 3
0000000040:	30 0D 0A 24 54 79 70 65	20 69 73 20 50 53 32 20	0)а\$Type is PS2
0000000050:	6D 6F 64 65 6C 20 38 30	0D 0A 24 54 79 70 65 20	model 80)а\$Type
0000000060:	69 73 20 50 43 20 43 6F	6E 76 65 72 74 69 62 6C	is PC Convertibl
0000000070:	65 0D 0A 24 54 79 70 65	20 69 73 20 50 43 6A 72	e)а\$Type is PCjr
0000000080:	0D 0A 24 45 52 52 4F 52	3A 20 4E 6F 20 74 79 70	)а\$ERROR: No typ
0000000090:	65 20 69 6E 20 74 61 62	6C 65 3A 20 0D 0A 24 56	e in table: )а\$V
00000000A0:	65 72 73 69 6F 6E 3A 20	20 2E 20 20 0D 0A 24 4F	ersion: . )а\$O
00000000B0:	45 4D 3A 20 20 0D 0A 24	55 73 65 72 3A 20 20 20	EM: )а\$User:
00000000C0:	20 20 20 20 20 24 D0 97	D0 BD D0 B0 D1 87 D0 B5	\$P-PSP°C†Pμ
00000000D0:	D0 BD D0 B8 D0 B5 20 D1	80 D0 B5 D0 B3 D0 B8 D1	PSPëPμ сЪPμPіPëC
00000000E0:	81 D1 82 D1 80 D0 B0 20	41 58 3D 20 0D 0A 24 24	ѓC,сЪP° AX= )а\$
00000000F0:	0F 3C 09 76 02 04 07 04	30 C3 51 8A E0 E8 EF FF	α<ov0♦♦♦0ГQлaипя
0000000100:	86 C4 B1 04 D2 E8 E8 E6	FF 59 C3 53 8A FC E8 E9	†Д±♦ТиижяYГSлЬий
0000000110:	FF 88 25 4F 88 05 4F 8A	C7 E8 DE FF 88 25 4F 88	я€%O€†Oл3иl0я€%O€
0000000120:	05 5B C3 51 52 32 E4 33	D2 B9 0A 00 F7 F1 80 CA	†[ГQR2д3TN° ычсЪK
0000000130:	30 88 14 4E 33 D2 3D 0A	00 73 F1 3C 00 74 04 0C	0€ŃN3T=ы sс< t♦Q
0000000140:	30 88 04 5A 59 C3 B8 00	F0 8E C0 26 A0 FE FF 3C	0€♦ZYГë рђA& юя<
0000000150:	FF 74 22 3C FE 74 24 3C	FB 74 20 3C FC 74 22 3C	ят"<ют\$<ыт <ьт"<
0000000160:	FA 74 24 3C F8 74 26 3C	FD 74 2E 3C F9 74 24 BA	ьт\$<шт&<эт.<шт\$е
0000000170:	83 01 EB 79 90 BA 03 01	EB 73 90 BA 10 01 EB 6D	ѓ@лyђє♥@лsђє►@лm
0000000180:	90 BA 20 01 EB 67 90 BA	2D 01 EB 61 90 BA 44 01	ђє @лgђє-@лађєD@
0000000190:	EB 5B 90 BA 5B 01 EB 55	90 BA 74 01 EB 01 90 E8	л[ђє[@лUђєт@л@ђи
00000001A0:	4B 00 C3 B4 30 CD 21 50	BE 9F 01 83 C6 09 E8 72	K Гг0H!Pсц@ѓЖоиг
00000001B0:	FF 58 8A C4 83 C6 03 E8	69 FF BA 9F 01 E8 2D 00	яXлдѓЖ♥иіяєц@и-
00000001C0:	BE AF 01 83 C6 05 8A C7	E8 58 FF BA AF 01 E8 1C	sİ@ѓЖ♦л3иXяєİ@иL
00000001D0:	00 BF B8 01 83 C7 0B 8B	C1 E8 2F FF 8A C3 E8 19	іë@ѓ3д<Би/ялђи↓
00000001E0:	FF 83 EF 02 89 05 BA B8	01 E8 01 00 C3 B4 09 CD	яѓп0%†єë@и@ ГгoH
00000001F0:	21 C3 E8 51 FF E8 AB FF	32 C0 B4 4C CD 21	!ГиQяи«я2ArLH!

Рис. 4. Шестнадцатеричное представление модуля .COM

# Шестнадцатеричное представление «плохого» модуля .EXE:

0000000000: 4D 5A FE 00 03 00 00 00	20 00 00 00 FF FF 00 00	MZю ♥      яя
0000000010: 00 00 C1 A8 00 01 00 00	1E 00 00 00 01 00 00 00	БЁ ☹    ▲    ☹
00000001E0: 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	
0000000200: 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	
0000000220: 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	
0000000240: 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	
0000000260: 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	
0000000280: 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	
00000002A0: 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	
00000002C0: 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	
00000002E0: 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	
0000000300: E9 EF 01 54 79 70 65 20	69 73 20 50 43 0D 0A 24	йп@Type is PC№\$
0000000310: 54 79 70 65 20 69 73 20	50 43 2F 58 54 0D 0A 24	Type is PC/XT№\$
0000000320: 54 79 70 65 20 69 73 20	41 54 0D 0A 24 54 79 70	Type is AT№\$Typ
0000000330: 65 20 69 73 20 50 53 32	20 6D 6F 64 65 6C 20 33	e is PS2 model 3
0000000340: 30 0D 0A 24 54 79 70 65	20 69 73 20 50 53 32 20	0№\$Type is PS2
0000000350: 6D 6F 64 65 6C 20 38 30	0D 0A 24 54 79 70 65 20	model 80№\$Type
0000000360: 69 73 20 50 43 20 43 6F	6E 76 65 72 74 69 62 6C	is PC Convertibl
0000000370: 65 0D 0A 24 54 79 70 65	20 69 73 20 50 43 6A 72	e№\$Type is PCjr
0000000380: 0D 0A 24 45 52 52 4F 52	3A 20 4E 6F 20 74 79 70	№\$ERROR: No typ
0000000390: 65 20 69 6E 20 74 61 62	6C 65 3A 20 0D 0A 24 56	e in table: №\$V
00000003A0: 65 72 73 69 6F 6E 3A 20	20 2E 20 20 0D 0A 24 4F	ersion: . №\$0
00000003B0: 45 4D 3A 20 20 0D 0A 24	55 73 65 72 3A 20 20 20	EM: №\$User:
00000003C0: 20 20 20 20 20 24 D0 97	D0 BD D0 B0 D1 87 D0 B5	\$P-PSP°C†Pμ
00000003D0: D0 BD D0 B8 D0 B5 20 D1	80 D0 B5 D0 B3 D0 B8 D1	PSPëPμ сЪPμPīPëC
00000003E0: 81 D1 82 D1 80 D0 B0 20	41 58 3D 20 0D 0A 24 24	ѓC,сЪP° AX= №\$ \$
00000003F0: 0F 3C 09 76 02 04 07 04	30 C3 51 8A E0 E8 EF FF	о<ov0♦♦♦0ГQЉаипя
0000000400: 86 C4 B1 04 D2 E8 E8 E6	FF 59 C3 53 8A FC E8 E9	†Д±♦ТиижяYГSЉый
0000000410: FF 88 25 4F 88 05 4F 8A	C7 E8 DE FF 88 25 4F 88	я€%O€+OЉ3иЮя€%O€
0000000420: 05 5B C3 51 52 32 E4 33	D2 B9 0A 00 F7 F1 80 CA	♣[ГQR2дЗТWэ чсЪK
0000000430: 30 88 14 4E 33 D2 3D 0A	00 73 F1 3C 00 74 04 0C	0€ЉNЗТ=э sc< т♦♀
0000000440: 30 88 04 5A 59 C3 B8 00	F0 8E C0 26 A0 FE FF 3C	0€♦ZYГё рЪA& юя<
0000000450: FF 74 22 3C FE 74 24 3C	FB 74 20 3C FC 74 22 3C	ят"<ют\$<ыт <ьт"<
0000000460: FA 74 24 3C F8 74 26 3C	FD 74 2E 3C F9 74 24 BA	ьт\$<шт&<эт.<шт\$е
0000000470: 83 01 EB 79 90 BA 03 01	EB 73 90 BA 10 01 EB 6D	ѓ0луђе♥0лsђе►0лm
0000000480: 90 BA 20 01 EB 67 90 BA	2D 01 EB 61 90 BA 44 01	ђе 0лgђе-0лађеD0
0000000490: EB 5B 90 BA 5B 01 EB 55	90 BA 74 01 EB 01 90 E8	л[ђе[0лUђет0л0ђи
00000004A0: 4B 00 C3 B4 30 CD 21 50	BE 9F 01 83 C6 09 E8 72	K Гр0H!Pсy0ѓЖоир
00000004B0: FF 58 8A C4 83 C6 03 E8	69 FF BA 9F 01 E8 2D 00	яXЉдѓЖ♥иіяеу0и-
00000004C0: BE AF 01 83 C6 05 8A C7	E8 58 FF BA AF 01 E8 1C	sЇ0ѓЖ♣Љ3иXяеЇ0иL
00000004D0: 00 BF B8 01 83 C7 0B 8B	C1 E8 2F FF 8A C3 E8 19	їё0ѓЗѿ<Би/яЉГи↓
00000004E0: FF 83 EF 02 89 05 BA B8	01 E8 01 00 C3 B4 09 CD	яѓп0%♣её0и0 Гр0H
00000004F0: 21 C3 E8 51 FF E8 AB FF	32 C0 B4 4C CD 21	!ГиQяи«я2ArLH!

Рис. 5. Шестнадцатеричное представление «плохого» модуля .EXE

0000000000: 4D 5A E7 01 03 00 01 00	20 00 00 00 FF FF 00 00	MZ40♥ ☹
0000000010: 00 02 0E BB 00 00 2D 00	1E 00 00 00 01 00 07 01	0J¶ - ▲ ☹ •☹
0000000020: 2D 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	-
000000003F0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000400: 54 79 70 65 20 69 73 20	50 43 0D 0A 24 54 79 70	Type is PC.☹\$Typ
00000000410: 65 20 69 73 20 50 43 2F	58 54 0D 0A 24 54 79 70	e is PC/XT.☹\$Typ
00000000420: 65 20 69 73 20 41 54 0D	0A 24 54 79 70 65 20 69	e is AT.☹\$Type i
00000000430: 73 20 50 53 32 20 6D 6F	64 65 6C 20 33 30 0D 0A	s PS2 model 30.☹
00000000440: 24 54 79 70 65 20 69 73	20 50 53 32 20 6D 6F 64	\$Type is PS2 mod
00000000450: 65 6C 20 38 30 0D 0A 24	54 79 70 65 20 69 73 20	el 80.☹\$Type is
00000000460: 50 43 20 43 6F 6E 76 65	72 74 69 62 6C 65 0D 0A	PC Convertible.☹
00000000470: 24 54 79 70 65 20 69 73	20 50 43 6A 72 0D 0A 24	\$Type is PCjr.☹\$
00000000480: 45 52 52 4F 52 3A 20 4E	6F 20 74 79 70 65 20 69	ERROR: No type i
00000000490: 6E 20 74 61 62 6C 65 3A	20 0D 0A 24 56 65 72 73	n table: ☹\$Vers
000000004A0: 69 6F 6E 3A 20 20 2E 20	20 0D 0A 24 4F 45 4D 3A	ion: . ☹\$OEM:
000000004B0: 20 20 0D 0A 24 55 73 65	72 3A 20 20 20 20 20 20	☹\$User:
000000004C0: 20 20 24 00 00 00 00 00	00 00 00 00 00 00 00 00	\$
000000004D0: E9 03 01 24 0F 3C 09 76	02 04 07 04 30 C3 51 8A	щ♥0\$α<ov0♦•♦0 QK
000000004E0: E0 E8 EF FF 86 C4 B1 04	D2 E8 E8 E6 FF 59 C3 53	ршя Ж-☹☹☹шщ Y S
000000004F0: 8A FC E8 E9 FF 88 25 4F	88 05 4F 8A C7 E8 DE FF	К№шщ И%ОИ♦OK  ш
00000000500: 88 25 4F 88 05 5B C3 51	52 32 E4 33 D2 B9 0A 00	И%ОИ♦[ QR2ф3-  ☹
00000000510: F7 F1 80 CA 30 88 14 4E	33 D2 3D 0A 00 73 F1 3C	ÿëA-0И¶N3-☹ së<
00000000520: 00 74 04 0C 30 88 04 5A	59 C3 B8 00 F0 8E C0 26	t♦φ0И♦ZY  ☹EO L&
00000000530: A0 FE FF 3C FF 74 22 3C	FE 74 24 3C FB 74 20 3C	a■ < t"<■t\$<√t <
00000000540: FC 74 22 3C FA 74 24 3C	F8 74 26 3C FD 74 2E 3C	№t"<·t\$<°t&<#t.<
00000000550: F9 74 24 BA 80 00 EB 79	90 BA 00 00 EB 73 90 BA	·t\$  A ыyP   ыsP
00000000560: 0D 00 EB 6D 90 BA 1D 00	EB 67 90 BA 2A 00 EB 61	♪ ымP  ☹ ыgP  * ыa
00000000570: 90 BA 41 00 EB 5B 90 BA	58 00 EB 55 90 BA 71 00	P  A ы[P  X ыUP  q
00000000580: EB 01 90 E8 4B 00 C3 B4	30 CD 21 50 BE 9C 00 83	ы0PшК  0=!P-Ь Г
00000000590: C6 09 E8 72 FF 58 8A C4	83 C6 03 E8 69 FF BA 9C	└ошг ХК-Г└♥wi   b
000000005A0: 00 E8 2D 00 BE AC 00 83	C6 05 8A C7 E8 58 FF BA	ш- -м Г└♣K  шX
000000005B0: AC 00 E8 1C 00 BF B5 00	83 C7 0B 8B C1 E8 2F FF	м шЛ γ└ Г  оЛ-ш/
000000005C0: 8A C3 E8 19 FF 83 EF 02	89 05 BA B5 00 E8 01 00	К└ш↓ Гя0Й♣  └ ш0
000000005D0: C3 B4 09 CD 21 C3 B8 20	00 8E D8 E8 4C FF E8 A6	└o=!└ О└шL шж
000000005E0: FF 32 C0 B4 4C CD 21		2└└L=!

Рис. 6. Шестнадцатеричное представление «хорошего» модуля .EXE

Отличия .COM от «плохого» .EXE в том, что в .COM программа идет без заголовков. В «плохом» .EXE есть неправильный заголовок, потому что он создавался по COM-программе.

В «хорошем» .EXE заголовок корректный, также EXE-файл стал немного больше в размерах.

### **Отличия форматов файлов COM и EXE модулей**

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

COM-файл содержит данные и машинные программы. Код располагается по адресу 0h, но при загрузке модуля устанавливается смещение 100h.

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

«Плохой» EXE содержит EXE-маркер (байты MZ), заголовок и таблицу настроек. В заголовке содержится различная служебная информация для загрузчика. В таблице настроек содержатся записи в формате сегмент: смещение. К смещениям в загрузочном модуле, на которые указывают значения в таблице, после загрузки программы в память будет прибавлен сегментный адрес. Код и данные располагаются в одном сегменте. Код располагается с 300h байта. С адреса 2 располагаются заголовок и таблица настроек после него.

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

В «хорошем» EXE код, данные и стек разделены на сегменты. С адреса 2 в «хорошем» EXE располагается заголовок, а после него идет таблица настроек. Код начинается с 400h, т.к. размер заголовка и таблицы — 200h, и размер стека — 200h.



## Шаг 5.

### Загрузка COM-модуля в основную память

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

ОС ищет подходящее по размеру место в оперативной памяти для COM-модуля. Далее она помещает в это место PSP, а по смещению в 100h помещает считанный с диска модуль.

2. Что располагается с адреса 0?

С адреса 0 располагается PSP размером в 100h байт.

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

CS, ES, DS и SS указывают на PSP (48DDh), а SP указывает на конец сегмента (FFFEh).

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

Стек определяется автоматически при ассемблировании и линковке. Стек, теоретически, занимает всю область сегмента. SP указывает на конец стека (FFFEh), а SS указывает на начало стека (48DDh).

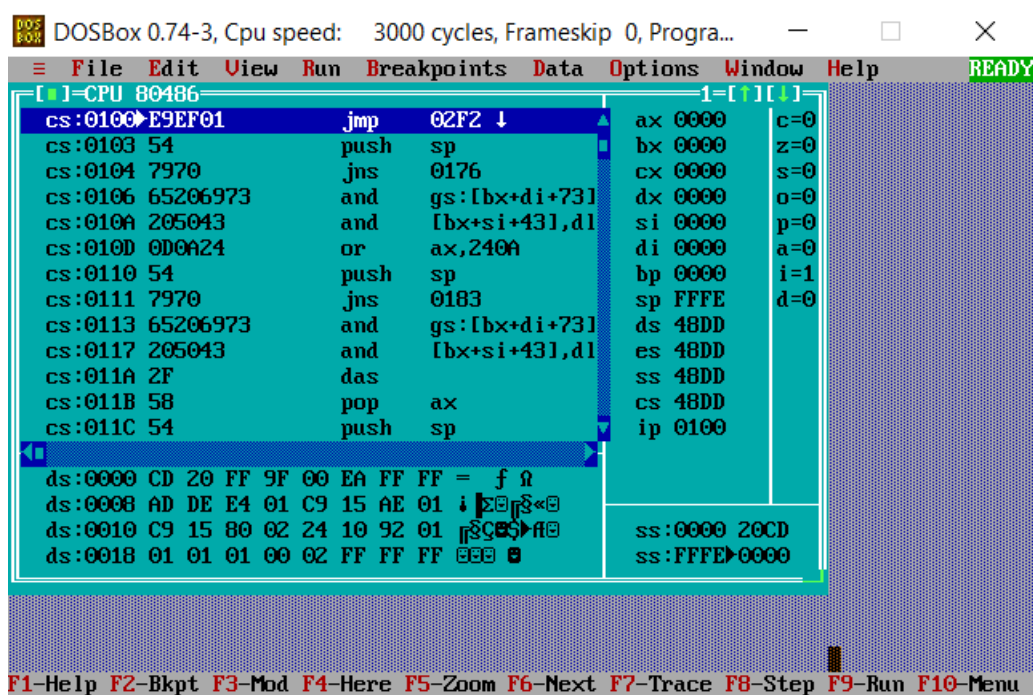


Рис. 7. Интерфейс TD.EXE (.COM)

## Шаг 6.

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

1. Как загружается «хороший» EXE? Какие значения имеют сегментные регистры?

Подготовка загрузки аналогична .COM. EXE загружается по смещению 100h от PSP. В процессе загрузки считывается информация EXE-заголовка и выполняется перемещение адресов сегментов. CS указывает на начало сегмента команд (491Ah). В IP загружается смещение точки входа в программу.

2. На что указывают регистры DS и ES?

На начало сегмента PSP (48DDh).

3. Как определяется стек?

Стек определяется на основе директивы .stack и SP указывает на конец сегмента стека.

4. Как определяется точка входа?

Точка входа определяется с помощью директивы END.

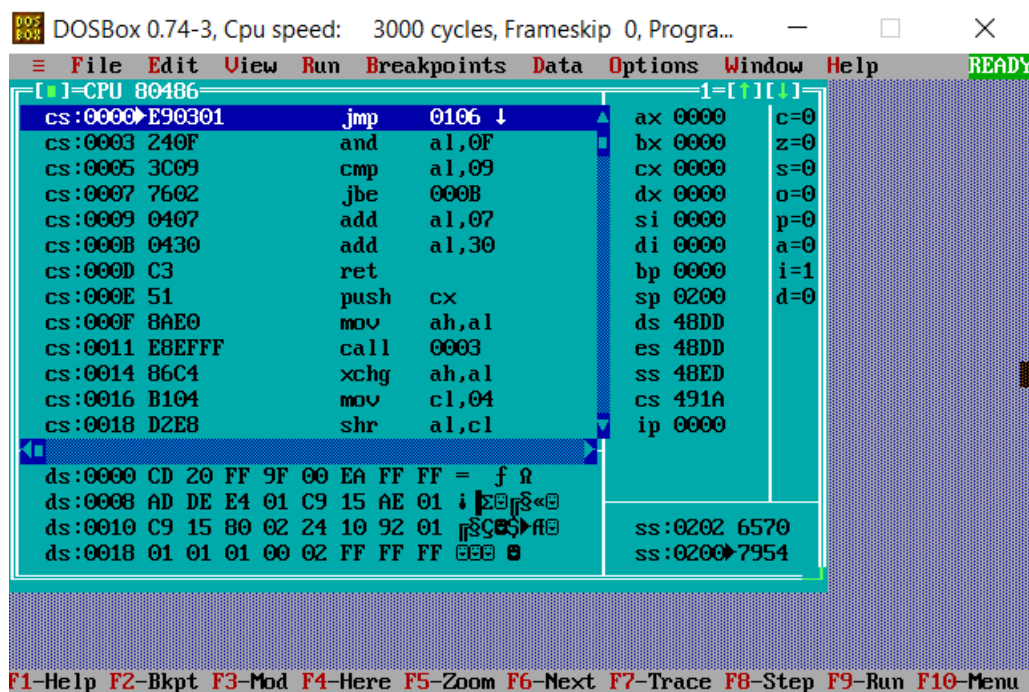


Рис. 8. Интерфейс TD.EXE (.EXE)

## **Заключение**

В процессе выполнения лабораторной работы были изучены структурные отличия .COM и .EXE модулей и получены навыки работы с Far Manager и отладчиком TD.EXE.

## Приложение А.

```
; Шаблон текста программы на ассемблере для модуля типа .COM
TESTPC SEGMENT
ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
ORG 100H
START: JMP BEGIN
; ДАННЫЕ
```

```
PC_T db 'Type is PC',0DH,0AH,'$'
PC_XT_T db 'Type is PC/XT',0DH,0AH,'$'
AT_TY db 'Type is AT',0DH,0AH,'$'
PS30_T db 'Type is PS2 model 30',0DH,0AH,'$'
PS80_T db 'Type is PS2 model 80',0DH,0AH,'$'
PCCON_T db 'Type is PC Convertible',0DH,0AH,'$'
PCjr_T db 'Type is PCjr',0DH,0AH,'$'
NO_T db 'ERROR: No type in table: ',0DH,0AH,'$'
```

```
VERSION db 'Version: . ',0DH,0AH,'$'
OEM db 'OEM: ',0DH,0AH,'$'
USER db 'User:    '$'
```

```
STRING db 'Значение регистра AX= ',0DH,0AH,'$'
;ПРОЦЕДУРЫ
```

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

```
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
```

```
; байт в AL переводится в два символа шестн. числа в AX
```

```
push CX
```

```
mov AH,AL
```

```
call TETR_TO_HEX
```

```
xchg AL,AH
```

```
mov CL,4
```

```
shr AL,CL
```

```
call TETR_TO_HEX ;в AL старшая цифра
```

```
pop CX ;в AH младшая
```

```
ret
```

```
BYTE_TO_HEX ENDP
```

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

```
WRD_TO_HEX PROC near
```

```
;перевод в 16 с/с 16-ти разрядного числа
```

```
; в AX - число, DI - адрес последнего символа
```

```

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

```

```

;-----
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
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

```

```

;-----

```

```

; КОД
PC_TYPE PROC near

```

```

    mov AX, 0f000h ;получаем тип ПК
    mov ES, AX
    mov AL, es:[0fffh]

```

```

    cmp AL, 0FFh
    je pc

```

```
cmp AL, 0FEh
je pc_xt
```

```
cmp AL, 0FBh
je pc_xt
```

```
cmp AL, 0FCh
je at_t
```

```
cmp AL, 0FAh
je ps30
```

```
cmp AL, 0F8h
je ps80
```

```
cmp AL, 0FDh
je pcjr
```

```
cmp AL, 0F9h
je pccon
```

```
mov dx, offset NO_T
jmp WRITE_STRING
```

```
pc:
    mov dx, offset PC_T
    jmp WRITE_STRING
pc_xt:
    mov dx, offset PC_XT_T
    jmp WRITE_STRING
at_t:
    mov dx, offset AT_TY
    jmp WRITE_STRING
ps30:
    mov dx, offset PS30_T
    jmp WRITE_STRING
ps80:
    mov dx, offset PS80_T
    jmp WRITE_STRING
pccon:
    mov dx, offset PCCON_T
    jmp WRITE_STRING
pcjr:
    mov dx, offset PCjr_T
    jmp WRITE_STRING_T
```

```
WRITE_STRING_T:
    call WRITE_STRING
    ret
```

PC\_TYPE ENDP

OS\_VERSION PROC near

```
        MOV AH, 30h
INT 21h
push AX

mov SI, offset VERSION
add SI, 9
call BYTE_TO_DEC
pop AX
mov AL, AH
add SI, 3
call BYTE_TO_DEC
mov DX, offset VERSION
call WRITE_STRING

mov SI, offset OEM
add SI, 5
mov AL, BH
call BYTE_TO_DEC
mov DX, offset OEM
call WRITE_STRING

mov DI, offset USER
add DI, 11
mov AX, CX
call WRD_TO_HEX
mov AL, BL
call BYTE_TO_HEX
sub DI, 2
mov [DI], AX
mov DX, offset USER
call WRITE_STRING
ret
```

OS\_VERSION ENDP

WRITE\_STRING PROC near

```
; Вывод строки текста из поля STRING
mov AH, 09h
int 21h
ret
```

WRITE\_STRING ENDP

BEGIN:

```
call PC_TYPE
call OS_VERSION
```

```
; Выход в DOS
xor AL,AL
mov AH,4Ch
int 21H
TESTPC ENDS
END START ;конец модуля, START - точка входа
```



## Приложение Б.

Astack SEGMENT STACK

DB 512 DUP (?)

Astack ENDS

DATA SEGMENT

; Данные

PC\_T db 'Type is PC',0DH,0AH,'\$'

PC\_XT\_T db 'Type is PC/XT',0DH,0AH,'\$'

AT\_TY db 'Type is AT',0DH,0AH,'\$'

PS30\_T db 'Type is PS2 model 30',0DH,0AH,'\$'

PS80\_T db 'Type is PS2 model 80',0DH,0AH,'\$'

PCCON\_T db 'Type is PC Convertible',0DH,0AH,'\$'

PCjr\_T db 'Type is PCjr',0DH,0AH,'\$'

NO\_T db 'ERROR: No type in table: ',0DH,0AH,'\$'

VERSION db 'Version: . ',0DH,0AH,'\$'

OEM db 'OEM: ',0DH,0AH,'\$'

USER db 'User: \$'

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:Astack

START: JMP BEGIN

; Процедуры

;-----

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

;байт в AL переводится в два символа шест. числа в AX

push CX

mov AH,AL

call TETR\_TO\_HEX

xchg AL,AH

mov CL,4

shr AL,CL

call TETR\_TO\_HEX ;в AL старшая цифра

pop CX ;в AH младшая

ret

BYTE\_TO\_HEX ENDP

;-----

WRD\_TO\_HEX PROC near

;перевод в 16 с/с 16-ти разрядного числа

; в AX - число, DI - адрес последнего символа

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

;-----

BYTE\_TO\_DEC PROC near

; перевод в 10с/с, SI - адрес поля младшей цифры

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

;-----

PC\_TYPE PROC near

mov AX, 0f000h ;получаем тип ПК

mov ES, AX

mov AL, es:[0fffh]

cmp AL, 0FFh

je pc

cmp AL, 0FEh

je pc\_xt

```
cmp AL, 0FBh
je pc_xt
```

```
cmp AL, 0FCh
je at_t
```

```
cmp AL, 0FAh
je ps30
```

```
cmp AL, 0F8h
je ps80
```

```
cmp AL, 0FDh
je pcjr
```

```
cmp AL, 0F9h
je pccon
```

```
mov dx, offset NO_T
jmp WRITE_STRING
```

```
pc:
```

```
    mov dx, offset PC_T
    jmp WRITE_STRING
```

```
pc_xt:
```

```
    mov dx, offset PC_XT_T
    jmp WRITE_STRING
```

```
at_t:
```

```
    mov dx, offset AT_TY
    jmp WRITE_STRING
```

```
ps30:
```

```
    mov dx, offset PS30_T
    jmp WRITE_STRING
```

```

ps80:
        mov dx, offset PS80_T
        jmp WRITE_STRING

pccon:
        mov dx, offset PCCON_T
        jmp WRITE_STRING

pcjr:
        mov dx, offset PCjr_T
        jmp WRITE_STRING_T

```

```

WRITE_STRING_T:
        call WRITE_STRING
        ret

```

```
PC_TYPE ENDP
```

```
OS PROC near
```

```

;версия
mov ah, 30h
int 21h
push ax
mov si, offset VERSION
add si, 9      ;смещение
call BYTE_TO_DEC
pop ax
mov al, ah
add si, 3      ;смещение
call BYTE_TO_DEC
mov dx, offset VERSION
call WRITE_STRING

```

```

;серийный номер OEM
mov si, offset OEM
add si, 5      ;смещение
mov al, bh

```

```
call BYTE_TO_DEC
mov dx, offset OEM
call WRITE_STRING
```

```
;серийный номер пользователя
```

```
mov di, offset USER
add di, 11    ;смещение
mov ax, cx
call WRD_TO_HEX
mov al, bl
call BYTE_TO_HEX
sub di, 2
mov [di], ax
mov dx, offset USER
call WRITE_STRING
ret
```

```
OS ENDP
```

```
WRITE_STRING PROC near
```

```
; Вывод строки текста из поля STRING
```

```
mov AH,09h
int 21h
ret
```

```
WRITE_STRING ENDP
```

```
; Код
```

```
BEGIN:
```

```
mov AX, DATA
mov DS, AX
call PC_TYPE
call OS
```

```
xor AL,AL
mov AH,4Ch
```

int 21H

CODE ENDS

END START; конец модуля, START - точка выхода