

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. 8383

Бессуднов Г. И.

Преподаватель

Ефремов М. А.

Санкт-Петербург

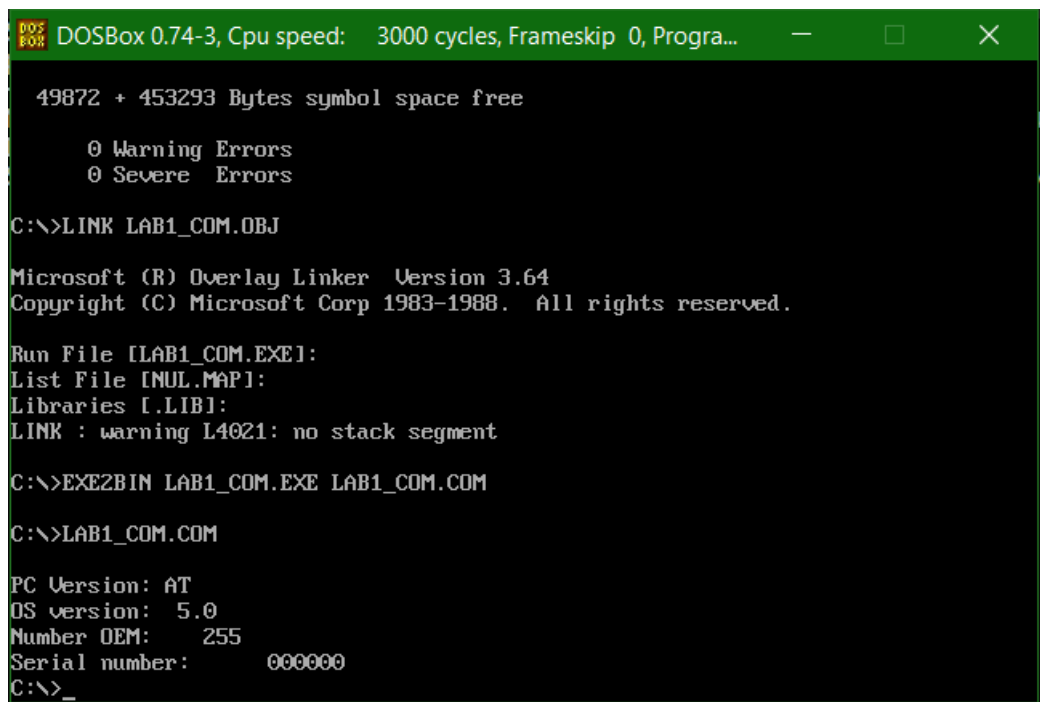
2020

Цель работы.

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

Ход работы.

Был написан код исходного **.COM** модуля, который определяет тип РС и версию системы. Код программы представлен в приложении А. Далее был построен **.COM** модуль и «плохой» **.EXE** модуль. Результаты выполнения программ показаны на рис. 1 и рис. 2 соответственно. В ходе линковки **.EXE** модуля было выдано предупреждение об отсутствии стека.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

49872 + 453293 Bytes symbol space free

    0 Warning Errors
    0 Severe Errors

C:\>LINK LAB1_COM.OBJ

Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Run File [LAB1_COM.EXE]:
List File [NUL.MAP]:
Libraries [LIB1]:
LINK : warning L4021: no stack segment

C:\>EXE2BIN LAB1_COM.EXE LAB1_COM.COM

C:\>LAB1_COM.COM

PC Version: AT
OS version: 5.0
Number OEM: 255
Serial number: 0000000
C:\>_
```

Рисунок 1 – Результат выполнения **.COM** модуля

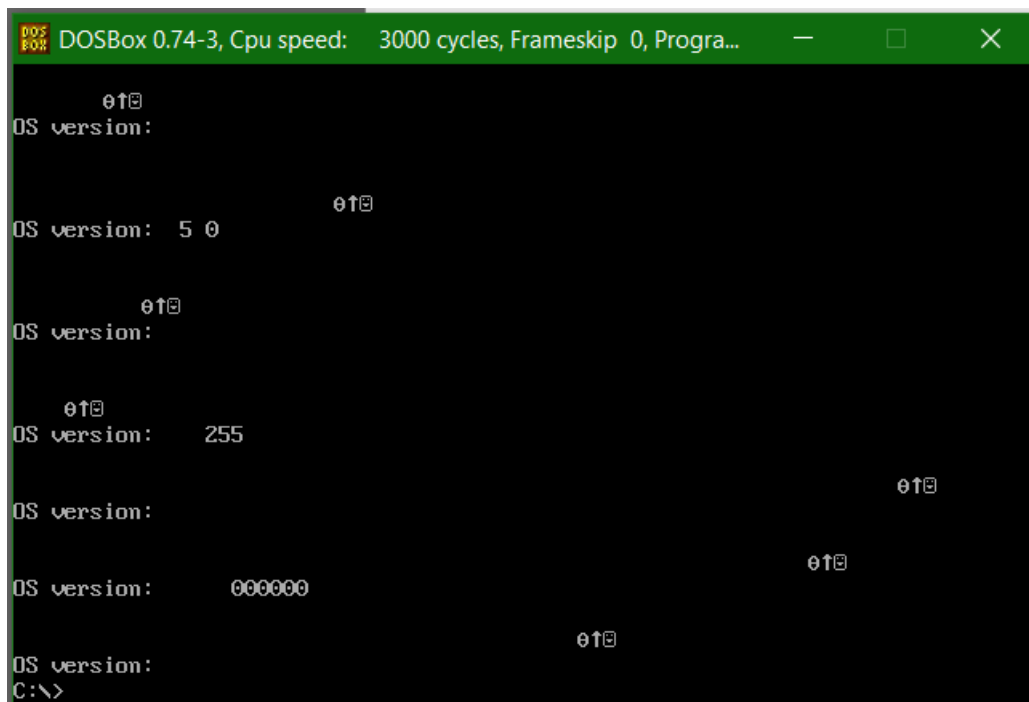


Рисунок 2 – Результат выполнения «плохого» **.EXE** модуля

Далее был написан текст для «хорошего» **.EXE** модуля. Код программы представлен в приложении Б. Результат выполнения «хорошего» **.EXE** модуля представлен на рис.3

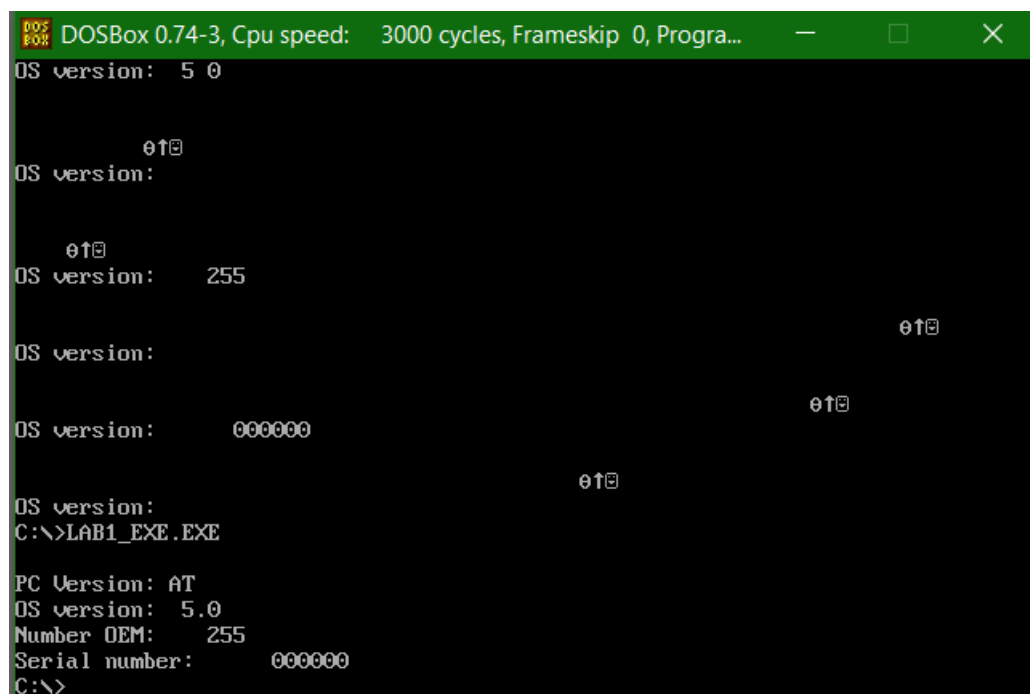


Рисунок 3 – Результат выполнения «хорошего» **.EXE** модуля

Видно, что результаты выполнения «хорошего» .EXE модуля и .COM модуля совпадают. На рис. 4-6 представлены файлы загрузочных модулей в шестнадцатеричном виде.

0000000000:	E9 18 01 0D 0A 4F 53 20	76 65 72 73 69 6F 6E 3A	éíèJOS version:
0000000010:	20 24 20 20 2E 20 20 24	0D 0A 4E 75 6D 62 65 72	\$. \$Number
0000000020:	20 4F 45 4D 3A 20 24 20	20 20 20 20 20 24 0D 0A	OEM: \$ \$
0000000030:	53 65 72 69 61 6C 20 6E	75 6D 62 65 72 3A 20 24	Serial number: \$
0000000040:	20 20 20 20 20 20 20 20	20 20 20 20 24 0D 0A 50	\$P
0000000050:	43 20 56 65 72 73 69 6F	6E 3A 20 24 50 43 24 50	C Version: \$PC\$P
0000000060:	43 2F 58 54 24 41 54 24	50 53 32 20 6D 6F 64 65	C/XT\$AT\$PS2 mode
0000000070:	6C 20 33 30 24 50 53 32	20 6D 6F 64 65 6C 20 38	1 30\$PS2 model 8
0000000080:	30 24 50 43 6A 72 24 50	53 32 20 6D 6F 64 65 6C	0\$PCjr\$PS2 model
0000000090:	20 35 30 2F 36 30 24 50	43 20 43 6F 6E 76 65 72	50/60\$PC Conver
00000000A0:	74 69 62 6C 65 24 4E 6F	74 20 66 6F 75 6E 64 20	tible\$Not found
00000000B0:	20 20 20 20 20 20 20 20	20 24 50 B4 09 CD 21 58	\$P'oi!X
00000000C0:	C3 24 0F 3C 09 76 02 04	07 04 30 C3 51 8A E0 E8	A\$<ove♦♦0AQ\$àè
00000000D0:	EF FF 86 C4 B1 04 D2 E8	E8 E6 FF 59 C3 53 8A FC	îÿ†A±♦0èèæÿYASSü
00000000E0:	E8 E9 FF 88 25 4F 88 05	4F 8A C7 E8 DE FF 88 25	èéÿ~%0~†0\$Cèbÿ~%
00000000F0:	4F 88 05 5B C3 51 52 32	E4 33 D2 B9 0A 00 F7 F1	o~†[AQR2ä30¹ ÷ñ
0000000100:	80 CA 30 88 14 4E 33 D2	3D 0A 00 73 F1 3C 00 74	€E0~¶N30= sñ< t
0000000110:	04 0C 30 88 04 5A 59 C3	52 5A C3 B8 00 F0 8E C0	♦º0~♦ZYARZA ðŽA
0000000120:	26 A0 FE FF BA 4D 01 E8	90 FF 3C FF 74 2B 3C FE	& pÿ°Mè¶ÿ<ÿt+<p
0000000130:	74 2D 3C FB 74 29 3C FC	74 4F 3C FA 74 33 3C F8	t-<üt)<üt0<üt3<ø
0000000140:	74 35 3C FD 74 37 3C F9	74 39 BE A6 01 83 C6 10	t5<ÿt7<üt9%!èfA»
0000000150:	E8 79 FF BA A6 01 EB 3F	90 BA 5C 01 EB 39 90 BA	èÿÿ°!èè?¶°\èè9¶°
0000000160:	5F 01 EB 33 90 BA 65 01	EB 2D 90 BA 87 01 EB 27	_èè3¶°èèè-¶°†èè'
0000000170:	90 BA 68 01 EB 21 90 BA	75 01 EB 1B 90 BA 82 01	¶°hèè!¶°uèè-¶°°,è
0000000180:	EB 15 90 BA 97 01 EB 0F	90 B4 C0 CD 15 26 8A 47	èš¶°-èè*¶°Aİ\$&\$G
0000000190:	03 3C 00 74 D0 EB D4 E8	20 FF B4 30 CD 21 50 BA	♥< tðè0è ÿ'OI!P°
00000001A0:	03 01 E8 15 FF BE 12 01	46 E8 49 FF 58 8A C4 83	♥èèšÿ%!èFèIÿXSÀf
00000001B0:	C6 03 E8 40 FF BA 12 01	E8 FF FE BA 18 01 E8 F9	A♥èèÿ°!èèÿb°†èèu
00000001C0:	FE BE 27 01 83 C6 05 8A	C7 E8 29 FF BA 27 01 E8	b¾'èfA†\$Cè)ÿ°'èè
00000001D0:	E8 FE BA 2E 01 E8 E2 FE	BF 40 01 83 C7 0A 8B C1	èp°.èèâp¿èèfCè<A
00000001E0:	E8 FA FE 8A C3 E8 E4 FE	83 EF 02 89 05 BA 40 01	èùp\$Aèâpfiè%†°èè
00000001F0:	E8 C7 FE 32 C0 B4 4C CD	21	èçb2A LI!

Рисунок 4 – .COM модуль

0000000240:	0D 0A 4F 53 20 76 65 72	73 69 6F 6E 3A 20 24 20	OS version: \$
0000000250:	20 2E 20 20 24 0D 0A 4E	75 6D 62 65 72 20 4F 45	. \$Number OE
0000000260:	4D 3A 20 24 20 20 20 20	20 20 24 0D 0A 53 65 72	M: \$ \$Ser
0000000270:	69 61 6C 20 6E 75 6D 62	65 72 3A 20 24 20 20 20	ial number: \$
0000000280:	20 20 20 20 20 20 20 20	20 24 0D 0A 50 43 20 56	\$PC V
0000000290:	65 72 73 69 6F 6E 3A 20	24 50 43 24 50 43 2F 58	ersion: \$PC/X
00000002A0:	54 24 41 54 24 50 53 32	20 6D 6F 64 65 6C 20 33	T\$AT\$PS2 model 3
00000002B0:	30 24 50 53 32 20 6D 6F	64 65 6C 20 38 30 24 50	0\$PS2 model 80\$P
00000002C0:	43 6A 72 24 50 53 32 20	6D 6F 64 65 6C 20 35 30	Cjr\$PS2 model 50
00000002D0:	2F 36 30 24 50 43 20 43	6F 6E 76 65 72 74 69 62	/60\$PC Convertib
00000002E0:	6C 65 24 4E 6F 74 20 66	6F 75 6E 64 20 20 20 20	le\$Not found
00000002F0:	20 20 20 20 20 20 24 00	00 00 00 00 00 00 00 00	\$
0000000300:	50 B4 09 CD 21 58 C3 24	0F 3C 09 76 02 04 07 04	P`oI!X\$*<ove♦♦
0000000310:	30 C3 51 8A E0 E8 EF FF	86 C4 B1 04 D2 E8 E8 E6	0AQ\$æiÿ†Ä±♦èèè
0000000320:	FF 59 C3 53 8A FC E8 E9	FF 88 25 4F 88 05 4F 8A	ÿYASSüèèÿ`%0`+OS
0000000330:	C7 E8 DE FF 88 25 4F 88	05 5B C3 51 52 32 E4 33	Cèbÿ`%0`+[AQR2ä3
0000000340:	D2 B9 0A 00 F7 F1 80 CA	30 88 14 4E 33 D2 3D 0A	Ó' ÷ñ€E0`ñN30=
0000000350:	00 73 F1 3C 00 74 04 0C	30 88 04 5A 59 C3 52 5A	sn< t♦90`♦ZYARZ
0000000360:	C3 50 2B C0 B8 04 00 8E	D8 58 B8 00 F0 8E C0 26	AP+A ♦ ZØX ðZA&
0000000370:	A0 FE FF BA 4A 00 E8 87	FF 3C FF 74 2B 3C FE 74	bÿ°j è†ÿ<ÿt+<pt
0000000380:	2D 3C FB 74 29 3C FC 74	4F 3C FA 74 33 3C F8 74	-<ût)<ût0<ût3<øt
0000000390:	35 3C FD 74 37 3C F9 74	39 BE A3 00 83 C6 10 E8	5<ÿt7<ût9%f fÆ>è
00000003A0:	70 FF BA A3 00 EB 3F 90	BA 59 00 EB 39 90 BA 5C	pÿ°f è?°Y è9°\
00000003B0:	00 EB 33 90 BA 62 00 EB	2D 90 BA 84 00 EB 27 90	è3°°b è-°° „ è'°
00000003C0:	BA 65 00 EB 21 90 BA 72	00 EB 1B 90 BA 7F 00 EB	°e è!°°r è-°°è è
00000003D0:	15 90 BA 94 00 EB 0F 90	B4 C0 CD 15 26 8A 47 03	\$°°” è°° A!\$&SG♥
00000003E0:	3C 00 74 D0 EB D4 E8 17	FF B4 30 CD 21 50 BA 00	< tDèOèiÿ`OI!P°
00000003F0:	00 E8 0C FF BE 0F 00 46	E8 40 FF 58 8A C4 83 C6	èèÿ%* Fè@ÿXSÄfÆ
0000000400:	03 E8 37 FF BA 0F 00 E8	F6 FE BA 15 00 E8 F0 FE	♥è7ÿ* èøb°\$ èðb
0000000410:	BE 24 00 83 C6 05 8A C7	E8 20 FF BA 24 00 E8 DF	%\$ fÆ+SCè ÿ°\$ èß
0000000420:	FE BA 2B 00 E8 D9 FE BF	3D 00 83 C7 0A 8B C1 E8	b°+ èUp¿= fC◻<Aè
0000000430:	F1 FE 8A C3 E8 DB FE 83	EF 02 89 05 BA 3D 00 E8	ñbSAè0bfï°%†°= è
0000000440:	BE FE 32 C0 B4 4C CD 21		¾b2A`LI!

Рисунок 5 – «Хороший» .EXE модуль

0000000180:	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	
00000001A0:	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	
00000001C0:	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	
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 18 01 0D 0A 4F 53 20	76 65 72 73 69 6F 6E 3A	é!e.NOS version:
0000000310:	20 24 20 20 2E 20 20 24	0D 0A 4E 75 6D 62 65 72	\$. \$NNumber
0000000320:	20 4F 45 4D 3A 20 24 20	20 20 20 20 20 24 0D 0A	OEM: \$ \$N
0000000330:	53 65 72 69 61 6C 20 6E	75 6D 62 65 72 3A 20 24	Serial number: \$
0000000340:	20 20 20 20 20 20 20 20	20 20 20 20 24 0D 0A 50	\$NCP
0000000350:	43 20 56 65 72 73 69 6F	6E 3A 20 24 50 43 24 50	C Version: \$PCSP
0000000360:	43 2F 58 54 24 41 54 24	50 53 32 20 6D 6F 64 65	C/XT\$AT\$PS2 mode
0000000370:	6C 20 33 30 24 50 53 32	20 6D 6F 64 65 6C 20 38	l 30\$PS2 model 8
0000000380:	30 24 50 43 6A 72 24 50	53 32 20 6D 6F 64 65 6C	0\$PCjr\$PS2 model
0000000390:	20 35 30 2F 36 30 24 50	43 20 43 6F 6E 76 65 72	50/60\$PC Conver
00000003A0:	74 69 62 6C 65 24 4E 6F	74 20 66 6F 75 6E 64 20	tible\$Not found
00000003B0:	20 20 20 20 20 20 20 20	20 24 50 B4 09 CD 21 58	\$P'oi!X
00000003C0:	C3 24 0F 3C 09 76 02 04	07 04 30 C3 51 8A E0 E8	A\$<ove♦♦0AQ\$Aè
00000003D0:	EF FF 86 C4 B1 04 D2 E8	E8 E6 FF 59 C3 53 8A FC	iy†A±♦0èèæÿYASSü
00000003E0:	E8 E9 FF 88 25 4F 88 05	4F 8A C7 E8 DE FF 88 25	èèÿ`%0`†OSÇèbÿ`%
00000003F0:	4F 88 05 5B C3 51 52 32	E4 33 D2 B9 0A 00 F7 F1	o`+ [AQR2ä30' ÷ñ
0000000400:	80 CA 30 88 14 4E 33 D2	3D 0A 00 73 F1 3C 00 74	€E0`¶N30= sñ< t
0000000410:	04 0C 30 88 04 5A 59 C3	52 5A C3 B8 00 F0 8E C0	♦q0`♦ZYARZA` òZA
0000000420:	26 A0 FE FF BA 4D 01 E8	90 FF 3C FF 74 2B 3C FE	& py°Mèè¶ÿ<ÿt+<b
0000000430:	74 2D 3C FB 74 29 3C FC	74 4F 3C FA 74 33 3C F8	t-<üt)<üt0<üt3<ø
0000000440:	74 35 3C FD 74 37 3C F9	74 39 BE A6 01 83 C6 10	t5<ÿt7<üt9% @fA»
0000000450:	E8 79 FF BA A6 01 EB 3F	90 BA 5C 01 EB 39 90 BA	èÿÿ° èè?¶°\èè9¶°
0000000460:	5F 01 EB 33 90 BA 65 01	EB 2D 90 BA 87 01 EB 27	_èè3¶°èèè-¶°‡èè'
0000000470:	90 BA 68 01 EB 21 90 BA	75 01 EB 1B 90 BA 82 01	¶°heè!¶°uèè-¶°°,è
0000000480:	EB 15 90 BA 97 01 EB 0F	90 B4 C0 CD 15 26 8A 47	è\$¶°-èè¶°¶°AIS&SG
0000000490:	03 3C 00 74 D0 EB D4 E8	20 FF B4 30 CD 21 50 BA	♥< tBè0è ÿ`OI!P°
00000004A0:	03 01 E8 15 FF BE 12 01	46 E8 49 FF 58 8A C4 83	♥èèÿ% @FeIÿXSf
00000004B0:	C6 03 E8 40 FF BA 12 01	E8 FF FE BA 18 01 E8 F9	A♥è@ÿ° èèÿb° èèü
00000004C0:	FE BE 27 01 83 C6 05 8A	C7 E8 29 FF BA 27 01 E8	b%°@fA†SÇè)ÿ°'èè
00000004D0:	E8 FE BA 2E 01 E8 E2 FE	BF 40 01 83 C7 0A 8B C1	èb°..èèâb;@@fÇ¶°<A
00000004E0:	E8 FA FE 8A C3 E8 E4 FE	83 EF 02 89 05 BA 40 01	èüpSAèâpfio%†°@è
00000004F0:	E8 C7 FE 32 C0 B4 4C CD	21	èçb2A`LI!

Рисунок 6 – «Плохой» .EXE модуль

Контрольные вопросы

Ниже приведены ответы на контрольные вопросы:

1. Отличия исходных текстов .COM и .EXE программ
 - 1.1. .COM программа содержит один сегмент
 - 1.2. .EXE содержит произвольное число сегментов (>=1)

- 1.3. В **.COM** программе должна быть директива **ORG 100h** для установки **IP** в **100h**. Так же нужна директива **Assume** для инициализации регистров.
- 1.4. Нет, не все. Нельзя использовать 64-битные команды и команды с указанием сегмента, так как у **COM** файлов отсутствует таблица настроек.
2. Отличие форматов файлов **.COM** и **.EXE** модулей
 - 2.1. **.COM** файл содержит только машинный код и данные программы. Код располагается с адреса **100h**
 - 2.2. В «плохом» **.EXE** модуле в одном сегменте содержатся данные и машинный код, код начинается с адреса **300h** с адреса **0h** располагается **PSP**
 - 2.3. «Хороший» **.EXE** файл состоит из нескольких сегментов, в которых по отдельности хранятся данные, стек и машинный код. В «плохом» **.EXE** файле нет стека
3. Загрузка **.COM** модуля в основную память
 - 3.1. Код располагается с адреса **100h**
 - 3.2. С адреса **0h** располагается **PSP**
 - 3.3. Сегментные регистры указывают на начало **PSP**
 - 3.4. Стек занимает всю доступную память.
4. Загрузка «хорошего» **.EXE** модуля в основную память
 - 4.1. В регистры **ES** и **DS** записывается адрес начала **PSP**. Регистр **CS** на сегмент кода, регистр **SP** - на начало сегмента стека.
 - 4.2. Регистры **ES** и **DS** указывают на начало **PSP**
 - 4.3. Стек определяется при помощи директивы **Assume**
 - 4.4. Точка входа определяется при помощи директивы **End**

Выводы.

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

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ ДЛЯ .COM ФАЙЛА

```

TESTPC      SEGMENT
              ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
              ORG 100H

START:       JMP      BEGIN

OS_VER DB 13,10,"OS VERSION: $"
OS_FORM DB "  .  $"
NUMBER DB 13, 10, "NUMBER OEM: $"
NUMBER_FORM DB "      $"
SERIAL_NUMBER DB 13, 10, "SERIAL NUMBER: $"
SERIAL_NUMBER_FORM DB "                $"
PC_VER DB 13, 10, "PC VERSION: $"
VER_PC DB "PC$"
VER_PCXT DB "PC/XT$"
VER_AT DB "AT$"
VER_PS2_30 DB "PS2 MODEL 30$"
VER_PS2_80 DB "PS2 MODEL 80$"
VER_PJR DB "PCJR$"
VER_PS2_5060 DB "PS2 MODEL 50/60$"
VER_CON DB "PC CONVERTIBLE$"
ERROR_MESSAGE DB "NOT FOUND          $"

;-----
PRINT_MESSEGE PROC NEAR
    PUSH AX
    MOV AH, 09H
    INT 21H
    POP AX
    RET
PRINT_MESSEGE 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
                POP     CX
                RET

```

```

BYTE_TO_HEX    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

```

```

;-----
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_PCL
        OR      AL,30H
        MOV     [SI],AL

END_PCL:    POP     DX
            POP     CX
            RET

BYTE_TO_DEC    ENDP
;-----

OS_VERSION PROC NEAR
    PUSH DX

    POP DX
    RET
OS_VERSION ENDP
;-----

BEGIN:
;-----PC VER-----
    MOV AX,0F000H
    MOV ES,AX
    MOV AL,ES:[0FFFEH]
    MOV DX, OFFSET PC_VER
    CALL PRINT_MESSEGE
    CMP AL, 0FFH ; PC
    JE PC
    CMP AL, 0FEH ; PC/XT
    JE XT
    CMP AL, 0FBH ; PC/XT
    JE XT
    CMP AL, 0FCH
    JE AT_PS2_5060
    CMP AL, 0FAH ; PC2 30
    JE PS2_30
    CMP AL, 0F8H ; PC2 80
    JE PS2_80
    CMP AL, 0FDH ; PCJR
    JE JR

```

```
CMP AL, 0F9H ; PC CONVERTIBLE
JE CON
```

ERROR:

```
MOV SI, OFFSET ERROR_MESSAGE
ADD SI, 16
CALL BYTE_TO_HEX
MOV DX, OFFSET ERROR_MESSAGE
JMP END_PC
```

PC:

```
MOV DX, OFFSET VER_PC
JMP END_PC
```

XT:

```
MOV DX, OFFSET VER_PCXT
JMP END_PC
```

PC_AT:

```
MOV DX, OFFSET VER_AT
JMP END_PC
```

PS2_5060:

```
MOV DX, OFFSET VER_PS2_5060
JMP END_PC
```

PS2_30:

```
MOV DX, OFFSET VER_PS2_30
JMP END_PC
```

PS2_80:

```
MOV DX, OFFSET VER_PS2_80
JMP END_PC
```

JR:

```
MOV DX, OFFSET VER_PJR
JMP END_PC
```

CON:

```
MOV DX, OFFSET VER_CON
JMP END_PC
```

AT_PS2_5060:

```
MOV AH, 192
INT 15H
MOV AL, ES:[BX+3]
CMP AL, 00H
JE PC_AT
JMP PS2_5060
```

END_PC:

```

CALL PRINT_MESSEGE

;-----OS-----
MOV AH, 30H
INT 21H
PUSH AX
MOV DX, OFFSET OS_VER
CALL PRINT_MESSEGE
MOV SI, OFFSET OS_FORM
INC SI
CALL BYTE_TO_DEC
POP AX
MOV AL, AH
ADD SI, 3
CALL BYTE_TO_DEC
MOV DX, OFFSET OS_FORM
CALL PRINT_MESSEGE
MOV DX, OFFSET NUMBER
CALL PRINT_MESSEGE
MOV SI, OFFSET NUMBER_FORM
ADD SI, 5
MOV AL, BH
CALL BYTE_TO_DEC
MOV DX, OFFSET NUMBER_FORM
CALL PRINT_MESSEGE
MOV DX, OFFSET SERIAL_NUMBER
CALL PRINT_MESSEGE
MOV DI, OFFSET SERIAL_NUMBER_FORM
ADD DI, 10
MOV AX, CX
CALL WRD_TO_HEX
MOV AL, BL
CALL BYTE_TO_HEX
SUB DI, 2
MOV [DI], AX
MOV DX, OFFSET SERIAL_NUMBER_FORM
CALL PRINT_MESSEGE

;-----
XOR AL,AL
MOV AH,4CH
INT 21H

```

```

TESTPC    ENDS
END        START

```

ПРИЛОЖЕНИЕ Б

КОД ПРОГРАММЫ ДЛЯ .EXE ФАЙЛА

```

ASTACK    SEGMENT    STACK
            DW 20H DUP(?)
            ;DW 100H
ASTACK    ENDS

DATA      SEGMENT
    OS_VER DB 13,10,"OS VERSION: $"
    OS_FORM DB "    .  $"
    NUMBER DB 13, 10, "NUMBER OEM: $"
    NUMBER_FORM DB "        $"
    SERIAL_NUMBER DB 13, 10, "SERIAL NUMBER: $"
    SERIAL_NUMBER_FORM DB "                $"
    PC_VER DB 13, 10, "PC VERSION: $"
    VER_PC DB "PC$"
    VER_PCXT DB "PC/XT$"
    VER_AT DB "AT$"
    VER_PS2_30 DB "PS2 MODEL 30$"
    VER_PS2_80 DB "PS2 MODEL 80$"
    VER_PJR DB "PCJR$"
    VER_PS2_5060 DB "PS2 MODEL 50/60$"
    VER_CON DB "PC CONVERTIBLE$"
    ERROR_MESSAGE DB "NOT FOUND        $"
DATA      ENDS

CODE      SEGMENT
    ASSUME CS:CODE,DS:DATA,SS:ASTACK
;-----
PRINT_MESSEGE PROC NEAR
    PUSH AX
    MOV AH, 09H
    INT 21H
    POP AX
    RET

```

```

PRINT_MESSEGE 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
                POP      CX
                RET
BYTE_TO_HEX    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
;-----

```

```

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_PCL
                OR      AL,30H
                MOV     [SI],AL

```

```

END_PCL:        POP     DX
                POP     CX
                RET

```

```

BYTE_TO_DEC    ENDP

```

```

;-----

```

```

OS_VERSION PROC NEAR

```

```

    PUSH DX

```

```

    POP DX

```

```

    RET

```

```

OS_VERSION ENDP

```

```

;-----

```

```

MAIN PROC FAR

```

```

    PUSH AX

```

```

    SUB AX,AX

```

```

    MOV AX,DATA

```

```

    MOV DS,AX

```

```

    POP AX

```

```

;-----PC VER-----

```

```

    MOV AX,0F000H

```



```

MOV ES,AX
MOV AL,ES:[0FFFEH]
    MOV DX, OFFSET PC_VER
    CALL PRINT_MESSEGE
    CMP AL, 0FFH ; PC
    JE PC
    CMP AL, 0FEH ; PC/XT
    JE XT
    CMP AL, 0FBH ; PC/XT
    JE XT
    CMP AL, 0FCH
    JE AT_PS2_5060
    CMP AL, 0FAH ; PC2 30
    JE PS2_30
    CMP AL, 0F8H ; PC2 80
    JE PS2_80
    CMP AL, 0FDH ; PCJR
    JE JR
    CMP AL, 0F9H ; PC CONVERTIBLE
    JE CON

```

ERROR:

```

    MOV SI, OFFSET ERROR_MESSAGE
    ADD SI, 16
    CALL BYTE_TO_HEX
    MOV DX, OFFSET ERROR_MESSAGE
    JMP END_PC

```

PC:

```

    MOV DX, OFFSET VER_PC
    JMP END_PC

```

XT:

```

    MOV DX, OFFSET VER_PCXT
    JMP END_PC

```

PC_AT:

```

    MOV DX, OFFSET VER_AT
    JMP END_PC

```

PS2_5060:

```

    MOV DX, OFFSET VER_PS2_5060
    JMP END_PC

```

PS2_30:

```

    MOV DX, OFFSET VER_PS2_30

```

```

        JMP END_PC
PS2_80:
        MOV DX, OFFSET VER_PS2_80
        JMP END_PC
JR:
        MOV DX, OFFSET VER_PJR
        JMP END_PC
CON:
        MOV DX, OFFSET VER_CON
        JMP END_PC
AT_PS2_5060:
        MOV AH, 192
        INT 15H
        MOV AL, ES:[BX+3]
        CMP AL, 00H
        JE PC_AT
        JMP PS2_5060
END_PC:
        CALL PRINT_MESSEGE

;-----OS-----
        MOV AH, 30H
        INT 21H
        PUSH AX
        MOV DX, OFFSET OS_VER
        CALL PRINT_MESSEGE
        MOV SI, OFFSET OS_FORM
        INC SI
        CALL BYTE_TO_DEC
        POP AX
        MOV AL, AH
        ADD SI, 3
        CALL BYTE_TO_DEC
        MOV DX, OFFSET OS_FORM
        CALL PRINT_MESSEGE
        MOV DX, OFFSET NUMBER
        CALL PRINT_MESSEGE
        MOV SI, OFFSET NUMBER_FORM
        ADD SI, 5
        MOV AL, BH
        CALL BYTE_TO_DEC

```

```

        MOV DX, OFFSET NUMBER_FORM
        CALL PRINT_MESSEGE
        MOV DX, OFFSET SERIAL_NUMBER
        CALL PRINT_MESSEGE
        MOV DI, OFFSET SERIAL_NUMBER_FORM
        ADD DI, 10
        MOV AX, CX
        CALL WRD_TO_HEX
        MOV AL, BL
        CALL BYTE_TO_HEX
        SUB DI, 2
        MOV [DI], AX
        MOV DX, OFFSET SERIAL_NUMBER_FORM
        CALL PRINT_MESSEGE
;-----
        XOR  AL,AL
        MOV  AH,4CH
        INT  21H

MAIN ENDP
CODE ENDS
END MAIN

```