

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. 7383

Преподаватель

МЕДВЕДЕВ И.С.

ЕФРЕМОВ М.А.

Санкт-Петербург

2019

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

В ходе работы были написаны некоторые функции и структуры данных, которые описаны в табл. 1-2.

Таблица 1 – Используемые функции

Название функции	Выполняемая задача
TETR_TO_HEX	Перевод числа из 2-ой в 16-ричную систему счисления.
BYTE_TO_HEX	Осуществляет перевод байта, помещенного в AL, в два символа в шестнадцатеричной системе счисления, помещая результат в AX.
WRD_TO_HEX	Переводит числа размером 2 байта в 16-ричную систему счисления. В AX – число, в DI – адрес последнего символа.
BYTE_TO_DEC	Переводит в 10-ную систему счисления, SI – адрес поля младшей цифры.
Output	Вывод сообщения на экран.
PC_VERSION	Вывод строки, содержащей версию ПК.
VERSION_MS_DOS	Определение версии ОС

SERIAL_NUMBER	Определение серийного номера пользователя
NUMBER_OEM	Определение номера OEM

Таблица 2 – Структуры данных

Название	Тип	Назначение
VersPC	db	Строка «Version PC»
VersMSDOS		Строка «Version MS-DOS: . »
NumOEM		Строка «Number OEM: »
SerNum		Строка «Serial Number: »
PC		Строка «PC»
PCXT		Строка «PC/XT»
AT		Строка «AT»
PS2_30		Строка «PS2 model 30»
PS2_80		Строка «PS2 model 80»
PC2_50_60		Строка «PS2 model 50/60»
PCjr		Строка «PCjr»
PC_Convert		Строка «PC Convertible»

Также в ходе работы были созданы .COM файл, «хороший» и «плохой» .EXE файлы. Результаты работы представлены на рис. 1-3.

```
C:\>LAB1_C.COM
Version PC: AT
Version MS-DOS: 5.0
Number OEM:240
Serial Number:000000
```

Рисунок 1 – Запуск .COM файла.

```

C:\>LAB1_C.EXE

Version PC:

240          000000

Version PC:          5 0

Version PC:          240

Version PC:          000000

Version PC:

```

Рисунок 2 – Запуск «плохого» .EXE файла.

```

C:\>LAB1_E.EXE
Version PC: AT
Version MS-DOS: 5.0
Number OEM:240
Serial Number:000000

```

Рисунок 3 – Запуск «хорошего» .EXE файла.

Также были рассмотрены структуры файлов. Структуры представлены на рис. 4 – 6. В конце работы .COM и .EXE файлы был запущен с помощью отладчика, как показано на рис. 7 – 8.

```

D:\LAB\LAB1_C.COM
00000000: E9 AC 01 56 65 72 73 69 6F 6E 20 50 43 3A 20 24 e7@Version PC: $
00000001: 56 65 72 73 69 6F 6E 20 4D 53 2D 44 4F 53 3A 20 Version MS-DOS:
00000002: 20 2E 20 20 0D 0A 24 4E 75 6D 62 65 72 20 4F 45 . JCSNumber OE
00000003: 4D 3A 20 20 0D 0A 24 53 65 72 69 61 6C 20 4E M: JCSSerial N
00000004: 75 6D 62 65 72 3A 20 20 20 20 0D 0A 24 50 umber: JCS
00000005: 43 0D 0A 24 50 43 2F 58 54 0D 0A 24 41 54 0D 0A CJCSPC/XTJCSATJC
00000006: 24 50 53 32 20 6D 6F 64 65 6C 20 33 30 0D 0A 24 $PS2 model 30JCS
00000007: 50 53 32 20 6D 6F 64 65 6C 20 35 30 2F 36 30 0D PS2 model 50/60J
00000008: 0A 24 50 53 32 20 6D 6F 64 65 6C 20 38 30 0D 0A C$PS2 model 80JCS
00000009: 24 50 43 6A 72 0D 0A 24 50 43 20 43 6F 6E 76 65 $PCjrJCSPC Conve
0000000A: 72 74 69 62 6C 65 0D 0A 24 B4 09 CD 21 C3 24 0F rtibleJCS?OI!AS*
0000000B: 3C 09 76 02 04 07 04 30 C3 51 8A C4 E8 EF FF 86 <OvO+OQAQSAeiyt
0000000C: C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 E9 FF A+Oee?yYASSueey
0000000D: 88 25 4F 88 05 4F 8A C7 32 E4 E8 DC FF 88 25 4F ?%O?ASOC2aeUy?%O
0000000E: 88 05 5B C3 51 52 50 32 E4 33 D2 B9 0A 00 F7 F1 ?IAQRP2a3O?C ?n
0000000F: 80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3D 00 00 ?EO?4N3O=C sn=
00000010: 76 04 0C 30 88 04 58 5A 59 C3 BB 00 F0 8E C3 26 vOQ?XZYA> ?ZA&
00000011: A0 FE FF BA 03 01 E8 90 FF 3C FF 74 20 3C FE 74 ?y?Oe?y<yt <?t
00000012: 22 3C FB 74 1E 3C FC 74 23 3C FA 74 25 3C F8 74 "CutA<ut#<utx<ot
00000013: 27 3C FC 74 29 3C FD 74 2B 3C F9 74 2D BA 4F 01 'Cut><yt+<ut-?O@
00000014: EB 2E 90 BA 54 01 E8 60 FF EB 25 90 BA 5C 01 EB e.??T@e`ye%??\@e
00000015: 1F 90 BA 61 01 EB 19 90 BA 70 01 EB 13 90 BA 82 V??a@e!??p@e!??'
00000016: 01 EB 0D 90 BA 91 01 EB 07 90 BA 98 01 EB 01 90 @eJ??'@e=???@e@?
00000017: E8 36 FF BA 30 CD 21 C3 BE 10 01 83 C6 10 E8 63 e6y?OI!A?>???ec
00000018: FF 83 C6 03 8A C4 E8 5B FF C3 8A C7 BE 27 01 83 y??S@e!yASC?'@?
00000019: C6 0D E8 4F FF C3 8A C3 E8 1E FF BF 38 01 83 C7 ?FeOyAS@e!y?8@?C
0000001A: 0E 89 05 8B C1 BF 38 01 83 C7 13 E8 1C FF C3 E8 J?A<A?8@?C!e-y@e
0000001B: 58 FF E8 C3 FF E8 D2 FF E8 DB FF BA 10 01 E8 E8 xyeAyeOyeUy?>@ee
0000001C: FE BA 27 01 E8 E2 FE BA 38 01 E8 DC FE 32 C0 B4 ???@ea??8@eU?2A?
0000001D: 4C CD 21 LI!

```

Рисунок 4 – Структура .COM файла.

JENLAPNARI C.EHE															
0000000000:	4D	5A	D9	00	03	00	00	00	00	00	00	00	00	00	00
0000000010:	00	00	00	00	00	00	01	00	00	00	00	00	00	00	00
0000000020:	6A	72	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000030:	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
0000000050:	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
0000000070:	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
0000000090:	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
00000000B0:	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
00000000D0:	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
00000000F0:	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
0000000110:	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
0000000130:	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
0000000150:	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
0000000170:	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
0000000190:	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
00000001B0:	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
00000001D0:	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
00000001F0:	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
0000000210:	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
0000000230:	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
0000000250:	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
0000000270:	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
0000000290:	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
00000002B0:	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
00000002D0:	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
00000002F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000300:	E9	B2	01	56	65	72	73	69	6F	6E	20	50	43	3A	20
0000000310:	56	65	72	73	69	6F	6E	20	4D	53	2D	44	4F	53	3A
0000000320:	20	2E	20	20	00	0A	24	4E	75	6D	62	65	72	20	4F
0000000330:	4D	3A	20	20	00	00	0A	24	53	65	72	69	61	6C	20
0000000340:	75	6D	62	65	72	2A	20	20	20	20	20	20	00	0A	24
0000000350:	43	0D	0A	24	50	43	2F	58	54	0D	0A	24	41	54	0D
0000000360:	24	50	53	32	20	6D	6F	64	65	6C	20	33	30	0D	0A
0000000370:	50	53	32	20	6D	6F	64	65	6C	20	35	30	2F	36	30
0000000380:	0A	24	50	53	32	20	6D	6F	64	65	6C	20	38	30	0D
0000000390:	24	50	43	6A	72	00	0A	24	50	43	20	43	6F	6E	76
00000003A0:	72	74	69	62	6C	65	0D	0A	24	B4	09	CD	21	C3	24
00000003B0:	3C	09	76	02	04	07	04	30	C3	51	8A	C4	E8	EF	FF
00000003C0:	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53	8A	FC	E8	E9
00000003D0:	88	25	4F	88	05	4F	8A	C7	32	E4	E8	DC	FF	88	25
00000003E0:	88	05	5B	C3	51	52	50	32	E4	33	D2	B9	0A	00	F7
00000003F0:	80	CA	30	88	14	4E	33	D2	3D	0A	00	73	F1	3D	00
0000000400:	76	04	0C	30	88	04	58	5A	59	C3	BB	00	F0	8E	C3
0000000410:	0A	FE	FF	BA	03	01	E8	90	FF	3C	FF	74	20	3C	FE
0000000420:	22	3C	FC	74	1E	3C	FC	74	23	3C	FA	74	25	3C	F8
0000000430:	27	3C	FC	74	29	3C	FD	74	2B	3C	F9	74	2D	BA	4F
0000000440:	EB	2E	90	BA	54	01	E8	60	FF	EB	25	90	BA	5C	01
0000000450:	1F	90	BA	61	01	E3	9	90	BF	70	01	EB	13	90	82
0000000460:	01	EB	0D	90	BA	91	01	EB	07	90	BA	98	01	EB	01
0000000470:	E8	36	FF	BA	30	CD	21	C3	BE	10	01	83	C6	10	E8
0000000480:	FF	83	C6	03	8A	C4	E8	5B	FF	C3	50	53	56	8A	C7
0000000490:	27	01	83	C6	0D	E8	4C	FF	5E	5B	58	C3	8A	C3	E8
00000004A0:	FF	BF	38	01	83	C7	0E	89	05	8B	C1	BF	38	01	83
00000004B0:	13	E8	16	FC	C3	E8	12	FF	E8	0D	FF	E8	C3	E8	12
00000004C0:	FF	BA	10	01	E8	E2	FE	BA	27	01	E8	DC	FE	BA	38
00000004D0:	E8	D6	FE	32	C0	B4	4C	CD	21						

Рисунок 5 – Структура «плохого» .EXE файла.

00000005F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000600:	56	65	72	73	69	6F	6E	20	50	43	3A	20	24	56	65
0000000610:	73	69	6F	6E	20	4D	53	2D	44	4F	53	3A	20	20	2E
0000000620:	20	0D	0A	24	4E	75	6D	62	65	72	20	4F	45	4D	3A
0000000630:	20	20	0D	0A	24	53	65	72	69	61	6C	20	4E	75	6D
0000000640:	65	72	3A	20	20	20	20	20	20	20	0D	0A	24	50	43
0000000650:	0A	24	50	43	2F	58	54	0D	0A	24	41	54	0D	0A	24
0000000660:	53	32	20	6D	6F	64	65	6C	20	33	30	0D	0A	24	50
0000000670:	32	20	6D	6F	64	65	6C	20	35	30	2F	36	30	0D	0A
0000000680:	50	53	32	20	6D	6F	64	65	6C	20	38	30	0D	0A	24
0000000690:	43	6A	72	0D	0A	24	50	43	20	43	6F	6E	76	65	72
00000006A0:	69	62	6C	65	0D	0A	24	00	00	00	00	00	00	00	00
00000006B0:	B4	09	CD	21	CB	24	0F	3C	09	76	02	04	07	04	30
00000006C0:	51	8A	C4	9A	05	00	4B	00	86	C4	B1	04	D2	E8	9A
00000006D0:	00	4B	00	59	CB	53	8A	FC	9A	10	00	4B	00	88	25
00000006E0:	88	05	4F	8A	C7	32	E4	9A	10	00	4B	00	88	25	4F
00000006F0:	05	5B	CB	51	52	50	32	E4	33	D2	B9	0A	00	F7	F1
0000000700:	CA	30	88	14	4E	33	D2	3D	0A	00	73	F1	3D	00	00
0000000710:	04	0C	30	88	04	58	5A	59	CB	BB	00	F0	8E	C3	26
0000000720:	FE	FF	BA	00	00	9A	00	00	4B	00	3C	FF	74	20	3C
0000000730:	74	22	3C	FB	74	1E	3C	FC	74	25	3C	FA	74	27	3C
0000000740:	74	29	3C	FC	74	2B	3C	FD	74	2D	3C	F9	74	2F	BA
0000000750:	00	EB	30	90	BA	52	00	9A	00	00	4B	00	EB	25	90
0000000760:	5A	00	EB	1F	90	BA	5F	00	EB	19	90	BA	6E	00	EB
0000000770:	90	BA	90	00	EB	0D	90	BA	8F	00	EB	07	90	BA	96
0000000780:	EB	01	90	9A	00	00	4B	00	B4	30	CD	21	CB	8D	36
0000000790:	00	83	C6	10	9A	43	00	4B	00	83	C6	03	8A	C4	9A
00000007A0:	00	4B	00	C3	8A	C7	8D	36	24	00	83	C6	0D	9A	43
00000007B0:	4B	00	CB	8A	C3	9A	10	00	4B	00	8D	3E	35	00	83
00000007C0:	0E	89	05	8B	C1	8D	3E	35	00	83	C7	13	9A	25	00
00000007D0:	00	CB	B8	40	00	8E	D8	9A	69	00	4B	00	E8	AE	FF
00000007E0:	F4	00	00	00	9A	03	01	4B	00	8D	16	0D	00	9A	00
00000007F0:	4B	00	8D	16	24	00	9A	00	00	4B	00	8D	16	35	00
0000000800:	00	00	4B	00	B4	4C	CD	21	CB						

Version PC: \$Ver
sion MS-DOS:
JOSNumber OEM:
JOSSerial Numb
er: JOSPC.
OSPC/XTJOSATJOSF
S2 model 30JOSPS
2 model 50/60JOS
PS2 model 180JOS
CjrJOSPC Convert
ibleJOS
!o1\$*K0u0\$*00\$
QK-b\$ K \$-0\$mb\$
K Y-SHMB\$ K MZ0
MZ0K120b\$ K MZ0M
ALTRP203\$00 gen
\$00M13\$00 se= v
\$000\$XZ1\$00 E0\$A
I b K < t < I
t < < \$t < \$t < t < < o
t < < \$t < < t < < t < M
00PJR b K MZPJ
Z MZPJ b PJ0P m
PJA MZPJ0P MZPJ
00Pb K 10=0\$H6F
G1\$ b C G1\$ b-C
K H1H6\$ G1\$ b-C
K K1\$ b C H2\$ G1
L1\$H1\$H5 G1\$ b-C
00 b 0\$H1 K mo b
Y K b0\$K H= F K
K H-\$ b K H-\$ b
K-\$1=0=

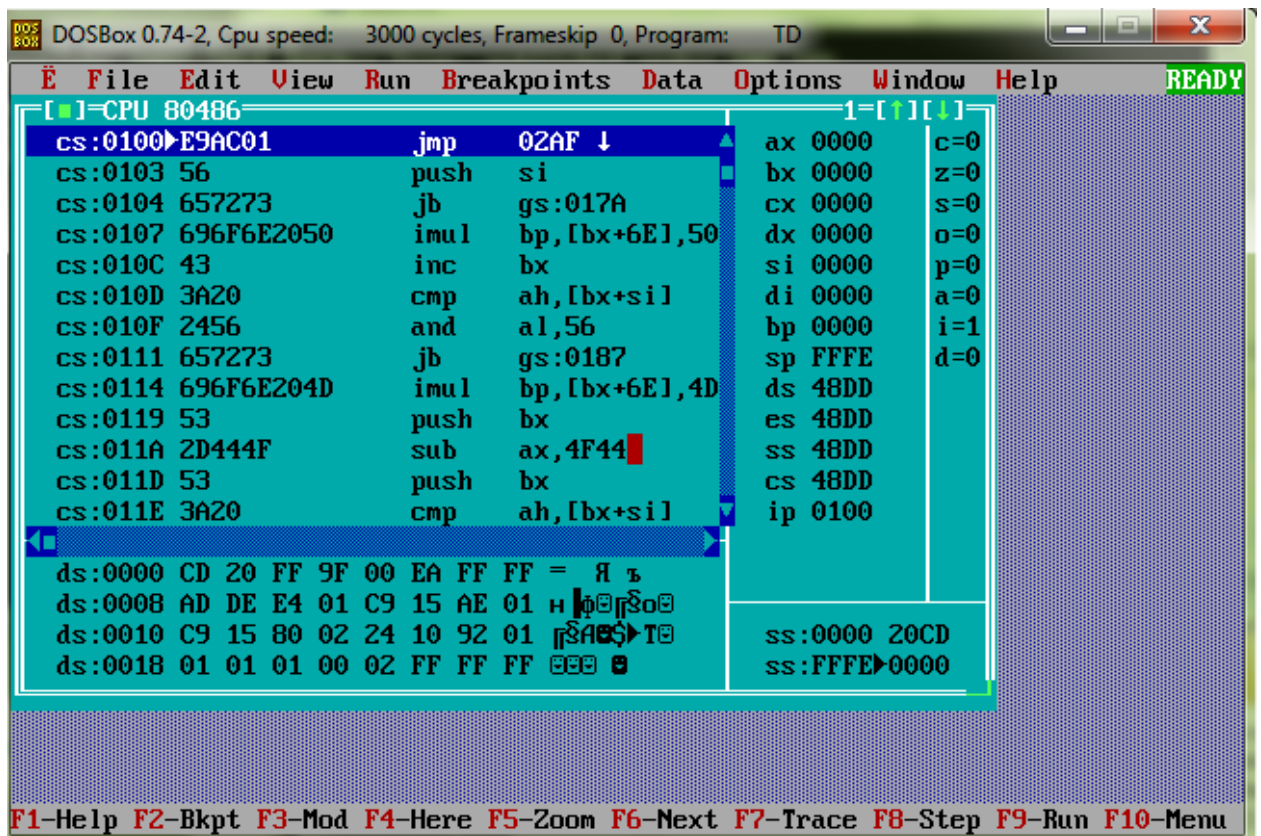


Рисунок 7 – Запуск .COM файла в дебагере.

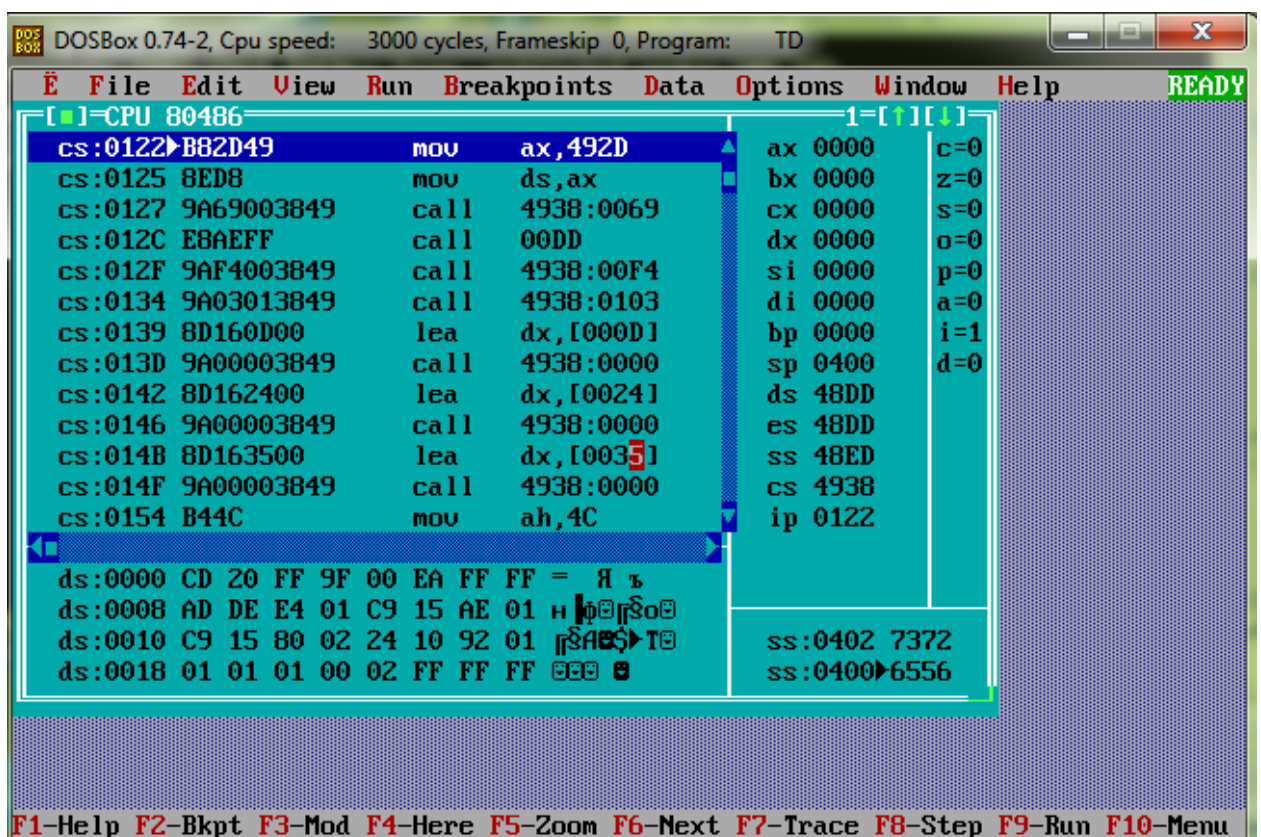


Рисунок 8 – Запуск .EXE файла в дебагере.

Просмотрев структуры и запустив структуры файлов, мы узнали, как располагаются те или иные сегменты.

Вывод

В ходе выполнения данной лабораторной работы были исследованы различия в структурах исходных текстов загрузочный модулей типов .COM и .EXE, структур файлов этих модулей и способах их загрузки в основную память.

Ответы на контрольные вопросы

Отличия исходных текстов COM и EXE программ

1) Сколько сегментов должна содержать COM-программа?

COM-программа содержит 1 сегмент.

2) Сколько сегментов должна содержать EXE-программа?

Три сегмента (сегмент стека, сегмент кода и сегмент данных).

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

Обязательная должна быть директива ORG 100h. Она нужна по той причине, что при загрузке модуля в ОП в начале COM-программы резервируется 256 байт для PSP.

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

Нет. Нельзя использовать команды, которые используют адреса сегментов, т.к. отсутствует таблица настроек.

Отличия форматов файлов COM и EXE модулей

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

COM-файл содержит единственный сегмент, в котором данные и машинные команды. Код начинается с адреса 100h.

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

В «плохом» EXE данные и код содержатся в одном сегменте. С адреса 0h идёт таблица настроек, которая состоит из длинных указателей

(смещение: сегмент). Она нужна для доступа к каждому сегменту кода. Код располагается с адреса 300h

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

В «хорошем» EXE данные, стек и код разделены по сегментам. Код программы начинается с адреса 600h.

Загрузка COM модуля в основную память

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

В ОП определяется адрес участка ОП, где достаточно места для загрузки модуля. Первые 100h байт занимает PSP, регистр SP устанавливается по адресу FFEh, в регистр IP записывается значение 100h, сегмент кода CS начинается с адреса 48DDh.

2) Что располагается с адреса 0?

PSP.

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

При загрузке программы они указывают на начало PSP (48DDh).

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

Регистр SP указывает на конец стека (FFFFh), SS – на начало (0h). Адреса расположены в диапазоне 0h – FFFEh

Загрузка «хорошего» EXE модуля в основную память

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

DS и ES устанавливаются на начало сегмента PSP (48DDh), SS – на начало сегмента стека (48Edh), CS – на начало сегмента команд (4938h).

2) На что указывают регистры DS и ES?

На начало PSP.

3) Как определяется стек?

Стек определяется с помощью директивы STACK.

4) Как определяется точка входа?

Точка входа определяется директивой END. Смещение точки входа загружается в регистр IP.

ПРИЛОЖЕНИЕ А. КОД ПРОГРАММЫ

lab1_c.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START:    JMP     BEGIN
;Данные
VersPC    db     'Version PC: $'
VersMSDOS db     'Version MS-DOS: . ',0DH,0AH,'$'
NumOEM    db     'Number OEM: ',0DH,0AH,'$'
SerNum    db     'Serial Number: ',0DH,0AH,'$'

PC         db     'PC',0DH,0AH,'$'
PCXT       db     'PC/XT',0DH,0AH,'$'
AT         db     'AT',0DH,0AH,'$'
PS2_30     db     'PS2 model 30',0DH,0AH,'$'
PS2_50_60  db     'PS2 model 50/60',0DH,0AH,'$'
PS2_80     db     'PS2 model 80',0DH,0AH,'$'
PCjr       db     'PCjr',0DH,0AH,'$'
PC_Convert db     'PC Convertible',0DH,0AH,'$'

Output     PROC near
    mov     ah,09h
    int     21h
    ret
Output     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
; байт в AL переводится в два символа шестн. числа в AX
    push    cx
    mov     al,ah
    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
        xor     ah,ah
        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
        push    ax
        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     ax,00h
        jbe     end_1
        or      al,30h
        mov     [si],al
end_1:          pop     ax
        pop     dx
        pop     cx
        ret

BYTE_TO_DEC     ENDP

PC_VERSION     PROC  near

        mov     bx,0F000h
        mov     es,bx
        mov     al,es:[0FFFEh]
        mov     dx,offset VersPC
        call    Output
        cmp     al,0FFh
        je      PC_lab

        cmp     al,0FEh
        je      PCXT_lab

        cmp     al,0FBh

```

```

je PCXT_lab

cmp al,0FCh
je AT_lab

cmp al,0FAh
je PS2_30_lab

cmp al,0F8h
je PS2_50_60_lab

cmp al,0FCh
je PS2_80_lab

cmp al,0FDh
je PCjr_lab

cmp al,0F9h
je PCConvert_lab

PC_lab:
    mov dx, offset PC
    jmp end_lab
PCXT_lab:
    mov dx, offset PCXT
    call Output
    jmp end_lab
AT_lab:
    mov dx, offset AT
    jmp end_lab
PS2_30_lab:
    mov dx, offset PS2_30
    jmp end_lab
PS2_50_60_lab:
    mov dx, offset PS2_50_60
    jmp end_lab

PS2_80_lab:
    mov dx, offset PS2_80
    jmp end_lab
PCjr_lab:
    mov dx, offset PCjr
    jmp end_lab
PCConvert_lab:
    mov dx, offset PC_Convert
    jmp end_lab

end_lab:
    call Output
    mov  ah,30h
    int  21h
    ret

```

PC_VERSION ENDP

VERSION_MS_DOS PROC near

```
    mov     si,offset VersMSDOS
    add     si,16
    call    BYTE_TO_DEC
    add     si,3
    mov     al,ah
    call    BYTE_TO_DEC
```

ret

VERSION_MS_DOS ENDP

NUMBER_OEM PROC near

```
    mov     al,bh
    lea     si,NumOEM
    add     si,13
    call    BYTE_TO_DEC
```

ret

NUMBER_OEM ENDP

SERIAL_NUMBER PROC near

```
    mov     al,bl
    call    BYTE_TO_HEX
    lea     di,SerNum
    add     di,14
    mov     [di],ax
    mov     ax,cx
    lea     di,SerNum
    add     di,19
    call    WRD_TO_HEX
```

ret

SERIAL_NUMBER ENDP

BEGIN:

```
    call    PC_VERSION
    call    VERSION_MS_DOS
    call    NUMBER_OEM
    call    SERIAL_NUMBER
```

```
    lea     dx,VersMSDOS
    call    Output
    lea     dx,NumOEM
    call    Output
    lea     dx,SerNum
    call    Output
```

```

        xor     al,al
        mov     ah,4ch
        int     21h

TESTPC   ENDS
        END     START

```

lab1_e.asm

```

EOL EQU '$'
AStack SEGMENT STACK
        DW 512 DUP(?)
AStack ENDS

DATA SEGMENT
VersPC      db      'Version PC: $'
VersMSDOS   db      'Version MS-DOS:  .  ',0DH,0AH,'$'
NumOEM      db      'Number OEM:      ',0DH,0AH,'$'
SerNum      db      'Serial Number:    ',0DH,0AH,'$'

PC          db      'PC',0DH,0AH,'$'
PCXT        db      'PC/XT',0DH,0AH,'$'
AT          db      'AT',0DH,0AH,'$'
PS2_30      db      'PS2 model 30',0DH,0AH,'$'
PS2_50_60   db      'PS2 model 50/60',0DH,0AH,'$'
PS2_80      db      'PS2 model 80',0DH,0AH,'$'
PCjr        db      'PCjr',0DH,0AH,'$'
PC_Convert  db      'PC Convertible',0DH,0AH,'$'
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE, DS:DATA, SS:AStack
Output   PROC FAR
        mov     ah,09h
        int     21h
        ret
Output   ENDP

TETR_TO_HEX  PROC FAR
        and     al,0fh
        cmp     al,09
        jbe     NEXT
        add     al,07
NEXT:      add     al,30h
        ret
TETR_TO_HEX ENDP

BYTE_TO_HEX  PROC FAR
        push    cx
        mov     al,ah
        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  FAR
        push    bx
        mov     bh,ah
        call    BYTE_TO_HEX
        mov     [di],ah
        dec     di
        mov     [di],al
        dec     di
        mov     al,bh
        xor     ah,ah
        call    BYTE_TO_HEX
        mov     [di],ah
        dec     di
        mov     [di],al
        pop     bx
        ret
WRD_TO_HEX      ENDP

BYTE_TO_DEC      PROC  FAR
        push    cx
        push    dx
        push    ax
        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     ax,00h
        jbe     end_1
        or      al,30h
        mov     [si],al
end_1:          pop     ax
        pop     dx
        pop     cx
        ret
BYTE_TO_DEC      ENDP

PC_VERSION      PROC  FAR

        mov     bx,0F000h
        mov     es,bx
        mov     al,es:[0FFFEh]
        mov     dx,offset VersPC
        call    Output

```

```

cmp al,0FFh
je PC_lab
cmp al,0FEh
je PCXT_lab
cmp al,0FBh
je PCXT_lab
cmp al,0FCh
je AT_lab
cmp al,0FAh
je PS2_30_lab
cmp al,0F8h
je PS2_80_lab
cmp al,0FDh
je PCjr_lab
cmp al,0F9h
je PCConvert_lab
PC_lab:
    mov dx, offset PC
    jmp end_lab
PCXT_lab:
    mov dx, offset PCXT
    jmp end_lab
AT_lab:
    mov dx, offset AT
    jmp end_lab
PS2_30_lab:
    mov dx, offset PS2_30
    jmp end_lab
PS2_80_lab:
    mov dx, offset PS2_80
    jmp end_lab
PCjr_lab:
    mov dx, offset PCjr
    jmp end_lab
PCConvert_lab:
    mov dx, offset PC_Convert
    jmp end_lab

end_lab:
    call Output
    mov ah,30h
    int 21h
    ret

```

PC_VERSION ENDP

VERSION_MS_DOS PROC NEAR

```

lea si,VersMSDOS
add si,16
call BYTE_TO_DEC
add si,3
mov al,ah

```



```

                call        BYTE_TO_DEC

                ret
VERSION_MS_DOS  ENDP

NUMBER_OEM      PROC FAR

                mov     al,bh
                lea     si,NumOEM
                add     si,13
                call    BYTE_TO_DEC

                ret
NUMBER_OEM      ENDP

SERIAL_NUMBER    PROC FAR
                mov     al,bl
                call    BYTE_TO_HEX
                lea     di,SerNum
                add     di,14
                mov     [di],ax
                mov     ax,cx
                lea     di,SerNum
                add     di,19
                call    WRD_TO_HEX

                ret
SERIAL_NUMBER    ENDP

Main            PROC FAR

                mov     ax, DATA
                mov     ds,ax
                call    PC_VERSION
                call    VERSION_MS_DOS
                call    NUMBER_OEM
                call    SERIAL_NUMBER

                lea     dx,VersMSDOS
                call    Output
                lea     dx,NumOEM
                call    Output
                lea     dx,SerNum
                call    Output

                mov     ah,4ch
                int     21h
                ret
Main            ENDP
CODE            ENDS
                END Main

```

