

МИНОБРНАУКИ РОССИИ
Санкт-Петербургский государственный
электротехнический университет
«ЛЭТИ» им. В.И. Ульянова (Ленина)
Кафедра математического обеспечения и применения ЭВМ

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

Студент гр. 8382

Терехов А.Е.

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.

1. С помощью данных в методических указаниях функций был написан исходник для загрузочного модуля с расширением .COM. Программа должна выводить сведения о типе IBM PC, версии MS-DOS и серийном номере OEM и пользователя.

2. После трансляции с использованием MASM и линковки была получена программа с расширением .EXE. Программа работает некорректно, как это можно заметить на рис.1.

```
C:\>LR1_COM.EXE

      щеТтип IBM PC:
      щеТтип IBM PC:          5 0          0000000
255
      щеТтип IBM PC:          0000000          255
      щеТтип IBM PC:          0000000
      щеТтип IBM PC:
```

Рис.1. Запуск "плохого" .EXE файла.

3. С помощью утилиты EXE2BIN.EXE был получен "хороший" .COM модуль, работающий корректно. Результат работы программы представлен на рис.2.

```
C:\>LR1_COM.COM
Тип IBM PC: AT or PS2 model 50-60
Версия MS-DOS: 5. 0
Серийный номер OEM: 255
Серийный номер пользователя: 0000000H
```

Рис.2. Запуск "хорошего" .COM файла.

4. Был написан исходный код для .EXE модуля, который должен выполнять те же действия, что и модуль на шаге 1.

5. После трансляции с использованием MASM и линковки была получена программа с расширением .EXE, работающая корректно, как это можно заметить на рис.3.

```
C:\>LR1_EXE.EXE
Тип IBM PC: AT or PS2 model 50-60
Версия MS-DOS: 5. 0
Серийный номер OEM: 255
Серийный номер пользователя: 000000H
```

Рис.3. Запуск "хорошего" .EXE файла.

6. С представлениями файлов в шестнадцатеричном виде можно ознакомиться на рисунках 4-6.

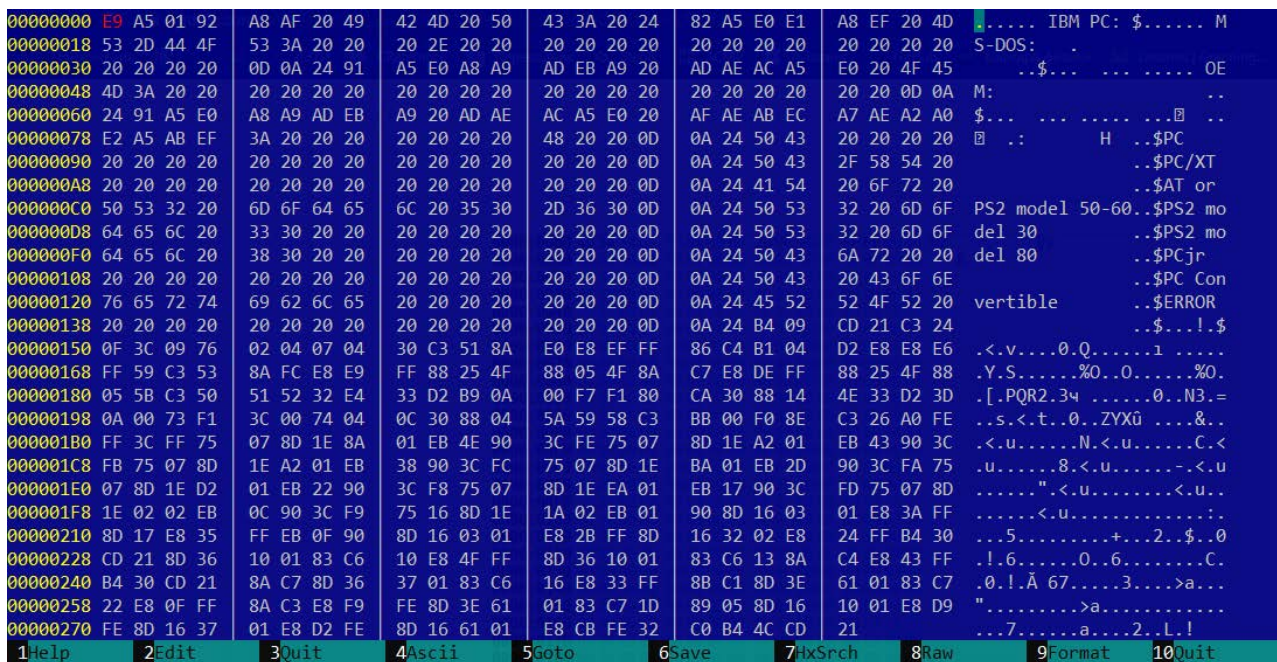


Рис.4. COM-файл в шестнадцатеричном виде.

00000000	4D 5A 85 01	03 00 00 00	20 00 00 00	FF FF 00 00	00 00 0E F4	00 01 00 00	MZ.....
00000018	1E 00 00 00	01 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000030	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000048	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000060	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000078	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000090	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000000A8	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000000C0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000000D8	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000000F0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000108	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000120	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000138	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000150	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000168	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000180	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000198	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000001B0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000001C8	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000001E0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000001F8	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000210	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000228	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000240	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000258	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000270	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000288	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000002A0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000002B8	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000002D0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000002E8	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000300	E9 A5 01 92	A8 AF 20 49	42 4D 20 50	43 3A 20 24	82 A5 E0 E1	A8 EF 20 4D IBM PC: \$.....
00000318	53 2D 44 4F	53 3A 20 20	20 2E 20 20	20 20 20 20	20 20 20 20	20 20 20 20	S-DOS: ..
00000330	20 20 20 20	0D 0A 24 91	A5 E0 A8 A9	AD EB A9 20	AD AE AC A5	E0 20 4F 45	..\$... .. 0
00000348	4D 3A 20 20	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	20 20 0D 0A	M: ..
00000360	24 91 A5 E0	A8 A9 AD EB	A9 20 AD AE	AC A5 E0 20	AF AE AB EC	A7 AE A2 A0	\$... ..
00000378	E2 A5 AB EF	3A 20 20 20	20 20 20 20	48 20 20 0D	0A 24 50 43	20 20 20 20	..\$PC
00000390	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 0D	0A 24 50 43	2F 58 54 20	..\$PC/XT
000003A8	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 0D	0A 24 41 54	20 6F 72 20	..\$AT or
000003C0	50 53 32 20	6D 6F 64 65	6C 20 35 30	2D 36 30 0D	0A 24 50 53	32 20 6D 6F	PS2 model 50-60..\$PS2 m
000003D8	64 65 6C 20	33 30 20 20	20 20 20 20	20 20 20 0D	0A 24 50 53	32 20 6D 6F	del 30 ..\$PS2 m
000003F0	64 65 6C 20	38 30 20 20	20 20 20 20	20 20 20 0D	0A 24 50 43	6A 72 20 20	del 80 ..\$PCjr
00000408	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 0D	0A 24 50 43	20 43 6F 6E	..\$PC Co
00000420	76 65 72 74	69 62 6C 65	20 20 20 20	20 20 20 0D	0A 24 45 52	52 4F 52 20	vertible ..\$ERROR
00000438	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 0D	0A 24 B4 09	CD 21 C3 24	..\$!..
00000450	0F 3C 09 76	02 04 07 04	30 C3 51 8A	E0 E8 EF FF	86 C4 B1 04	D2 E8 E8 E6	<.v...0.Q.....1 ...
00000468	FF 59 C3 53	8A FC E8 E9	FF 88 25 4F	88 05 4F 8A	C7 E8 DE FF	88 25 4F 88	.Y.S.....%0..0.....%0
00000480	05 5B C3 50	51 52 32 E4	33 D2 B9 0A	00 F7 F1 80	CA 30 88 14	4E 33 D2 3D	.[.PQR2.340..N3.
00000498	0A 00 73 F1	3C 00 74 04	0C 30 88 04	5A 59 58 C3	BB 00 F0 8E	C3 26 A0 FE	..s.<.t..0..ZYXü&.
000004B0	FF 3C FF 75	07 8D 1E 8A	01 EB 4E 90	3C FE 75 07	8D 1E A2 01	EB 43 90 3C	<.u.....N.<.u.....C.
000004C8	FB 75 07 8D	1E A2 01 EB	38 90 3C FC	75 07 8D 1E	BA 01 EB 2D	90 3C FA 75	u.....8.<.u.....<.u.
000004E0	07 8D 1E D2	01 EB 22 90	3C F8 75 07	8D 1E EA 01	EB 17 90 3C	FD 75 07 8D".<.u.....<.u.
000004F8	1E 02 02 EB	0C 90 3C F9	75 16 8D 1E	1A 02 EB 01	90 8D 16 03	01 E8 3A FF<.u.....
00000510	8D 17 E8 35	FF EB 0F 90	8D 16 03 01	E8 2B FF 8D	16 32 02 E8	24 FF B4 30	...5.....+.2..\$..
00000528	CD 21 8D 36	10 01 83 C6	10 E8 4F FF	8D 36 10 01	83 C6 13 8A	C4 E8 43 FF	!.6.....0..6.....C
00000540	B4 30 CD 21	8A C7 8D 36	37 01 83 C6	16 E8 33 FF	8B C1 8D 3E	61 01 83 C7	0.!.Ä 67.....3....>a..
00000558	22 E8 0F FF	8A C3 E8 F9	FE 8D 3E 61	01 83 C7 1D	89 05 8D 16	10 01 E8 D9	".....>a.....
00000570	FE 8D 16 37	01 E8 D2 FE	8D 16 61 01	E8 CB FE 32	C0 B4 4C CD	21	...7.....a.....2..L..

Рис.5. "Плохой" EXE-файл в шестнадцатеричном виде.

00000420	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20
00000430	20 0D 0A 24	91 A5 E0 A8	A9 AD EB A9	20 AD AE AC ..\$.
00000440	A5 E0 20 4F	45 4D 3A 20	20 20 20 20	20 20 20 20 .. OEM:
00000450	20 20 20 20	20 20 20 20	0D 0A 24 91	A5 E0 A8 A9 ..\$.
00000460	AD EB A9 20	AD AE AC A5	E0 20 AF AE	AB EC A7 AE
00000470	A2 A0 E2 A5	AB EF 3A 20	20 20 20 20	20 20 48 0D H.
00000480	0A 24 50 43	20 20 20 20	20 20 20 20	20 20 20 20 ..\$PC
00000490	20 20 20 20	20 20 20 0D	0A 24 50 43	2F 58 54 20 ..\$PC/XT
000004A0	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 0D
000004B0	0A 24 41 54	20 6F 72 20	50 53 32 20	6D 6F 64 65 ..\$AT or PS2 mode
000004C0	6C 20 35 30	2D 36 30 0D	0A 24 50 53	32 20 6D 6F l 50-60..\$PS2 mo
000004D0	64 65 6C 20	33 30 20 20	20 20 20 20	20 20 20 0D del 30
000004E0	0A 24 50 53	32 20 6D 6F	64 65 6C 20	38 30 20 20 ..\$PS2 model 80
000004F0	20 20 20 20	20 20 20 0D	0A 24 50 43	6A 72 20 20 ..\$PCjr
00000500	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 0D
00000510	0A 24 50 43	20 43 6F 6E	76 65 72 74	69 62 6C 65 ..\$PC Convertible
00000520	20 20 20 20	20 20 20 0D	0A 24 45 52	52 4F 52 20 ..\$ERROR
00000530	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 0D
00000540	0A 24 00 00	00 00 00 00	00 00 00 00	00 00 00 00 ..\$.
00000550	B4 09 CD 21	C3 24 0F 3C	09 76 02 04	07 04 30 C3 ..!. \$.<v...0
00000560	51 8A E0 E8	EF FF 86 C4	B1 04 D2 E8	E8 E6 FF 59 Q.....Y
00000570	C3 53 8A FC	E8 E9 FF 88	25 4F 88 05	4F 8A C7 E8 ..\$.
00000580	DE FF 88 25	4F 88 05 5B	C3 50 51 52	32 E4 33 D2 ..%. [.PQR2.34
00000590	B9 0A 00 F7	F1 80 CA 30	88 14 4E 33	D2 3D 0A 00 ..\$.
000005A0	73 F1 3C 00	74 04 0C 30	88 04 5A 59	58 C3 2B C0 s.<.t..0.ZYX.+
000005B0	50 B8 20 00	8E 0D BB 00	F0 8E C3 26	0A FE FF 3C P.
000005C0	FF 75 07 8D	1E 82 00 EB	4E 90 3C FE	75 07 8D 1E ..u...N.<.u...
000005D0	9A 00 EB 43	90 3C FB 75	07 8D 1E 9A	00 EB 38 90 ..C.<.u...8
000005E0	3C FC 75 07	8D 1E B2 00	EB 2D 90 3C	FA 75 07 8D <.u...<.u...
000005F0	1E CA 00 EB	22 90 3C F8	75 07 8D 1E	E2 00 07 8D .." "<.u...
00000600	90 3C FD 75	07 8D 1E FA	00 EB 0C 90	3C F9 75 16 <.u...<.u...
00000610	8D 1E 12 01	EB 01 90 8D	16 00 00 E8	32 FF 8D 17 ..\$.
00000620	EB 2D FF EB	0F 90 8D 16	00 00 E8 23	FF 8D 16 2A ..\$.
00000630	01 E8 1C FF	B4 30 CD 21	8D 36 0D 00	83 C6 10 E80.!.6.....
00000640	47 FF 8D 36	0D 00 83 C6	13 8A CA E8	2B FF B4 30 G..6.....;..0
00000650	CD 21 8A C7	8D 36 34 00	83 C6 16 E8	3F FF 8B C1 ..!.Ã 64.....+..
00000660	8D 3E 5B 00	83 C7 22 E8	07 FF 8A C3	E8 F1 FE 8D >[....."
00000670	3E 5B 00 83	C7 1D 89 05	8D 16 0D 00	E8 D1 FE 8D >[.....
00000680	16 34 00 E8	CA FE 8D 16	5B 00 E8 C3	FE 32 C0 B4 ..\$.
00000690	AC CD 21			4. !

Рис.6 (конец). "Хороший" EXE-файл в шестнадцатеричном виде.

7. Запуск в отладчике TD COM-программы представлен на рисунке 7.

The screenshot shows the CPU 80486 debugger window. The main pane displays assembly code with the following instructions and registers:

Address	Instruction	Register	Value
cs:0100	jmp	ax	0000
cs:0103	xchg	dx, ax	0000
cs:0104	test	al, AF	0000
cs:0106	and	[bx+di+42], cl	0000
cs:0109	dec	bp	0000
cs:010A	and	[bx+si+43], dl	0000
cs:010D	cmp	ah, [bx+si]	0000
cs:010F	and	al, 82	FFFE
cs:0111	movsw	ds	48DD
cs:0112	loopnz	es	48DD
cs:0114	test	al, EF	48DD
cs:0116	and	[di+53], cl	48DD
cs:0119	sub	ax, 4F44	0100

The registers on the right are:

Register	Value
ax	0000
bx	0000
cx	0000
dx	0000
si	0000
di	0000
bp	0000
sp	FFFE
ds	48DD
es	48DD
ss	48DD
cs	48DD
ip	0100

The bottom pane shows the memory dump:

Address	Hex Data	ASCII
ds:0000	CD 20 FF 9F 00 EA FF FF	= f Ω
ds:0008	AD DE E4 01 C9 15 AE 01	␣ ␣ ␣ ␣ ␣ ␣ ␣
ds:0010	C9 15 80 02 24 10 92 01	␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
ds:0018	01 01 01 00 02 FF FF FF	␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣

The status bar at the bottom shows the current address range: ss:0000 20CD ss:FFFE 0000.

Рис.7. Запуск СОМ-программы в отладчике.

8. Запуск в отладчике TD EXE-программы представлен на рисунке 8.

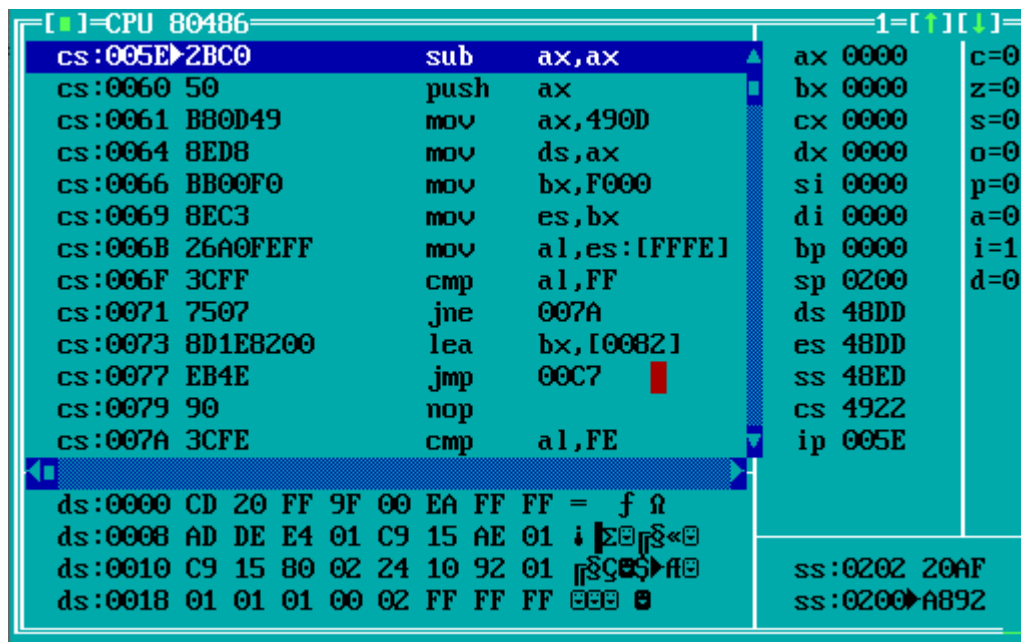


Рис.8. Запуск EXE-программы в отладчике.

Ответы на вопросы.

Отличия исходных текстов COM и EXE программ.

1) Сколько сегментов должна содержать COM-программа?

Один сегмент CODESEG.

2) EXE-программа?

Количество допустимых сегментов определяется используемой моделью памяти:

- small – один сегмент кода, один сегмент данных;
- compact – один сегмент кода, несколько сегментов данных;
- medium – несколько сегментов кода, один сегмент данных;
- large – несколько сегментов кода, несколько сегментов данных;
- huge – много сегментов кода, много сегментов данных.

А также сегмент стека.

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

Директива ASSUME, сообщающая ассемблеру информацию о

соответствии между сегментными регистрами и сегментами (сегментом в данном случае). Директива `ORG`, сообщающая компилятору о смещении адресации внутри кода.

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

В силу того, что в COM-программе сегментные регистры определяются в момент запуска, а не трансляции (компиляции) нельзя использовать команды, работающие с сегментами, команды, а также команды, размера больше 64Кб, или команды, работающие с 64-битными регистрами.

Отличия форматов файлов COM и EXE модулей.

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

В начале файла можно заметить данные, которые используются в программе, затем следует сам код. COM-программа содержит один сегмент, не превышающий 64Кб. Код располагается с начала программы, но при запуске ему всегда предшествует блок памяти для PSP длиной 100H байт.

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

"Плохой" EXE-файл содержит только один сегмент. В самом начале идет заголовок EXE-файла, в котором содержится полезная информация, такая как сигнатура, длина заголовка в 16-байтных параграфах, значение SP и IP при входе и т.д. Заголовок занимает адреса 00-1BH (28 байт). За заголовком следует таблица настроек адресов, которая занимает ровно столько места, сколько указано в заголовке. Сам код в данном EXE-файле начинается с адреса 300H.

3) Какова структура "хорошего" EXE-файла? Чем он отличается от "плохого" EXE-файла?

Также, как и в "плохом" в "хорошем" в самом начале идет заголовок и таблица настроек адресов. "Хороший" EXE-файл содержит три сегмента: стека (в шестнадцатеричном виде представлен множеством символов S), данных и кода. Сам код в "хорошем" EXE-файле начинается с адреса 400H, так как в отличие от "плохого" в нем определен стек.

Загрузка COM модуля в основную память.

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

Система выделяет свободный сегмент, в первые 256 байт этого сегмента записывается PSP, непосредственно за ним загружается содержимое COM-файла без изменений, то есть с адреса 0100H, указатель стека (регистр SP) устанавливается на конец сегмента.

2) Что располагается с адреса 0?

Program Segment Prefix (PSP) – область памяти размером 256 (0100h) байт, предшествующая программе при ее загрузке.

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

Все сегментные регистры имеют значения 48DDH. Они указывают на начало выделенной системой памяти, то есть на PSP.

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

Стек расположен в конце выделенной памяти, сразу после содержимого COM-файла. В стек записывается 0000H (адрес возврата для команды ret).

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, SS – на начало сегмента стека, CS – на начало сегмента команд. В IP загружается адрес первой команды.

2) На что указывают регистры DS и ES?

DS и ES указывают на начало префикса программного сегмента (PSP).

3) Как определяется стек?

Стек определяется в коде следующим образом:

<имя сегмента> SEGMENT STACK

DW <количество выделяемой памяти> DUP(<значение по

умолчанию>)

<имя сегмента> ENDS

ASSUME SS:AStack

4) Как определяется точка входа?

Точка входа – начальное значение IP – вычисляется с помощью директивы
END <метка>.

Вывод.

В ходе выполнения работы были написаны два варианта программы на языке ассемблера. Программа выводит на экран сообщения, содержащие информацию о некоторых свойствах системы. В процессе были получены три загрузочных модуля: COM из исходника для получения COM-файла, EXE из того же исходника, работающий некорректно, и EXE из исходника для EXE. Были выявлены различия в исходниках, структурах загрузочных модулей, порядке запуска этих двух вариантов.

ПРИЛОЖЕНИЕ А

Исходный код COM-программы.

```
CODESEG SEGMENT
    ASSUME cs:CODESEG, ds:CODESEG, es:NOTHING, ss:NOTHING
    ORG 100H
START: jmp BEGIN
; ===Данные===
TYPE_PC          db'Тип IBM PC: ','$'
MSDOS_VERSION     db'Версия MS-DOS:      ' ,0DH,0AH,'$'
OEM_NUMBER        db'Серийный номер OEM:    ' ,0DH,0AH,'$'
USER_NUMBER       db'Серийный номер пользователя: H' ,0DH,0AH,'$'
CASE_FF           db'PC                      ' ,0DH,0AH,'$'
CASE_FE_FB        db'PC/XT                  ' ,0DH,0AH,'$'
CASE_FC           db'AT or PS2 model 50-60   ' ,0DH,0AH,'$'
CASE_FA           db'PS2 model 30            ' ,0DH,0AH,'$'
CASE_F8           db'PS2 model 80            ' ,0DH,0AH,'$'
CASE_FD           db'PCjr                   ' ,0DH,0AH,'$'
CASE_F9           db'PC Convertible          ' ,0DH,0AH,'$'
ERROR_MSG         db'ERROR                  ' ,0DH,0AH,'$'
; Процедура печати строки
WriteMsg PROC near
    mov ah,09h
    int 21h
    ret
WriteMsg ENDP
;-----
TETR_TO_HEX PROC near
    and al,0Fh
    cmp al,09
    jbe NEXT
    add al,07
NEXT:  add al,30h; код нуля
    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 байта в 10с/с, si - адрес поля младшей цифры
; al содержит исходный байт
        push ax
        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_l
        or al,30h
        mov [si],al
end_l:  pop dx
        pop cx
        pop ax
        ret
BYTE_TO_DEC ENDP
;-----
BEGIN:
;получим тип IBM PC
        mov bx, 0F000h
        mov es, bx
        mov al, es:[0FFFEh]
        cmp al, 00FFh
        jne FE
        lea bx, CASE_FF
        jmp CONCAT
FE:
        cmp al, 00FEh
        jne FB
        lea bx, CASE_FE_FB
        jmp CONCAT
FB:
        cmp al, 00FBh
        jne FC
        lea bx, CASE_FE_FB
        jmp CONCAT

```

```

FC:
    cmp al, 00FCh
    jne FA
    lea bx, CASE_FC
    jmp CONCAT

FA:
    cmp al, 00FAh
    jne F8
    lea bx, CASE_FA
    jmp CONCAT

F8:
    cmp al, 00F8h
    jne FD
    lea bx, CASE_F8
    jmp CONCAT

FD:
    cmp al, 00FDh
    jne F9
    lea bx, CASE_FD
    jmp CONCAT

F9:
    cmp al, 00F9h
    jne ERROR
    lea bx, CASE_F9
    jmp CONCAT

CONCAT:
    lea dx, TYPE_PC
    call WriteMsg
    lea dx, [bx]
    call WriteMsg
    jmp DOSBOX_VERS

ERROR:
    lea dx, TYPE_PC
    call WriteMsg
    lea dx, ERROR_MSG
    call WriteMsg

; получим версию досбокса
DOSBOX_VERS:
    mov ah, 30h; функция вернет в al старший ah младший номер версии
    int 21h
    lea si, MSDOS_VERSION
    add si, 16
    call BYTE_TO_DEC
    lea si, MSDOS_VERSION
    add si, 19
    mov al, ah
    call BYTE_TO_DEC

; серийный номер OEM
    mov ah, 30h
    int 21h
    mov al, bh
    lea si, OEM_NUMBER
    add si, 22
    call BYTE_TO_DEC

; серийный номер пользователя
    mov ax, cx
    lea di, USER_NUMBER

```

```

add di, 34
call WRD_TO_HEX
mov al, bl
call BYTE_TO_HEX
lea di, USER_NUMBER
add di, 29
mov [di], ax

lea dx, MSDOS_VERSION
call WriteMsg
lea dx, OEM_NUMBER
call WriteMsg
lea dx, USER_NUMBER
call WriteMsg
xor al, al
mov ah, 4Ch
int 21h
CODESEG ENDS
END START ;конец модуля, START - точка входа

```


ПРИЛОЖЕНИЕ Б

Исходный код EXE-программы.

```
AStack SEGMENT STACK
    DW 100h DUP(5353H)
AStack ENDS
; ===Данные===
DATA SEGMENT
TYPE_PC      db'Тип IBM PC: ','$'
MSDOS_VERSION db'Версия MS-DOS: '
OEM_NUMBER   db'Серийный номер OEM: '
USER_NUMBER  db'Серийный номер пользователя: '
CASE_FF      db'PC'
CASE_FE_FB   db'PC/XT'
CASE_FC      db'AT or PS2 model 50-60'
CASE_FA      db'PS2 model 30'
CASE_F8      db'PS2 model 80'
CASE_FD      db'PCjr'
CASE_F9      db'PC Convertible'
ERROR_MSG    db'ERROR'
DATA ENDS

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

; Процедура печати строки
WriteMsg PROC near
    mov ah,09h
    int 21h
    ret
WriteMsg ENDP
;-----
TETR_TO_HEX PROC near
    and al,0Fh
    cmp al,09
    jbe NEXT
    add al,07
NEXT:    add al,30h; код нуля
    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 байта в 10с/с, si - адрес поля младшей цифры
; al содержит исходный байт
        push ax
        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_l
        or al,30h
        mov [si],al
end_l:   pop dx
        pop cx
        pop ax
        ret
BYTE_TO_DEC ENDP
;-----
Main PROC FAR
        sub ax, ax
        push ax
        mov ax, DATA
        mov ds, ax
;получим тип IBM PC
        mov bx, 0F000h
        mov es, bx
        mov al, es:[0FFFEh]
        cmp al, 00FFh
        jne FE
        lea bx, CASE_FF
        jmp CONCAT
FE:
        cmp al, 00FEh
        jne FB
        lea bx, CASE_FE_FB
        jmp CONCAT
FB:
        cmp al, 00FBh
        jne FC
        lea bx, CASE_FE_FB
        jmp CONCAT
FC:
        cmp al, 00FCh
        jne FA

```

```

        lea bx, CASE_FC
        jmp CONCAT
FA:
        cmp al, 00FAh
        jne F8
        lea bx, CASE_FA
        jmp CONCAT
F8:
        cmp al, 00F8h
        jne FD
        lea bx, CASE_F8
        jmp CONCAT
FD:
        cmp al, 00FDh
        jne F9
        lea bx, CASE_FD
        jmp CONCAT
F9:
        cmp al, 00F9h
        jne ERROR
        lea bx, CASE_F9
        jmp CONCAT
CONCAT:
        lea dx, TYPE_PC
        call WriteMsg
        lea dx, [bx]
        call WriteMsg
        jmp DOSBOX_VERS
ERROR:
        lea dx, TYPE_PC
        call WriteMsg
        lea dx, ERROR_MSG
        call WriteMsg
; получим версию досбокса
DOSBOX_VERS:
        mov ah, 30h; функция вернет в al старший ah младший номер версии
        int 21h
        lea si, MSDOS_VERSION
        add si, 16
        call BYTE_TO_DEC
        lea si, MSDOS_VERSION
        add si, 19
        mov al, ah
        call BYTE_TO_DEC
; серийный номер OEM
        mov ah, 30h
        int 21h
        mov al, bh
        lea si, OEM_NUMBER
        add si, 22
        call BYTE_TO_DEC
; серийный номер пользователя
        mov ax, cx
        lea di, USER_NUMBER
        add di, 34
        call WRD_TO_HEX
        mov al, bl
        call BYTE_TO_HEX
        lea di, USER_NUMBER
        add di, 29
        mov [di], ax

        lea dx, MSDOS_VERSION

```



```
        call WriteMsg
        lea dx, OEM_NUMBER
        call WriteMsg
        lea dx, USER_NUMBER
        call WriteMsg
        xor al,al
        mov ah,4Ch
        int 21h
Main ENDP
CODE ENDS
END Main
```