

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. 0382

Корсунов А.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Постановка задачи

Цель работы.

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

Сведения о функциях и структурах данных управляющей программы.

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа ASCII (результат записывается в AL)
BYTE_TO_HEX	Перевод байта из AL переводится в два символа шестн. числа в AX (в AL старшая цифра в AH младшая цифра)
WRD_TO_HEX	Перевод в 16 с/с 16-ти разрядного числа (в AX – число, DI – адрес последнего символа)
BYTE_TO_DEC	Перевод в 10 с/с (SI – адрес поля младшей цифры)
PRINT_PC_TYPE	Вывод типа ПК на экран
PRINT_SYSTEM_VERSION	Вывод версии DOS на экран

Шаг 1. Написать текст исходного .COM модуля, который определяет тип PC и версию системы. Построить «плохой» .EXE модуль, полученный из исходного текста для .COM модуля.

Шаг 2. Написать текст исходного .EXE модуля, который выполняет те же функции, что и модуль в Шаге 1, постройте и отладьте его. Таким образом, будет получен «хороший» .EXE.

Шаг 3. Сравнить исходные тексты для .COM и .EXE модулей. Ответить

на контрольные вопросы «Отличия исходных текстов COM и EXE программ».

Шаг 4. Запустить FAR и открыть (F3/F4) файл загрузочного модуля «хорошего» .COM и файл «плохого» .EXE в шестнадцатеричном виде. Затем открыть (F3/F4) файл загрузочного модуля «хорошего» .EXE и сравнить его с предыдущими файлами. Ответить на контрольные вопросы «Отличия форматов файлов COM и EXE модулей».

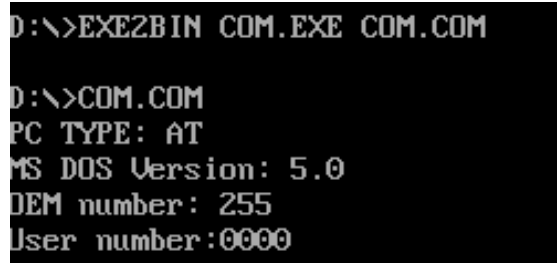
Шаг 5. Открыть отладчик TD.EXE и загрузить .COM. Ответить на контрольные вопросы «Загрузка COM модуля в основную память.»
Представить в отчете план загрузки модуля .COM в основную память.

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

ШАГ 7. Оформить отчет в соответствии с требованиями. В отчете необходимо привести скриншоты. Для файлов — их вид в шестнадцатеричном виде, для загрузочных модулей — в отладчике.

Выполнение работы

Шаг 1. Был написан текст исходного **.COM** модуля, который определяет РС и версию системы. Был построен «плохой» **.EXE**, полученный из исходного текста для **.COM** модуля.



```
D:\>EXE2BIN COM.EXE COM.COM
D:\>COM.COM
PC TYPE: AT
MS DOS Version: 5.0
OEM number: 255
User number: 0000
```

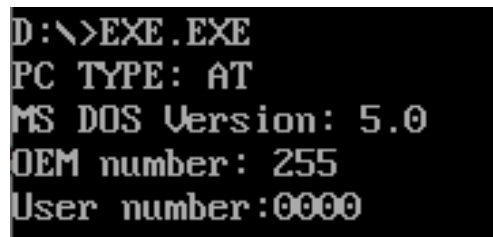
Рисунок 1. Демонстрация работы **.COM** модуля



```
D:\>COM.EXE
PC TYPE: PC
MS DOS Version: 5.0
OEM number: 255
User number: 0000
```

Рисунок 2. Демонстрация работы «плохого» **.EXE** модуля

Шаг 2. Был написан текст исходного **.EXE** модуля, который выполняет те же функции, что и модуль в Шаге 1.



```
D:\>EXE.EXE
PC TYPE: AT
MS DOS Version: 5.0
OEM number: 255
User number: 0000
```

Рисунок 3. Демонстрация работы исходного **.EXE** модуля

Шаг 3. Было произведено сравнение исходных текстов для **.COM** и **.EXE** модулей. На основе этого были даны ответы на контрольные вопросы «Отличия исходных текстов COM и EXE программ».

1. Сколько сегментов должна содержать COM-программа?

Ответ: COM-программа содержит один сегмент, в нем хранятся данные и код,

стек же генерируется автоматически и в располагается в конце сегмента.

2. EXE-программа?

Ответ: EXE-программа содержит несколько сегментов (данных, кодов, стека), но должна содержать как минимум один.

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

Ответ: Обязательна должны быть:

а) директива ORG 100h – потому как адресация имеет смещение в 256 байт от нулевого адреса (т.е. после PSP размера 100h).

б) директива ASSUME – чтобы сегмент данных и сегмент кода указывали на один общий сегмент.

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

Ответ: Нельзя использовать команды, которые берут адрес сегмента (mov <регистр>, seg <имя сегмента>). Это происходит из-за того, что в .COM-файле нет таблицы настроек, в которой хранится информация о типе адресов и их расположении в коде. Эта таблица подключается на этапе линковки (в COM-программе это не происходит).

Шаг 4. Были открыты через FAR файл загрузочного модуля «хорошего» .COM и файл «плохого» .EXE в шестнадцатеричном виде. Затем открыт файл загрузочного модуля «хорошего» .EXE и произведено его сравнение с предыдущими файлами. На основе этого были даны ответы на контрольные вопросы «Отличия форматов файлов COM и EXE модулей».

C:\Dos\box\COM.COM																
0000000000:	E9	F6	01	50	43	20	54	59	50	45	3A	20	50	43	0D 0A	éö@PC TYPE: PC
0000000010:	24	50	43	20	54	59	50	45	3A	20	50	43	2F	58	54 0D	\$PC TYPE: PC/XT
0000000020:	0A	24	50	43	20	54	59	50	45	3A	20	41	54	0D 0A 24	\$PC TYPE: AT	
0000000030:	50	43	20	54	59	50	45	3A	20	50	53	32	20	6D 6F 64	PC TYPE: PS2 mod	
0000000040:	65	6C	20	33	30	0D 0A 24	50	43	20	54	59	50	45	3A	e1 30	\$PC TYPE:
0000000050:	20	50	53	32	20	6D 6F 64	65	6C	20	35	30	20	6F 72	PS2 model 50 or		
0000000060:	20	36	30	0D 0A 24	50	43	20	54	59	50	45	3A	20	50	60	\$PC TYPE: P
0000000070:	53	32	20	6D 6F 64	65	6C	20	38	30	0D 0A 24	50	43	S2 model 80	\$PC		
0000000080:	20	54	59	50	45	3A 20 50	43	6A	72	0D 0A 24	50	43	TYPE: PCjr	\$PC		
0000000090:	20	54	59	50	45	3A 20 50	43	20	43	6F 6E 76	65	72	TYPE: PC Conver			
00000000A0:	74	69	62	6C 65 0D 0A 24	4D	53	20	44	4F 53	20	56	tible	\$MS DOS V			
00000000B0:	65	72	73	69 6F 6E 3A 20	20	2E	20	20	0D 0A 24	4F	ersion:					
00000000C0:	45	4D	20	6E 75 6D 62 65	72	3A	20	20	20	0D 0A	EM number:					
00000000D0:	24	55	73	65 72 20 6E 75	6D	62	65	72	3A	20	20	20	\$User number:			
00000000E0:	20	0D 0A 24 24 0F 3C 09	76	02	04	07	04	30	C3	51						
00000000F0:	8A	E0	E8	EF FF 86 C4 B1	04	D2	E8	E8	E6	FF	59	C3	Šäëÿ†Ä±			
0000000100:	53	8A	FC	E8 E9 FF 88 25	4F	88	05	4F	8A	C7	E8	DE	SŠüëÿ~%0^			
0000000110:	FF	88	25	4F 88 05 5B C3	51	52	32	E4	33	D2	B9	0A	ÿ~%0^+ [ÄQR2ä30¹			
0000000120:	00	F7	F1	80 CA 30 88 14	4E	33	D2	3D	0A	00	73	F1	÷ñ€Ê0^ĴN30= sñ			
0000000130:	3C	00	74	04 0C 30 88 04	5A	59	C3	50	06	52	B8	00	< t90^ZYÄP			
0000000140:	F0	8E	C0	26 A0 FE FF 3C	FF	74	20	3C	FE	74	22	3C	ðŽ& pÿ<ÿt <bt"<			
0000000150:	FB	74	1E	3C FC 74 20 3C	FA	74	22	3C	FC	74	24	3C	ût▲<üt <üt"<üt\$<			
0000000160:	F8	74	26	3C FD 74 28 3C	F9	74	2A	BA	03	01	EB	2B	øt&<ÿt(<üt*0ë+			
0000000170:	90	BA	11	01 EB 25 90 BA	22	01	EB	1F	90	BA	30	01	▯◊◊ë%▯◊"0ë▯◊00			
0000000180:	EB	19	90	BA 48 01 EB 13	90	BA	66	01	EB	0D	90	BA	ë↓▯◊H0ë!!▯◊f0ë▯◊			
0000000190:	7E	01	EB	07 90 BA 8E 01	EB	01	90	B4	09	CD	21	5A	~0ë◊▯Ž0ë0▯◊óÍ!Z			
00000001A0:	07	58	C3	50 B4 09 CD 21	58	C3	50	53	51	57	56	2B	•XÄP´óÍ!XÄPSQWV+			
00000001B0:	C0	B4	30	CD 21 BE A8 01	83	C6	10	E8	5A	FF	8A	C4	Ä´0Í!¼"0fÄ→èZÿŠÄ			
00000001C0:	83	C6	03	E8 52 FF BA A8	01	E8	D7	FF	BE	BF	01	83	fÄ♥èRÿ◊"0ë×ÿ¼¿0f			
00000001D0:	C6	0E	8A	C7 E8 41 FF BA	BF	01	E8	C6	FF	BF	D1	01	ÆðŠÇèAÿ◊¿0ëÆÿ¿Ñ0			
00000001E0:	83	C7	0F	8B C1 E8 18 FF	8A	C3	E8	02	FF	BA	D1	01	fÇ◊◊Äè↑ÿŠÄè0ÿ◊Ñ0			
00000001F0:	E8	B0	FF	5E 5F 59 5B 58	C3	E8	3F	FF	E8	AB	FF	32	è°ÿ^_Y[XÄè?ÿè«ÿ2			
0000000200:	C0	B4	4C	CD 21									Ä´LÍ!			

Рисунок 4. Структура COM-файла в 16-ричном виде

C:\Dos\box\COM.EXE																		
0000000000:	4D	5A	05	01	03	00	00	00	20	00	00	00	FF	FF	00	00	MZ+❤	ÿÿ
0000000010:	00	00	00	00	00	01	00	00	3E	00	00	00	01	00	FB	50	0 > 0 ûP	
0000000020:	6A	72	00	00	00	00	00	00	00	00	00	00	00	00	00	00	j r	
0000000030:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000040:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000050:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000060:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000070:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000080:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000090:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000100:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000110:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000120:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000130:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

Рисунок 5. Структура «плохого» EXE-файла в 16-ричном виде

00000000140:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000150:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000160:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000170:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000180:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000190:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000001A0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000001B0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000001C0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000001D0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000001E0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000001F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000200:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000210:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000220:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000230:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000240:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000250:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000260:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000270:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000280:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000290:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000002A0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000002B0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000002C0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000002D0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000002E0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000002F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000300:	E9 F6 01 50 43 20 54 59	50 45 3A 20 50 43 0D 0A	éö0PC TYPE: PC)☐
00000000310:	24 50 43 20 54 59 50 45	3A 20 50 43 2F 58 54 0D	\$PC TYPE: PC/XT)☐
00000000320:	0A 24 50 43 20 54 59 50	45 3A 20 41 54 0D 0A 24	☐\$PC TYPE: AT)☐\$
00000000330:	50 43 20 54 59 50 45 3A	20 50 53 32 20 6D 6F 64	PC TYPE: PS2 mod
00000000340:	65 6C 20 33 30 0D 0A 24	50 43 20 54 59 50 45 3A	el 30)☐\$PC TYPE:
00000000350:	20 50 53 32 20 6D 6F 64	65 6C 20 35 30 20 6F 72	PS2 model 50 or
00000000360:	20 36 30 0D 0A 24 50 43	20 54 59 50 45 3A 20 50	60)☐\$PC TYPE: P
00000000370:	53 32 20 6D 6F 64 65 6C	20 38 30 0D 0A 24 50 43	S2 model 80)☐\$PC
00000000380:	20 54 59 50 45 3A 20 50	43 6A 72 0D 0A 24 50 43	TYPE: PCjr)☐\$PC
00000000390:	20 54 59 50 45 3A 20 50	43 20 43 6F 6E 76 65 72	TYPE: PC Conver
000000003A0:	74 69 62 6C 65 0D 0A 24	4D 53 20 44 4F 53 20 56	tible)☐\$MS DOS V
000000003B0:	65 72 73 69 6F 6E 3A 20	20 2E 20 20 0D 0A 24 4F	ersion: .)☐\$0
000000003C0:	45 4D 20 6E 75 6D 62 65	72 3A 20 20 20 0D 0A	EM number:)☐
000000003D0:	24 55 73 65 72 20 6E 75	6D 62 65 72 3A 20 20 20	\$User number:
000000003E0:	20 0D 0A 24 24 0F 3C 09	76 02 04 07 04 30 C3 51)☐\$ \$a<ov0♦♦♦0ÄQ
000000003F0:	8A E0 E8 EF FF 86 C4 B1	04 D2 E8 E8 E6 FF 59 C3	Šaëiÿ†Ä±♦0èèæÿYÄ
00000000400:	53 8A FC E8 E9 FF 88 25	4F 88 05 4F 8A C7 E8 DE	sŠüèëÿ~%0^+0SÇèb
00000000410:	FF 88 25 4F 88 05 5B C3	51 52 32 E4 33 D2 B9 0A	ÿ~%0^+ [ÄQR2ä30¹☐
00000000420:	00 F7 F1 80 CA 30 88 14	4E 33 D2 3D 0A 00 73 F1	÷ñ€Ê0^ŸJN30=☐ sñ
00000000430:	3C 00 74 04 0C 30 88 04	5A 59 C3 50 06 52 B8 00	< t♦90^♦ZYÄP♣R,

Рисунок 6. Структура «плохого» EXE-файла в 16-ричном виде

0000000440: F0 8E C0 26 A0 FE FF 3C	FF 74 20 3C FE 74 22 3C	đŽ& pÿ<ÿt <pt"<
0000000450: FB 74 1E 3C FC 74 20 3C	FA 74 22 3C FC 74 24 3C	ût▲<ût <ût"<ût\$<
0000000460: F8 74 26 3C FD 74 28 3C	F9 74 2A BA 03 01 EB 2B	øt&<ÿt(<ût*°♥@ë+
0000000470: 90 BA 11 01 EB 25 90 BA	22 01 EB 1F 90 BA 30 01	▣°◀@ë%▣°"@ë▼▣°00
0000000480: EB 19 90 BA 48 01 EB 13	90 BA 66 01 EB 0D 90 BA	ë↓▣°H@ë!!▣°f@ë)▣°
0000000490: 7E 01 EB 07 90 BA 8E 01	EB 01 90 B4 09 CD 21 5A	~@ë•▣°Ž@ë@▣°°oÍ!Z
00000004A0: 07 58 C3 50 B4 09 CD 21	58 C3 50 53 51 57 56 2B	•XĀP°oÍ!XĀPSQW+
00000004B0: C0 B4 30 CD 21 BE A8 01	83 C6 10 E8 5A FF 8A C4	À°0Í!%°°fÆ°èZÿŠĀ
00000004C0: 83 C6 03 E8 52 FF BA A8	01 E8 D7 FF BE BF 01 83	fÆ♥èRÿ°°@è×ÿ%¿@f
00000004D0: C6 0E 8A C7 E8 41 FF BA	BF 01 E8 C6 FF BF D1 01	ÆŖŠÇèAÿ°¿@èÆÿ¿Ñ@
00000004E0: 83 C7 0F 8B C1 E8 18 FF	8A C3 E8 02 FF BA D1 01	fÇ°◀Aè↑ÿŠĀè@ÿ°Ñ@
00000004F0: E8 B0 FF 5E 5F 59 5B 58	C3 E8 3F FF E8 AB FF 32	è°ÿ^_Y[XĀè?ÿè«ÿ2
0000000500: C0 B4 4C CD 21		À°LÍ!

Рисунок 7. Структура «плохого» EXE-файла в 16-ричном виде

0000000000: 4D 5A 18 00 07 00 01 00	20 00 00 00 FF FF 00 00	MZ↑ • @ яя
0000000010: 00 08 5C AD 15 01 8F 00	1E 00 00 00 01 00 18 01	▣\~\$@U ▲ @ ↑@
0000000020: 8F 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	U
0000000030: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000040: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000050: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000060: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000070: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000080: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000090: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000A0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000B0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000C0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000D0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000E0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000000F0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000100: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000110: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000120: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000130: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000140: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000150: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000160: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000170: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000180: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
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	

Рисунок 8. Структура «хорошего» EXE-файла в 16-ричном виде

00000009E0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000009F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000A00:	50 43 20 54 59 50 45 3A	20 50 43 0D 0A 24 50 43	PC TYPE: PC/XT
0000000A10:	20 54 59 50 45 3A 20 50	43 2F 58 54 0D 0A 24 50	TYPE: PC/XT
0000000A20:	43 20 54 59 50 45 3A 20	41 54 0D 0A 24 50 43 20	C TYPE: AT
0000000A30:	54 59 50 45 3A 20 50 53	32 20 6D 6F 64 65 6C 20	TYPE: PS2 model
0000000A40:	33 30 0D 0A 24 50 43 20	54 59 50 45 3A 20 50 53	30
0000000A50:	32 20 6D 6F 64 65 6C 20	35 30 20 6F 72 20 36 30	2 model 50 or 60
0000000A60:	0D 0A 24 50 43 20 54 59	50 45 3A 20 50 53 32 20	TYPE: PS2
0000000A70:	6D 6F 64 65 6C 20 38 30	0D 0A 24 50 43 20 54 59	model 80
0000000A80:	50 45 3A 20 50 43 6A 72	0D 0A 24 50 43 20 54 59	PE: PCjr
0000000A90:	50 45 3A 20 50 43 20 43	6F 6E 76 65 72 74 69 62	PE: PC Convertib
0000000AA0:	6C 65 0D 0A 24 4D 53 20	44 4F 53 20 56 65 72 73	le
0000000AB0:	69 6F 6E 3A 20 20 2E 20	20 0D 0A 24 4F 45 4D 20	ion: .
0000000AC0:	6E 75 6D 62 65 72 3A 20	20 20 20 0D 0A 24 55 73	number:
0000000AD0:	65 72 20 6E 75 6D 62 65	72 3A 20 20 20 20 0D 0A	er number:
0000000AE0:	24 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	\$
0000000AF0:	24 0F 3C 09 76 02 04 07	04 30 C3 51 8A E0 E8 EF	\$<ov♦♦♦ГQлбаип
0000000B00:	FF 86 C4 B1 04 D2 E8 E8	E6 FF 59 C3 53 8A FC E8	яТД±♦ТиижяYGSльи
0000000B10:	E9 FF 88 25 4F 88 05 4F	8A C7 E8 DE FF 88 25 4F	йя€%0€♦0л3и0я€%0
0000000B20:	88 05 5B C3 51 52 32 E4	33 D2 B9 0A 00 F7 F1 80	€♦[ГQR2д3Тл№ чсЪ
0000000B30:	CA 30 88 14 4E 33 D2 3D	0A 00 73 F1 3C 00 74 04	K0€JN3T= sc < t♦
0000000B40:	0C 30 88 04 5A 59 C3 50	06 52 B8 00 F0 8E C0 26	90€♦ZYГР♦Rё рfA&
0000000B50:	A0 FE FF 3C FF 74 20 3C	FE 74 22 3C FB 74 1E 3C	юя<ят <ют"<ытΔ<
0000000B60:	FC 74 20 3C FA 74 22 3C	FC 74 24 3C F8 74 26 3C	ьт <ьт"<ьт\$<шт&<
0000000B70:	FD 74 28 3C F9 74 2A BA	00 00 EB 2B 90 BA 0E 00	эт(<шт*е л+ћєЃ
0000000B80:	EB 25 90 BA 1F 00 EB 1F	90 BA 2D 00 EB 19 90 BA	л%ћє▼ л▼ћє- л↓ћє
0000000B90:	45 00 EB 13 90 BA 63 00	EB 0D 90 BA 7B 00 EB 07	Е л!!ћєс ллћє{ л•
0000000BA0:	90 BA 8B 00 EB 01 90 B4	09 CD 21 5A 07 58 C3 50	ћє< л0ћгоН!Z•ХГР
0000000BB0:	B4 09 CD 21 58 C3 50 53	51 57 56 2B C0 B4 30 CD	гоН!ХГPSQWV+Ar0H
0000000BC0:	21 BE A5 00 83 C6 10 E8	5A FF 8A C4 83 C6 03 E8	!sГ' фЖ►иZялбдфЖ▼и
0000000BD0:	52 FF BA A5 00 E8 D7 FF	BE BC 00 83 C6 0E 8A C7	РяєГ' иЧяsj фЖлбЗ
0000000BE0:	E8 41 FF BA BC 00 E8 C6	FF BF CE 00 83 C7 0F 8B	иАяєј иЖяіО фЗα<
0000000BF0:	C1 E8 18 FF 8A C3 E8 02	FF BA CE 00 E8 B0 FF 5E	Би↑ялГи0яєО и°я^
0000000C00:	5F 59 5B 58 C3 2B C0 B8	80 00 8E D8 E8 38 FF E8	_Y[ХГ+АёЪ fши8яи
0000000C10:	A4 FF 32 C0 B4 4C CD 21		ня2ArLN!

Рисунок 9. Структура «хорошего» EXE-файла в 16-ричном виде

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

Ответ. В COM-файле все данные размещаются в одном сегменте, он включает в себя инструкции процессора, директивы и описания переменных, адресация в COM-файле начинается с 0h.

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

Ответ. Структура «плохого» EXE-файла нарушена (состоит из одного сегмента, в котором хранятся и данные и код, а сегмента стека вообще нет). Код располагается с адреса 300h. С адреса 0 располагается таблица настроек.

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

Ответ: структура «хорошего» EXE-файла состоит из трех сегментов (сегменты данных, стека и кода). Отсутствует директива ORG 100h (код начинается без смещения).

Шаг 5. Был открыт отладчик TD.EXE и загружен .COM. На основе этого были даны ответы на контрольные вопросы «Загрузка COM модуля в основную память».

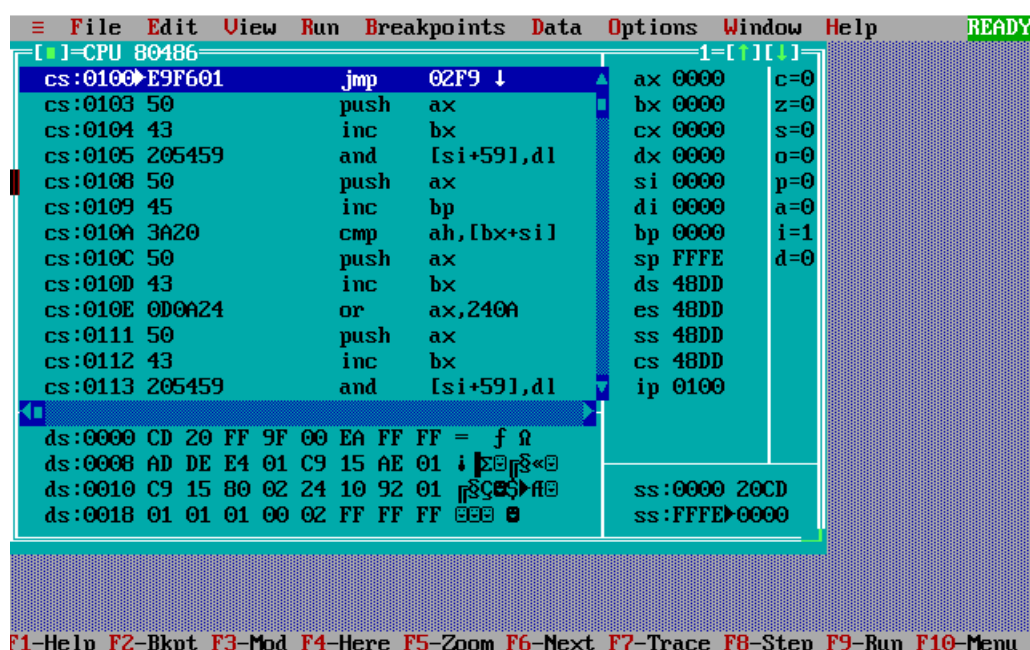


Рисунок 10. COM-файл в отладчике TD

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

Ответ: В начале выделяется свободный сегмент, с нулевого адреса этого сегмента в ОП загружается PSP. Сам код располагается после PSP, т.е. с адреса 100h.

2. Что располагается с адреса 0?

Ответ: PSP.

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

Ответ: все сегментные регистры имеют одно и то же значение — 48DD. Они указывают на начало PSP.

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

Ответ: в COM-программе стек генерируется автоматически. Под стек отведен весь сегмент программы. Он занимает адреса с 0000h до FFFEH (в начале указатель SP расположен в FFFEH).

Шаг 6. Был открыт отладчик TD.EXE и загружен «хороший» .EXE. На основе этого были даны ответы на контрольные вопросы «Загрузка «хорошего» EXE модуля в основную память».

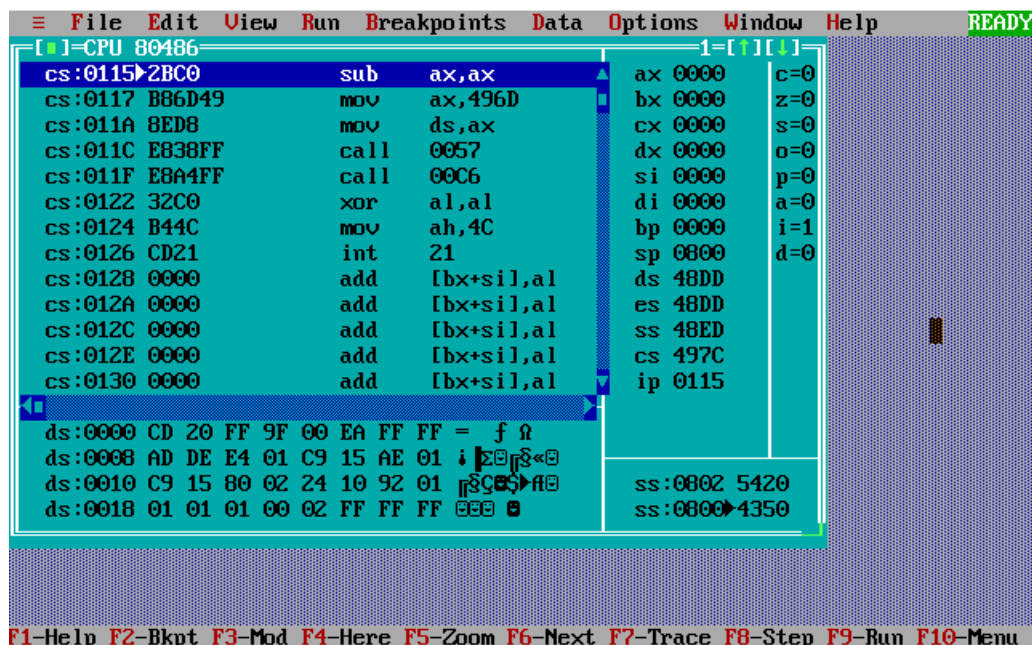


Рисунок 11. Хороший EXE-файл в отладчике TD

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

Ответы: так же в ОП загружается PSP с нулевого адреса, после чего

загружается сам модуль файла. Сегментные регистры: DS, ES – 48DD, SS – 48ED, CS – 497C.

2. На что указывают регистры DS и ES?

Ответ: на начало PSP.

3. Как определяется стек?

Ответ: стек определяется пользователем с помощью директивы SEGMENT STACK.

4. Как определяется точка входа?

Ответ: точка входа определяется с помощью директивы END.

Вывод.

В ходе выполнения лабораторной работы было проведено исследование различий в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

Приложение А

Исходный код программы

Название файла: com.asm

```
TESTPC    SEGMENT

            ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
            ORG 100H
START:     JMP BEGIN

PC db 'PC TYPE: PC', 0DH, 0AH, '$'
PC_XT db 'PC TYPE: PC/XT', 0DH, 0AH, '$'
AT db 'PC TYPE: AT', 0DH, 0AH, '$'
PS2_MODEL_30 db 'PC TYPE: PS2 model 30', 0DH, 0AH, '$'
PS2_MODEL_50_OR_60 db 'PC TYPE: PS2 model 50 or 60', 0DH, 0AH, '$'
PS2_MODEL_80 db 'PC TYPE: PS2 model 80', 0DH, 0AH, '$'
PCjr db 'PC TYPE: PCjr', 0DH, 0AH, '$'
PC_CONVERTIBLE db 'PC TYPE: PC Convertible', 0DH, 0AH, '$'
DOS_VERSION db 'MS DOS Version: . ', 0DH, 0AH, '$'
OEM_NUMBER db 'OEM number:   ', 0DH, 0AH, '$'
USER_NUMBER db 'User number:  ', 0DH, 0AH, '$'

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_l
or AL, 30h
mov [SI], AL
end_l: pop DX
pop CX
ret*

BYTE_TO_DEC ENDP

PRINT_PC_TYPE PROC NEAR

*push AX
push ES*

push DX

mov AX, 0F000h
mov ES, AX
mov AL, ES:[0FFFEh]

cmp AL, 0FFh
je pc_type

cmp AL, 0FEh
je pc_xt_type

cmp AL, 0FBh
je pc_xt_type

cmp AL, 0FCh
je at_type

cmp AL, 0FAh
je ps2_model_30_type

cmp AL, 0FCh
je ps2_model_50_or_60_type

cmp AL, 0F8h
je ps2_model_80_type

cmp AL, 0FDh
je pcjr_type

cmp AL, 0F9h
je pc_convertible_type

pc_type:
mov DX, offset PC
jmp PRINT_MESSAGE

pc_xt_type:
mov DX, offset PC_XT
jmp PRINT_MESSAGE

at_type:
mov DX, offset AT
jmp PRINT_MESSAGE


```

ps2_model_30_type:
    mov DX, offset PS2_MODEL_30
    jmp PRINT_MESSAGE

ps2_model_50_or_60_type:
    mov DX, offset PS2_MODEL_50_OR_60
    jmp PRINT_MESSAGE

ps2_model_80_type:
    mov DX, offset PS2_MODEL_80
    jmp PRINT_MESSAGE

pcjr_type:
    mov DX, offset PCjr
    jmp PRINT_MESSAGE

pc_convertible_type:
    mov DX, offset PC_CONVERTIBLE
    jmp PRINT_MESSAGE

PRINT_MESSAGE:
    mov AH, 09h
    int 21h

    pop DX
    pop ES
    pop AX

    ret
PRINT_PC_TYPE ENDP

PRINT_MES_SYS PROC near
    push AX
    mov AH, 09h
    int 21h
    pop AX
    ret
PRINT_MES_SYS ENDP

PRINT_SYSTEM_VERSION PROC near
    push AX
    push BX
    push CX

```

push DI
push SI

sub AX, AX
mov AH, 30h
int 21h

mov SI, offset DOS_VERSION
add SI, 16
call BYTE_TO_DEC
mov AL, AH ; AH - DOS VERSION
add SI, 3
call BYTE_TO_DEC
mov DX, offset DOS_VERSION
call PRINT_MES_SYS

mov SI, offset OEM_NUMBER
add SI, 14
mov AL, BH ; BH - OEM NUMBER
call BYTE_TO_DEC
mov DX, offset OEM_NUMBER
call PRINT_MES_SYS

mov DI, offset USER_NUMBER
add DI, 15
mov AX, CX
call WRD_TO_HEX
mov AL, BL
call BYTE_TO_HEX
mov DX, offset USER_NUMBER
call PRINT_MES_SYS

pop SI
pop DI
pop CX
pop BX
pop AX

ret
PRINT_SYSTEM_VERSION ENDP

BEGIN:
call PRINT_PC_TYPE
call PRINT_SYSTEM_VERSION

```
xor AL, AL  
mov AH, 4Ch  
int 21h
```

```
TESTPC ENDS  
END START
```

Название файла exe.asm

MYSTACK SEGMENT STACK

DW 1024 DUP(?)

MYSTACK ENDS

DATA SEGMENT

PC db 'PC TYPE: PC', 0DH, 0AH, '\$'

PC_XT db 'PC TYPE: PC/XT', 0DH, 0AH, '\$'

AT db 'PC TYPE: AT', 0DH, 0AH, '\$'

PS2_MODEL_30 db 'PC TYPE: PS2 model 30', 0DH, 0AH, '\$'

PS2_MODEL_50_OR_60 db 'PC TYPE: PS2 model 50 or 60', 0DH, 0AH, '\$'

PS2_MODEL_80 db 'PC TYPE: PS2 model 80', 0DH, 0AH, '\$'

PCjr db 'PC TYPE: PCjr', 0DH, 0AH, '\$'

PC_CONVERTIBLE db 'PC TYPE: PC Convertible', 0DH, 0AH, '\$'

DOS_VERSION db 'MS DOS Version: . ', 0DH, 0AH, '\$'

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

USER_NUMBER db 'User number: ', 0DH, 0AH, '\$'

DATA ENDS

CODE SEGMENT

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

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_l
    or AL, 30h
    mov [SI], AL
end_l: pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

PRINT_PC_TYPE PROC NEAR
    push AX

```

push ES
push DX

mov AX, 0F000h
mov ES, AX
mov AL, ES:[0FFFEh]

cmp AL, 0FFh
je pc_type

cmp AL, 0FEh
je pc_xt_type

cmp AL, 0FBh
je pc_xt_type

cmp AL, 0FCh
je at_type

cmp AL, 0FAh
je ps2_model_30_type

cmp AL, 0FCh
je ps2_model_50_or_60_type

cmp AL, 0F8h
je ps2_model_80_type

cmp AL, 0FDh
je pcjr_type

cmp AL, 0F9h
je pc_convertible_type

pc_type:
mov DX, offset PC
jmp PRINT_MESSAGE

pc_xt_type:
mov DX, offset PC_XT
jmp PRINT_MESSAGE

at_type:
mov DX, offset AT

```

        jmp PRINT_MESSAGE

ps2_model_30_type:
    mov DX, offset PS2_MODEL_30
    jmp PRINT_MESSAGE

ps2_model_50_or_60_type:
    mov DX, offset PS2_MODEL_50_OR_60
    jmp PRINT_MESSAGE

ps2_model_80_type:
    mov DX, offset PS2_MODEL_80
    jmp PRINT_MESSAGE

pcjr_type:
    mov DX, offset PCjr
    jmp PRINT_MESSAGE

pc_convertible_type:
    mov DX, offset PC_CONVERTIBLE
    jmp PRINT_MESSAGE

PRINT_MESSAGE:
    mov AH, 09h
    int 21h

    pop DX
    pop ES
    pop AX

    ret
PRINT_PC_TYPE ENDP

PRINT_MES_SYS PROC near
    push AX
    mov AH, 09h
    int 21h
    pop AX
    ret
PRINT_MES_SYS ENDP

PRINT_SYSTEM_VERSION PROC near
    push AX
    push BX

```


push CX
push DI
push SI

sub AX, AX
mov AH, 30h
int 21h

mov SI, offset DOS_VERSION
add SI, 16
call BYTE_TO_DEC
mov AL, AH ; AH - DOS VERSION
add SI, 3
call BYTE_TO_DEC
mov DX, offset DOS_VERSION
call PRINT_MES_SYS

mov SI, offset OEM_NUMBER
add SI, 14
mov AL, BH ; BH - OEM NUMBER
call BYTE_TO_DEC
mov DX, offset OEM_NUMBER
call PRINT_MES_SYS

mov DI, offset USER_NUMBER
add DI, 15
mov AX, CX
call WRD_TO_HEX
mov AL, BL
call BYTE_TO_HEX
mov DX, offset USER_NUMBER
call PRINT_MES_SYS

pop SI
pop DI
pop CX
pop BX
pop AX

ret
PRINT_SYSTEM_VERSION ENDP

MAIN PROC far
sub AX, AX

```
mov AX, DATA
mov DS, AX

call PRINT_PC_TYPE
call PRINT_SYSTEM_VERSION

xor AL, AL
mov AH, 4Ch
int 21h
MAIN ENDP

CODE ENDS
END MAIN
```