

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

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

Студент гр. 9383

Крейсманн К.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

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

Задание.

Шаг 1. Напишите текст исходного .COM модуля, который определяет тип РС и версию системы. Это довольно простая задача и для тех, кто уже имеет опыт программирования на ассемблере, это будет небольшой разминкой. Для тех, кто раньше не сталкивался с программированием на ассемблере, это неплохая задача для первого опыта. За основу возьмите шаблон, приведенный в разделе «Основные сведения». Необходимые сведения о том, как извлечь требуемую информацию, представлены в следующем разделе. Ассемблерная программа должна читать содержимое предпоследнего байта ROM BIOS, по таблице, сравнивая коды, определять тип РС и выводить строку с названием модели. Если код не совпадает ни с одним значением, то двоичный код переводиться в символьную строку, содержащую запись шестнадцатеричного числа и выводиться на экран в виде соответствующего сообщения. Затем определяется версия системы. Ассемблерная программа должна по значениям регистров AL и AH формировать текстовую строку в формате xx.yy, где xx - номер основной версии, а yy - номер модификации в десятичной системе счисления, формировать строки с серийным номером OEM и серийным номером пользователя. Полученные строки выводятся на экран. Отладьте полученный исходный модуль. Результатом выполнения этого шага будет «хороший» .COM модуль, а также необходимо построить «плохой» .EXE, полученный из исходного текста для .COM модуля.

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

Шаг 3. Сравните исходные тексты для .COM и .EXE модулей. Ответьте на контрольные вопросы «Отличия исходных текстов COM и EXE программ».

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

Шаг 5. Откройте отладчик TD.EXE и загрузите .COM. Ответьте на контрольные вопросы «Загрузка COM модуля в основную память». Представьте в отчете план загрузки модуля .COM в основную память.

Шаг 6. Откройте отладчик TD.EXE и загрузите «хороший» .EXE. Ответьте на контрольные вопросы «Загрузка «хорошего» EXE модуля в основную память».

Шаг 7. Оформление отчета в соответствии с требованиями. В отчете необходимо привести скриншоты. Для файлов их вид в шестнадцатеричном виде, для загрузочных модулей – в отладчике

Контрольные вопросы по лабораторной работе

Отличия исходных текстов COM и EXE программ

- 1) Сколько сегментов должна содержать COM-программа?
- 2) EXE-программа?
- 3) Какие директивы должны обязательно быть в тексте COM-программы?
- 4) Все ли форматы команд можно использовать в COM-программе?

Отличия форматов файлов COM и EXE модулей

- 1) Какова структура файла COM? С какого адреса располагается код?
- 2) Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с адреса 0?
- 3) Какова структура файла «хорошего» EXE? Чем он отличается от файла «плохого» EXE?

Загрузка COM модуля в основную память

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

- 2) Что располагается с адреса 0?
- 3) Какие значения имеют сегментные регистры? На какие области памяти они указывают?
- 4) Как определяется стек? Какую область памяти он занимает? Какие адреса?

Загрузка «хорошего» EXE модуля в основную память

- 1) Как загружается «хороший» EXE? Какие значения имеют сегментные регистры?
- 2) На что указывают регистры DS и ES?
- 3) Как определяется стек?
- 4) Как определяется точка входа?

Выполнение работы

Шаг 1. Была разработана программа для модуля .COM (исходный код в приложении А.) Из нее, в результате ассемблирования и линковки был получен «плохой» .EXE модуль, а из него был получен «хороший» .COM модуль. Результаты работы см. на рисунках 1 и 2.

```
C:\>lab1_com.com
Type of PC : AT
DOS version: 5.0
OEM number: 255
User serial number: 000000h
```

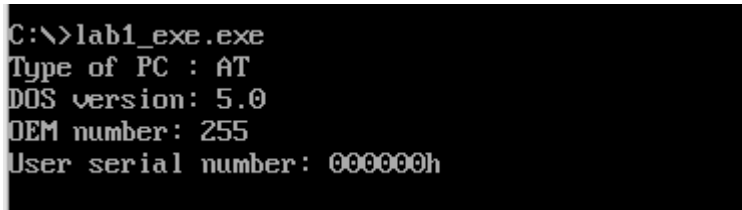
Рисунок 1 - Пример работы модуля .COM

```
C:\>lab1_com.exe

5 0
255
0+Type of PC : PC
0+Type of PC : PC
0+Type of PC : PC
0+Type of PC : PC
```

Рисунок 2 - Пример работы модуля "плохого" .EXE

Шаг 2. Также была разработана программа для «хорошего» .EXE модуля, исходных код находится в приложении Б. Результат работы модуля на рисунке 3.



```
C:\>lab1_exe.exe
Type of PC : AT
DOS version: 5.0
OEM number: 255
User serial number: 000000h
```

Рисунок 3 - Пример работы "хорошего" модуля .EXE

Шаг 3. Ответы на вопросы:

- 1) Сколько сегментов должна содержать COM-программа?

COM-программа должна содержать один сегмент, потому что данные и код находятся в одном сегменте, а стек устанавливается на последнюю ячейку сегмента автоматически.

- 2) EXE-программа?

EXE-программа должна содержать минимум 1 сегмент, допускается использование и более 3 сегментов. При этом сегменты стека, кода и данных отделены друг от друга.

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

В COM-программе должна быть директива `org 100h`, чтобы сместить адресацию на 256 байт. Это нужно, потому что первые 256 байт занимает PSP. Также, с помощью директивы `ASSUME` нужно привязать сегменты данных и кода в один сегмент.

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

В COM-программе не существует таблицы настроек, которая есть в EXE-программе (Relocation Table).

Поэтому команды вида `mov [register], seg [segment]` нельзя использовать.

Шаг 4.

На рисунке 4 - .COM модуль в шестнадцатеричном виде, на рисунках 5,6 – «плохой» .EXE модуль, на рисунках 7,8 – «хороший» .EXE модуль.

C:\OS\lab1\LAB1_COM.COM																h	1251	570	Col	0	100%	21:12
0000000000:	E9	2B	02	54	79	70	65	20	6F	66	20	50	43	20	3A	20						й+0Type of PC :
0000000010:	50	43	0D	0A	24	54	79	70	65	20	6F	66	20	50	43	20						PC:Type of PC
0000000020:	3A	20	50	43	2F	58	54	0D	0A	24	54	79	70	65	20	6F						: PC/XT:Type o
0000000030:	66	20	50	43	20	3A	20	41	54	0D	0A	24	54	79	70	65						f PC : AT:Type
0000000040:	20	6F	66	20	50	43	20	3A	20	50	53	32	20	6D	6F	64						of PC : PS2 mod
0000000050:	65	6C	20	33	30	0D	0A	24	54	79	70	65	20	6F	66	20						el 30:Type of
0000000060:	50	43	20	3A	20	50	53	32	20	6D	6F	64	65	6C	20	35						PC : PS2 model 5
0000000070:	30	20	6F	72	20	36	30	0D	0A	24	54	79	70	65	20	6F						0 or 60:Type o
0000000080:	66	20	50	43	20	3A	20	50	53	32	20	6D	6F	64	65	6C						f PC : PS2 model
0000000090:	20	38	30	0D	0A	24	54	79	70	65	20	6F	66	20	50	43						80:Type of PC
00000000A0:	20	3A	20	50	43	6A	72	0D	0A	24	54	79	70	65	20	6F						: PCjr:Type o
00000000B0:	66	20	50	43	3A	20	50	43	20	43	6F	6E	76	65	72	74						f PC: PC Convert
00000000C0:	69	62	6C	65	0D	0A	24	45	72	72	6F	72	20	0D	0A	24						ible:Error :\$
00000000D0:	44	4F	53	20	76	65	72	73	69	6F	6E	3A	20	20	2E	20						DOS version: .
00000000E0:	20	0D	0A	24	4F	45	4D	20	6E	75	6D	62	65	72	3A	20						:OEM number:
00000000F0:	20	20	0D	0A	24	55	73	65	72	20	73	65	72	69	61	6C						:User serial
0000000100:	20	6E	75	6D	62	65	72	3A	20	20	20	20	20	20	20	68						number: h
0000000110:	0D	0A	24	24	0F	3C	09	76	02	04	07	04	30	C3	51	8A						:<ov0♦♦0ГQЪ
0000000120:	E0	E8	EF	FF	86	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53						аипя†д±♦ТиижяYGS
0000000130:	8A	FC	E8	E9	FF	88	25	4F	88	05	4F	8A	C7	E8	DE	FF						льийя€%O€†OЪзиЮя
0000000140:	88	25	4F	88	05	5B	C3	51	52	32	E4	33	D2	B9	0A	00						€%O€†[ГQR2дЗТ№
0000000150:	F7	F1	80	CA	30	88	14	4E	33	D2	3D	0A	00	73	F1	3C						чсЪK0€JN3T= sc<
0000000160:	00	74	04	0C	30	88	04	5A	59	C3	B8	00	F0	8E	C0	BF						т♦%0€♦ZYГё рЪAї
0000000170:	FE	FF	26	8A	25	80	FC	FF	75	06	BA	03	01	EB	5F	90						юя&ь%Ъьяи€♥0л_ђ
0000000180:	80	FC	FE	75	06	BA	15	01	EB	54	90	80	FC	FB	75	06						Ъьюи€§0лТђЪьии€
0000000190:	BA	15	01	EB	49	90	80	FC	FC	75	06	BA	2A	01	EB	3E						е§0лIђЪьии€*0л>
00000001A0:	90	80	FC	FA	75	06	BA	3C	01	EB	33	90	80	FC	FC	75						ђЪьии€<0лЗђЪьи
00000001B0:	06	BA	58	01	EB	28	90	80	FC	F8	75	06	BA	96	01	EB						€X0л(ђЪьии€-0л
00000001C0:	1D	90	80	FC	FD	75	06	BA	AA	01	EB	12	90	80	FC	FD						↵ђЪьии€€0л†ђЪьэ
00000001D0:	75	06	BA	AA	01	EB	07	90	BA	C7	01	EB	01	90	B4	09						и€€0л•ђэ30л0ђГо
00000001E0:	CD	21	C3	B4	30	CD	21	BE	D0	01	83	C6	0D	50	E8	56						Н!Гг0Н!sР0ѓЖЉРиV
00000001F0:	FF	58	83	C6	03	8A	C4	E8	4D	FF	BA	D0	01	B4	09	CD						яXѓЖЉЉДиМяеР0ѓоН
0000000200:	21	BE	E4	01	83	C6	0E	8A	C7	E8	3B	FF	BA	E4	01	B4						!сд0ѓЖЉЉзи;яед0ѓ
0000000210:	09	CD	21	BF	F5	01	83	C7	19	8B	C1	E8	11	FF	8A	C3						оН!ix0ѓз↓<Би<яЉГ
0000000220:	E8	FB	FE	89	45	FE	BA	F5	01	B4	09	CD	21	C3	E8	39						иыю%Еюех0ѓоН!Ги9
0000000230:	FF	E8	AF	FF	32	C0	B4	4C	CD	21												яиїя2AГЛH!

Рисунок 4 - .COM

C:\OS\lab1\LAB1_COM.EXE								h	1251	1338	Col 0	0%	21:1					
000000000:	4D	5A	3A	01	03	00	00	00	20	00	00	00	FF	FF	00	00	MZ:♥	яя
0000000010:	00	00	E5	6D	00	01	00	00	1E	00	00	00	01	00	00	00	em ☹ ▲ ☹	
0000000020:	00	00	00	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	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		

Рисунок 5 - "плохой" .EXE №1

C:\OS\lab1\LAB1_COM.EXE																h	1251	1338	Col 0	100%	21:19	
00000002F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000300:	E9	2B	02	54	79	70	65	20	6F	66	20	50	43	20	3A	20	й+0Type of PC :					
0000000310:	50	43	0D	0A	24	54	79	70	65	20	6F	66	20	50	43	20	PCType of PC					
0000000320:	3A	20	50	43	2F	58	54	0D	0A	24	54	79	70	65	20	6F	: PC/XTType o					
0000000330:	66	20	50	43	20	3A	20	41	54	0D	0A	24	54	79	70	65	f PC : ATType					
0000000340:	20	6F	66	20	50	43	20	3A	20	50	53	32	20	6D	6F	64	of PC : PS2 mo					
0000000350:	65	6C	20	33	30	0D	0A	24	54	79	70	65	20	6F	66	20	el 30Type of					
0000000360:	50	43	20	3A	20	50	53	32	20	6D	6F	64	65	6C	20	35	PC : PS2 model5					
0000000370:	30	20	6F	72	20	36	30	0D	0A	24	54	79	70	65	20	6F	0 or 60Type o					
0000000380:	66	20	50	43	20	3A	20	50	53	32	20	6D	6F	64	65	6C	f PC : PS2 model					
0000000390:	20	38	30	0D	0A	24	54	79	70	65	20	6F	66	20	50	43	80Type of PC					
00000003A0:	20	3A	20	50	43	6A	72	0D	0A	24	54	79	70	65	20	6F	: PCjrType o					
00000003B0:	66	20	50	43	3A	20	50	43	20	43	6F	6E	76	65	72	74	f PC: PC Convert					
00000003C0:	69	62	6C	65	0D	0A	24	45	72	72	6F	72	20	0D	0A	24	ibleType					
00000003D0:	44	4F	53	20	76	65	72	73	69	6F	6E	3A	20	20	2E	20	DOS version: .					
00000003E0:	20	0D	0A	24	4F	45	4D	20	6E	75	6D	62	65	72	3A	20	Type OEM number:					
00000003F0:	20	20	0D	0A	24	55	73	65	72	20	73	65	72	69	61	6C	Type User serial					
0000000400:	20	6E	75	6D	62	65	72	3A	20	20	20	20	20	20	20	68	number:					
0000000410:	0D	0A	24	24	0F	3C	09	76	02	04	07	04	30	C3	51	8A	Type \$					

Рисунок 6 - "плохой" .EXE №2

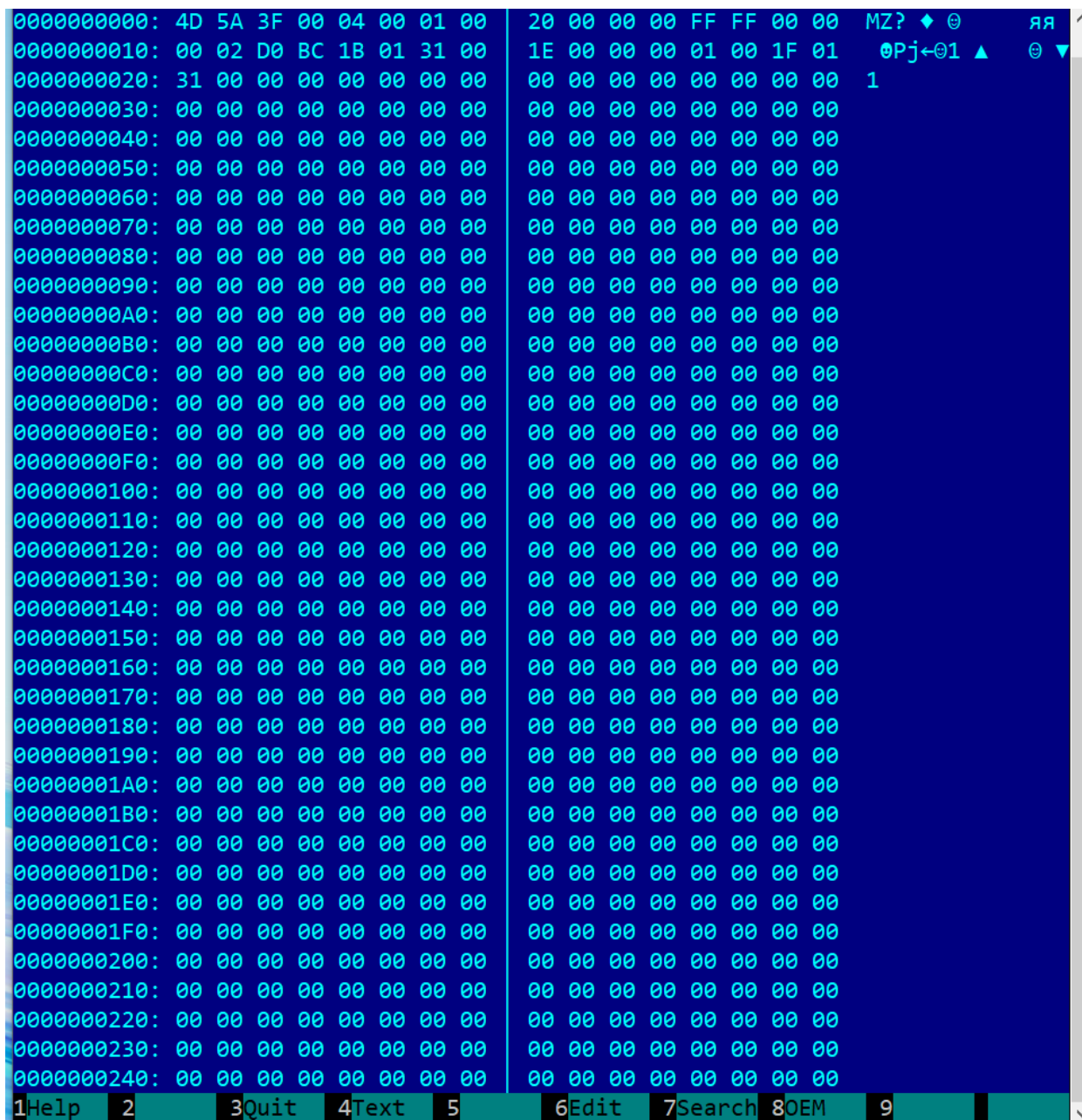


Рисунок 7 - "хороший" .EXE №1

00000003F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000400:	54 79 70 65 20 6F 66 20	50 43 20 3A 20 50 43 0D	Type of PC : PC
0000000410:	0A 24 54 79 70 65 20 6F	66 20 50 43 20 3A 20 50	Type of PC :
0000000420:	43 2F 58 54 0D 0A 24 54	79 70 65 20 6F 66 20 50	C/XTType of
0000000430:	43 20 3A 20 41 54 0D 0A	24 54 79 70 65 20 6F 66	C : ATType o
0000000440:	20 50 43 20 3A 20 50 53	32 20 6D 6F 64 65 6C 20	PC : PS2 model
0000000450:	33 30 0D 0A 24 54 79 70	65 20 6F 66 20 50 43 20	30Type of PC
0000000460:	3A 20 50 53 32 20 6D 6F	64 65 6C 20 35 30 20 6F	: PS2 model 50
0000000470:	72 20 36 30 0D 0A 24 54	79 70 65 20 6F 66 20 50	r 60Type of
0000000480:	43 20 3A 20 50 53 32 20	6D 6F 64 65 6C 20 38 30	C : PS2 model 8
0000000490:	0D 0A 24 54 79 70 65 20	6F 66 20 50 43 20 3A 20	Type of PC :
00000004A0:	50 43 6A 72 0D 0A 24 54	79 70 65 20 6F 66 20 50	PCjrType of
00000004B0:	43 3A 20 50 43 20 43 6F	6E 76 65 72 74 69 62 6C	C: PC Convertib
00000004C0:	65 0D 0A 24 45 72 72 6F	72 20 0D 0A 24 44 4F 53	eType of PC
00000004D0:	20 76 65 72 73 69 6F 6E	3A 20 20 2E 20 20 0D 0A	version: .
00000004E0:	24 4F 45 4D 20 6E 75 6D	62 65 72 3A 20 20 20 0D	OEM number:
00000004F0:	0A 24 55 73 65 72 20 73	65 72 69 61 6C 20 6E 75	User serial n
0000000500:	6D 62 65 72 3A 20 20 20	20 20 20 20 68 0D 0A 24	mber: h
0000000510:	24 0F 3C 09 76 02 04 07	04 30 C3 51 8A E0 E8 EF	\$<ov0000QЪаи
0000000520:	FF 86 C4 B1 04 D2 E8 E8	E6 FF 59 C3 53 8A FC E8	я†Д±◆ТиижяУГ\$ль
0000000530:	E9 FF 88 25 4F 88 05 4F	8A C7 E8 DE FF 88 25 4F	йя€%O€+Oльзиюя€%
0000000540:	88 05 5B C3 51 52 32 E4	33 D2 B9 0A 00 F7 F1 80	€+[ГQR2дЗТN\$ чс
0000000550:	CA 30 88 14 4E 33 D2 3D	0A 00 73 F1 3C 00 74 04	K0€JN3T= sc< t
0000000560:	0C 30 88 04 5A 59 C3 B8	00 F0 8E C0 BF FE FF 26	♀0€◆ZYГё рfAіюя
0000000570:	8A 25 80 FC FF 75 06 BA	00 00 EB 5F 90 80 FC FE	льЪъюи€е л_ђъ
0000000580:	75 06 BA 12 00 EB 54 90	80 FC FB 75 06 BA 12 00	и€е\$ лTђъьии€е\$
0000000590:	EB 49 90 80 FC FC 75 06	BA 27 00 EB 3E 90 80 FC	лIђъъьии€е' л>ђъ
00000005A0:	FA 75 06 BA 39 00 EB 33	90 80 FC FC 75 06 BA 55	ьии€е9 лзђъъьии€е
00000005B0:	00 EB 28 90 80 FC F8 75	06 BA 93 00 EB 1D 90 80	л(ђъъьии€е“ л•ђъ
00000005C0:	FC FD 75 06 BA A7 00 EB	12 90 80 FC FD 75 06 BA	ьэии€е\$ лђђъъэии€е
00000005D0:	A7 00 EB 07 90 BA C4 00	EB 01 90 B4 09 CD 21 C3	\$ л•ђъед л0ђъгон!
00000005E0:	B4 30 CD 21 BE CD 00 83	C6 0D 50 E8 56 FF 58 83	г0Н!sH ѓЖЈриVяX
00000005F0:	C6 03 8A C4 E8 4D FF BA	CD 00 B4 09 CD 21 BE E1	Ж♥льДиМяеН гон!s
0000000600:	00 83 C6 0E 8A C7 E8 3B	FF BA E1 00 B4 09 CD 21	ѓЖльзи;яеб гон
0000000610:	BF F2 00 83 C7 19 8B C1	E8 11 FF 8A C3 E8 FB FE	їт ѓЗ↓<Би<яльгиы
0000000620:	89 45 FE BA F2 00 B4 09	CD 21 C3 2B C0 50 B8 20	%Еюет гон!Г+АРё
0000000630:	00 8E D8 E8 31 FF E8 A7	FF 32 C0 B4 4C CD 21	ѳши1яи\$я2AгLN!

1Help 2 3Quit 4Text 5 6Edit 7Search 8OEM 9

Рисунок 8 - "хороший" .EXE №2

Ответы на вопросы:

- Какова структура файла COM? С какого адреса располагается код?
 Файл состоит из одного сегмента, содержит данные и машинные команды. COM-файл имеет ограничение в размере - 64 КБ. Код начинается с адреса 0h, при загрузке модуля устанавливается смещение 256 байт.
- Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с адреса 0?

Файл состоит из одного сегмента. Сегмент с кодом и данными начинается с адреса 300h. С адреса 0h расположена таблица разметки.

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

Файл состоит из трех сегментов – сегмента стека, данных и сегмента кода. Этот файл может иметь неограниченный размер. С 0h адреса расположена валидная таблица настроек. Код начинается с адреса 400h, после PSP и стека.

Различия: - разделение сегментов, - размер смещения (300h и 400h).

Шаг 5. Загрузка COM модуля в основную память.

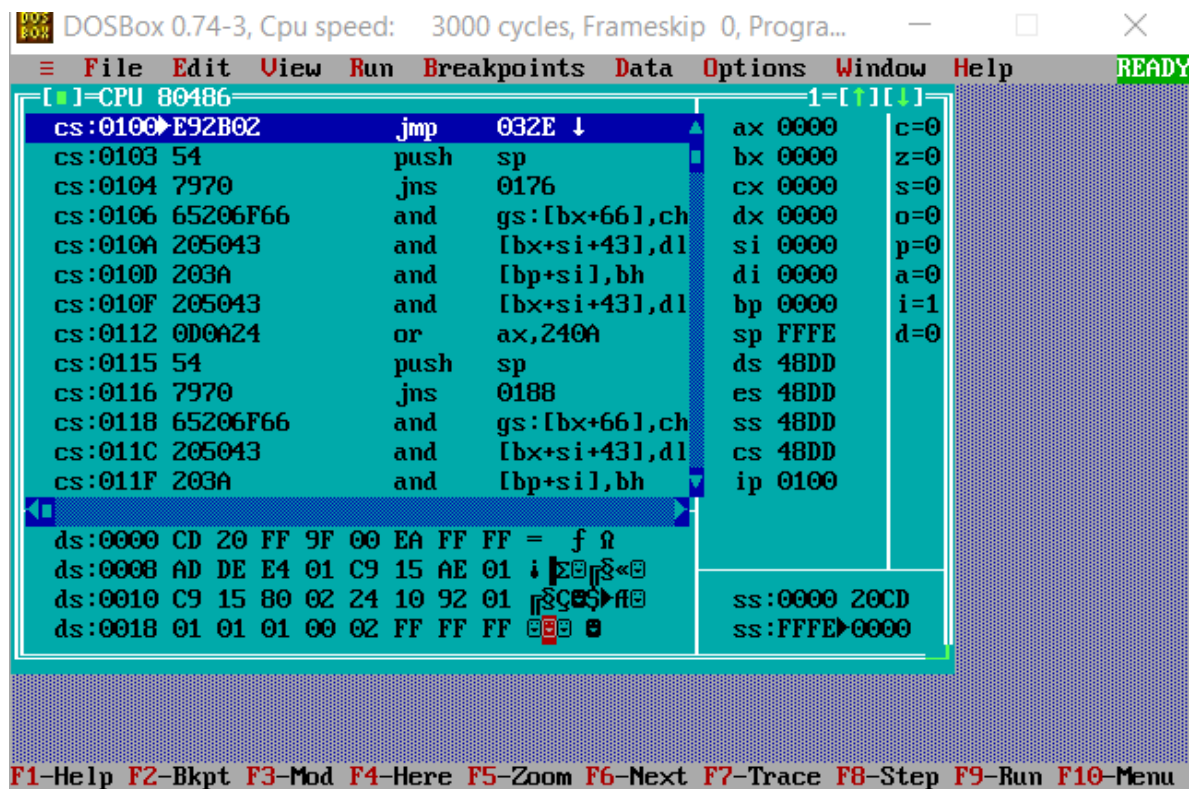


Рисунок 9 - Загрузка COM

Ответы на вопросы:

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

Сначала ищется место в оперативной памяти для COM модуля. Затем туда помещается PSP и по смещению 100h считанный модуль.

2) Что располагается с адреса 0?

PSP, размером 100h.

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

CS,ES,DS,SS указывают на PSP – 48DD, SP – на конец сегмента - FFFE.

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

Стек определяется автоматически при загрузке. Стек занимает всю область сегмента, возможны ситуации, когда стек накладывается на код или данные . Адреса расположены в диапазоне 0h-ffffh. Если для программы размер сегмента – 64 кб – недостаточный, то DOS устанавливает стек в конце памяти, выделяемой загрузчиком для исполнения программы.

Шаг 6. Загрузка «хорошего» EXE модуля в память

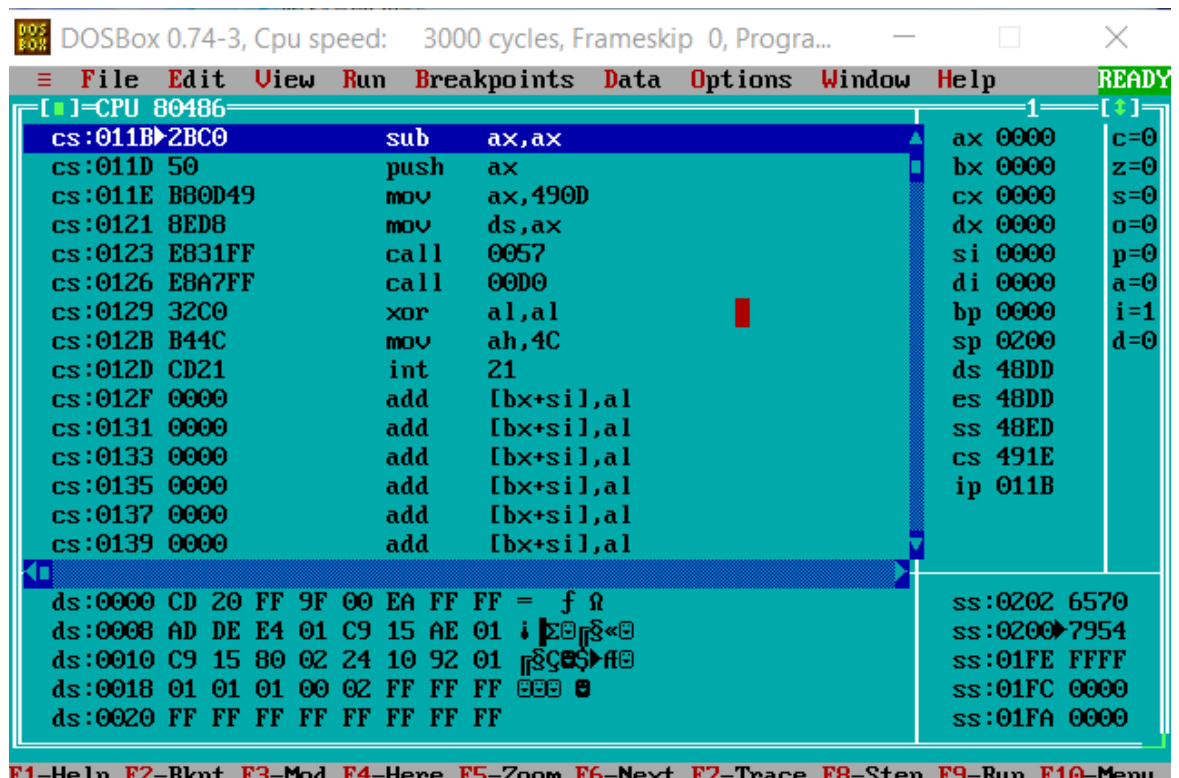


Рисунок 10 - Загрузка EXE

Ответы на вопросы:

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

Загрузка начинается с адреса 0100h относительно PSP. Стандартная часть заголовка считывается в память, определяется длина тела загрузочного модуля, определяется начальный сегмент, загрузочный модуль считывается в начальный сегмент, считывается таблица разметки в рабочую память. DS – 48DD, ES – 48DD, CS – 491E, SS – 48ED.

2) На что указывают регистры DS и ES?

На начало сегмента PSP.

3) Как определяется стек?

Стек определяется при помощи директивы .STACK. SS указывает на начало сегмента, SP на конец.

4) Как определяется точка входа?

С помощью директивы END.

Выводы.

Исследованы и сравнены структуры исходных текстов модулей типов .COM и .EXE, структуры файлов загрузочных модулей и способы их загрузки в основную память.

Приложение А.

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START:  JMP BEGIN
; Данные

PC db 'Type of PC : PC' , 0dh, 0ah, '$'
PCXT db 'Type of PC : PC/XT' , 0dh, 0ah, '$'
AT db 'Type of PC : AT', 0dh, 0ah, '$'
PS230 db 'Type of PC : PS2 model 30', 0dh, 0ah, '$'
PS250 db 'Type of PC : PS2 model 50 or 60', 0dh, 0ah, '$'
PS280 db 'Type of PC : PS2 model 80', 0dh, 0ah, '$'
PCjr db 'Type of PC : PCjr', 0dh, 0ah, '$'
PCCONVERTIBLE db 'Type of PC: PC Convertible' , 0dh, 0ah, '$'
ERROR db 'Error ', 0dh, 0ah, '$'
DOS_VERSION db 'DOS version: . ', 0dh, 0ah, '$'
OEM_SERIAL db 'OEM number: ', 0dh, 0ah, '$'
USER_SERIAL db 'User serial number:      h', 0dh, 0ah, '$'

; Процедуры
;-----
TETR_TO_HEX PROC near
    and AL, 0Fh
    cmp AL, 09
    jbe NEXT
    add AL, 07
NEXT:  add AL, 30h
    ret
TETR_TO_HEX ENDP
;-----
BYTE_TO_HEX PROC near
; байт в AL преобразуется 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- числа, DI- адрес последнего символа
    push BX
    mov BH, AH
    call BYTE_TO_HEX
    mov [DI], AH
    dec DI
    mov [DI], AL
    dec DI
    mov AL, BH
```

```

        call BYTE_TO_HEX
        mov [DI],AH
        dec DI
        mov [DI],AL
        pop BX
        ret
WRD_TO_HEX ENDP
;-----
BYTE_TO_DEC PROC near
; перевод в 10с/с , SI - адрес поля младшей цифры
        push CX
        push DX
        xor AH,AH
        xor DX,DX
        mov CX,10
loop_bd: div CX
        or DL,30h
        mov [SI],DL
        dec SI
        xor DX,DX
        cmp AX,10
        jae loop_bd
        cmp AL,00h
        je end_1
        or AL,30h
        mov [SI],AL
end_1:  pop DX
        pop CX
        ret
BYTE_TO_DEC ENDP
;-----
; КОД

PC_TYPE proc near
        mov ax, 0f000h
        mov es,ax
        mov di, 0ffffh
        mov ah, es:[di]

        ;PC
        cmp ah, 0FFh
        jne PCXTmetka1
        mov dx, offset PC
        jmp print_type

PCXTmetka1:
        cmp ah, 0FEh
        jne PCXTmetka2
        mov dx,offset PCXT
        jmp print_type

PCXTmetka2:
        cmp ah,0FBh
        jne ATmetka
        mov dx, offset PCXT
        jmp print_type

ATmetka:
        cmp ah,0FCh
        jne Model30metka
        mov dx,offset AT
        jmp print_type

Model30metka:

```



```

        cmp ah,0FAh
        jne Model150or60metka
        mov dx, offset PS230
        jmp print_type

Model150or60metka:
        cmp ah,0FCh
        jne Model180metka
        mov dx,offset PS250
        jmp print_type

Model180metka:
        cmp ah,0F8h
        jne PCjrmetka
        mov dx,offset PCjr
        jmp print_type

PCjrmetka:
        cmp ah,0FDh
        jne PCConvertiblemetka
        mov dx, offset PCConvertible
        jmp print_type

PCConvertiblemetka:
        cmp ah,0FDh
        jne Errormetka
        mov dx, offset PCConvertible
        jmp print_type

Errormetka:
        mov dx,offset Error
        jmp print_type

print_type:
        mov ah,9h
        int 21h

        ret
PC_TYPE endp

OS_TYPE proc near

        mov ah,30h
        int 21h

        mov si,offset dos_version
        add si,13
        push ax
        call BYTE_TO_DEC

        pop ax
        add si,3
        mov al,ah
        call BYTE_TO_DEC

        mov dx,offset dos_version
        mov ah,9h
        int 21h

        mov si,offset oem_serial
        add si,14
        mov al,bh
        call BYTE_TO_DEC

```

```

        mov dx, offset oem_serial
        mov ah,9h
        int 21h

        mov di, offset user_serial
        add di,25
        mov ax,cx
        call WRD_TO_HEX
        mov al,bl
        call BYTE_TO_HEX
        mov [di-2],ax

        mov dx,offset user_serial
        mov ah,9h
        int 21h

        ret
OS_TYPE endp

BEGIN:
        call PC_TYPE
        call OS_TYPE
; Выход в DOS
        xor AL,AL
        mov AH,4Ch
        int 21H
TESTPC ENDS
END START ;конец модуля, START - точка входа

```

ПРИЛОЖЕНИЕ Б

AStack SEGMENT STACK

DW 256 DUP(?)

AStack ENDS

DATA SEGMENT

```
PC db 'Type of PC : PC' , 0dh, 0ah, '$'
PCXT db 'Type of PC : PC/XT' , 0dh, 0ah, '$'
AT db 'Type of PC : AT', 0dh, 0ah, '$'
PS230 db 'Type of PC : PS2 model 30', 0dh, 0ah, '$'
PS250 db 'Type of PC : PS2 model 50 or 60', 0dh, 0ah, '$'
PS280 db 'Type of PC : PS2 model 80', 0dh, 0ah, '$'
PCjr db 'Type of PC : PCjr', 0dh, 0ah, '$'
PCCONVERTIBLE db 'Type of PC: PC Convertible' , 0dh, 0ah, '$'
ERROR db 'Error ', 0dh, 0ah, '$'
DOS_VERSION db 'DOS version: . ', 0dh, 0ah, '$'
OEM_SERIAL db 'OEM number: ', 0dh, 0ah, '$'
USER_SERIAL db 'User serial number:      h', 0dh, 0ah, '$'
```

DATA ENDS

CODE SEGMENT

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

```
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 ;в AL старшая цифра
    pop CX           ;в AH младшая
    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_1
        or AL,30h
        mov [SI],AL
end_1:  pop DX
        pop CX
        ret
BYTE_TO_DEC ENDP

PC_TYPE proc near
    mov ax, 0f000h
    mov es,ax
    mov di, 0ffffh
    mov ah, es:[di]

    ;PC
    cmp ah, 0FFh
    jne PCXTmetka1
    mov dx, offset PC
    jmp print_type

PCXTmetka1:
    cmp ah, 0FEh
    jne PCXTmetka2
    mov dx,offset PCXT
    jmp print_type

PCXTmetka2:
    cmp ah,0FBh
    jne ATmetka
    mov dx, offset PCXT
    jmp print_type

ATmetka:
    cmp ah,0FCh
    jne Model30metka
    mov dx,offset AT
    jmp print_type

Model30metka:
    cmp ah,0FAh
    jne Model50or60metka
    mov dx, offset PS230
    jmp print_type

Model50or60metka:
    cmp ah,0FCh

```

```

        jne Model80metka
        mov dx,offset PS250
        jmp print_type

Model80metka:
        cmp ah,0F8h
        jne PCjrmetka
        mov dx,offset PCjr
        jmp print_type

PCjrmetka:
        cmp ah,0FDh
        jne PCConvertiblemetka
        mov dx, offset PCConvertible
        jmp print_type

PCConvertiblemetka:
        cmp ah,0FDh
        jne Errormetka
        mov dx, offset PCConvertible
        jmp print_type

Errormetka:
        mov dx,offset Error
        jmp print_type

print_type:
        mov ah,9h
        int 21h

        ret
PC_TYPE endp

OS_TYPE proc near

        mov ah,30h
        int 21h

        mov si,offset dos_version
        add si,13
        push ax
        call BYTE_TO_DEC

        pop ax
        add si,3
        mov al,ah
        call BYTE_TO_DEC

        mov dx,offset dos_version
        mov ah,9h
        int 21h

        mov si,offset oem_serial
        add si,14
        mov al,bh
        call BYTE_TO_DEC

        mov dx, offset oem_serial
        mov ah,9h
        int 21h

        mov di, offset user_serial
        add di,25

```

```

        mov ax,cx
        call WRD_TO_HEX
        mov al,bl
        call BYTE_TO_HEX
        mov [di-2],ax

        mov dx,offset user_serial
        mov ah,9h
        int 21h

        ret
OS_TYPE endp

MAIN PROC FAR
        sub ax,ax
        push ax
        mov ax,data
        mov ds,ax

        call PC_TYPE
        call OS_TYPE

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

CODE ENDS
        END MAIN

```