

**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. 9382

\_\_\_\_\_

Кузьмин Д. И.

Преподаватель

\_\_\_\_\_

Ефремов М. А.

Санкт-Петербург

2021

### **Цель работы.**

Исследование различий в структурах 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 PC 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 PC 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.

### **Выполнение работы.**

1. Первым шагом было создание исходного кода для com файла. В нем содержится один сегмент TESTPC. С помощью ASSUME он задается в соответствие с сегментными регистрами CS и DS.

2. Далее следует директива org 100h для отступа в 256 байт.

3. Далее были созданы участки памяти для строк, содержащих информацию о системе.

4. Затем созданы процедуры для загрузки требуемой информации и ее Вывода.

5. Далее был создан исходный код для EXE файла. В нем сегмент кода и данных разделены, а также инициализирован сегмент стека.

Демонстрацию работы программы см. в приложении Б

Исходный код см. в приложении А

## **Контрольные вопросы.**

### **1.1. Сколько сегментов должна содержать COM-программа?**

COM программа должна содержать единственный сегмент, в котором размещаются программный код и данные.

### **1.2 .EXE-программа?**

EXE программа должна иметь минимум один сегмент.

### **1.3. Какие директивы обязательно должны быть в тексте COM-программы?**

ORG 100h для смещения на 256 байт. ASSUME для задания соответствия между единственным сегментом и сегментными регистрами.

### **1.4. Все ли форматы команд можно использовать в COM-программе?**

Нельзя использовать команды, где используются перемещаемые сегменты (relocatable segment), например содержащие оператор seg, потому что нет таблицы настройки. Таблица настройки содержит смещения, которые применяются к адресам при загрузке программы в память.

### **2.1. Какова структура файла COM? С какого адреса располагается код?**

Файл COM состоит из единственного сегмента, хранящего код и данные. Код располагается с адреса 0. Максимальный размер файла – 64 кб.

### **2.2. Какова структура файла плохого EXE? С какого адреса располагается код? Что располагается с адреса 0?**

В плохом EXE код и данные хранятся в одном сегменте. С адреса 0 располагается заголовок, содержащий информацию о файле. Код располагается с адреса 300h.

### **2.3. Какова структура "хорошего" EXE? Чем он отличается от файла "плохого" EXE?**

В "хорошем" EXE код и данные хранятся в разных сегментах. Есть заголовок и таблица настройки, содержащая информацию об адресах, которые изменятся при загрузке программы в память. В «плохом» EXE код и данные не разделяются на сегменты, также «плохой» EXE не учитывает стек и смещается

на 100h. Итоговое смещение будет 300h, в то время как в «хорошем» EXE стек располагается в отдельном сегменте и нет дополнительного смещения 100h. Итоговое смещение – 280h.

3.1. Какой формат загрузки модуля COM? С какого адреса располагается код?

Система заполняет зарезервированные 256 байт для PSP. Затем инициализирует сегментные регистры, которые все указывают на начало PSP. Регистр IP инициализируется 100h, то есть первым байтом после PSP; с этого места располагается код. SP инициализируется FFFEh. Программе выделяется 64кб.

3.2. Что располагается с адреса 0?

Программный префикс PSP, который загружает система.

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

Сегментные регистры имеют одинаковые значения равные 48DD. Они указывают на начало PSP.

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

Стек определяется автоматически и располагается с адреса 0h до FFFEh.

4.1. Как загружается "хороший" EXE? Какие значения имеют сегментные регистры?

К началу программы пристраивается PSP размером 256 байт. Происходит настройка адресов сегментов в соответствии с таблицей настройки. Затем инициализируются сегментные регистры. DS и ES указывают на PSP – 48DD, SS - на начало сегмента стека 48ED, CS - на начало сегмента кода 4907. В указатель IP загружается точка входа в программу, в SP - конец сегмента стека.

4.2. На что указывают регистры DS и ES?

DS и ES указывают на начало PSP.

#### 4.3. Как определяется стек?

Размер стека задается в программе. При загрузке в память В SS загружается начало сегмента. В SP - конец.

#### 4.4. Как определяется точка входа?

Точка входа определяется операндом директивы END.

### **Выводы.**

Были изучены различия в структурах исходных текстов модулей типов .COM и .EXE, структуры файлов загрузочных модулей и способы их загрузки в основную память.

## ПРИЛОЖЕНИЕ А. ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ

C:\os\LR1COM.COM																	h	1252	582
0000000000:	E9	31	02	49	42	4D	20	50	43	20	54	79	70	65	3A	20	é10IBM PC Type:		
0000000010:	50	43	0D	0A	24	49	42	4D	20	50	43	20	54	79	70	65	PC)IBM PC Type		
0000000020:	3A	20	50	43	2F	58	54	0D	0A	24	49	42	4D	20	50	43	: PC/XT)IBM PC		
0000000030:	20	54	79	70	65	3A	20	41	54	0D	0A	24	49	42	4D	20	Type: AT)IBM		
0000000040:	50	43	20	54	79	70	65	3A	20	50	53	32	20	6D	6F	64	PC Type: PS2 mod		
0000000050:	2E	20	33	30	0D	0A	24	49	42	4D	20	50	43	20	54	79	. 30)IBM PC Ty		
0000000060:	70	65	3A	20	50	53	32	20	6D	6F	64	2E	20	33	30	2F	pe: PS2 mod. 30/		
0000000070:	36	30	0D	0A	24	49	42	4D	20	50	43	20	54	79	70	65	60)IBM PC Type		
0000000080:	3A	20	50	53	32	20	6D	6F	64	2E	20	36	30	0D	0A	24	: PS2 mod. 60)S		
0000000090:	49	42	4D	20	50	43	20	54	79	70	65	3A	20	50	43	6A	IBM PC Type: PCj		
00000000A0:	72	0D	0A	24	49	42	4D	20	50	43	20	54	79	70	65	3A	r)IBM PC Type:		
00000000B0:	20	50	43	20	43	6F	6E	76	65	72	74	69	62	6C	65	0D	PC Convertible)		
00000000C0:	0A	24	55	6E	6B	6E	6F	77	6E	20	74	79	70	65	3A	20	)Unknown type:		
00000000D0:	20	0D	0A	24	4D	53	2D	44	4F	53	20	56	65	72	73	69	)MS-DOS Versi		
00000000E0:	6F	6E	3A	20	20	2E	20	20	0D	0A	24	4F	45	4D	20	4E	on: . )OEM N		
00000000F0:	75	6D	62	65	72	3A	20	20	20	0D	0A	24	55	73	65	72	umber: )User		
0000000100:	20	73	65	72	69	61	6C	20	6E	75	6D	62	65	72	3A	20	serial number:		
0000000110:	20	20	20	20	20	20	20	0D	0A	24	24	0F	3C	09	76	02	)\$<ov		
0000000120:	04	07	04	30	C3	B4	09	CD	21	C3	51	8A	E0	E8	EA	FF	♦♦0Ã´oÍ!AQŠàèèÿ		
0000000130:	86	C4	B1	04	D2	E8	E8	E1	FF	59	C3	53	8A	FC	E8	E9	†Ä±♦0èèáÿYÄŠSüèé		
0000000140:	FF	88	25	4F	88	05	4F	8A	C7	E8	DE	FF	88	25	4F	88	ÿ~%0^♦OŠCèbÿ~%0^		
0000000150:	05	5B	C3	51	52	32	E4	33	D2	B9	0A	00	F7	F1	80	CA	†[ÄQR2ä3D¹Š ÷ñ€Ê		
0000000160:	30	88	14	4E	33	D2	3D	0A	00	73	F1	3C	00	74	04	0C	0^ŸN3D=Š sñ< t♦Q		
0000000170:	30	88	04	5A	59	C3	53	BB	00	F0	8E	C3	26	A0	FE	FF	0^♦ZYÄS» ðŽÄ& þÿ		
0000000180:	5B	C3	3C	FF	74	1F	3C	FE	74	22	3C	FC	74	25	3C	FA	[Ä<ÿt▼<þt"<üt%<ú		
0000000190:	74	28	3C	FC	74	2B	3C	F8	74	2E	3C	FD	74	31	3C	F9	t(<üt+<øt.<ýt1<ù		
00000001A0:	74	34	EB	39	90	BA	03	01	E8	7A	FF	C3	BA	15	01	E8	t4ë9▯▼0èzÿÄ°Š0è		
00000001B0:	73	FF	C3	BA	2A	01	E8	6C	FF	C3	BA	3C	01	E8	65	FF	sÿÄ°*0è1ÿÄ°<0èèÿ		

Рис 1. - Файл загрузочного модуля COM.

C:\os\LR1BEXE.EXE																h	1252	1350	
000000000:	4D	5A	46	01	03	00	00	00	00	20	00	00	00	FF	FF	00	00	MZF0♥	ÿÿ
000000010:	00	00	CB	30	00	01	00	00	00	1E	00	00	00	01	00	00	00	Ë0 0 ▲ 0	
000000020:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000030:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000040:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000050:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000060:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000070:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000080:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000090:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000100:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000110:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000120:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000130:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000140:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000150:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000160:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000170:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000180:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000190:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000001A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000001B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000001C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000001D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000001E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000001F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000200:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000210:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000220:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000230:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000240:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000250:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000260:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000270:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000280:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000290:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000002A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000002B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000002C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000002D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000002E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000002F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000300:	E9	31	02	49	42	4D	20	50		43	20	54	79	70	65	3A	20	é10IBM PC Type:	
000000310:	50	43	0D	0A	24	49	42	4D		20	50	43	20	54	79	70	65	PC,8\$IBM PC Type	

Рис. 2 - Файл загрузочного модуля «плохого» EXE



C:\os\LR1GEXE.EXE																h 1252 1237	
0000000000:	4D	5A	D5	00	03	00	01	00	20	00	00	00	FF	FF	00	00	MZÖ ♥ ⊕ ÿÿ
0000000010:	80	00	6D	1A	01	1A	00	1E	00	00	00	01	00	1E	01	€ `m→⊕→ ▲ ⊕ ▲⊕	
0000000020:	1A	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		
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:	49	42	4D	20	50	43	20	54	79	70	65	3A	20	50	43	0D IBM PC Type: PC♪	
0000000290:	0A	24	49	42	4D	20	50	43	20	54	79	70	65	3A	20	50 ☞\$IBM PC Type: P	
00000002A0:	43	2F	58	54	0D	0A	24	49	42	4D	20	50	43	20	54	79 C/XT♪☞\$IBM PC Ty	
00000002B0:	70	65	3A	20	41	54	0D	0A	24	49	42	4D	20	50	43	20 pe: AT♪☞\$IBM PC	
00000002C0:	54	79	70	65	3A	20	50	53	32	20	6D	6F	64	2E	20	33 Type: PS2 mod. 3	
00000002D0:	30	0D	0A	24	49	42	4D	20	50	43	20	54	79	70	65	3A 0♪☞\$IBM PC Type:	
00000002E0:	20	50	53	32	20	6D	6F	64	2E	20	33	30	2F	36	30	0D PS2 mod. 30/60♪	
00000002F0:	0A	24	49	42	4D	20	50	43	20	54	79	70	65	3A	20	50 ☞\$IBM PC Type: P	

Рис. 3 - Файл загрузочного модуля «хорошего» EXE

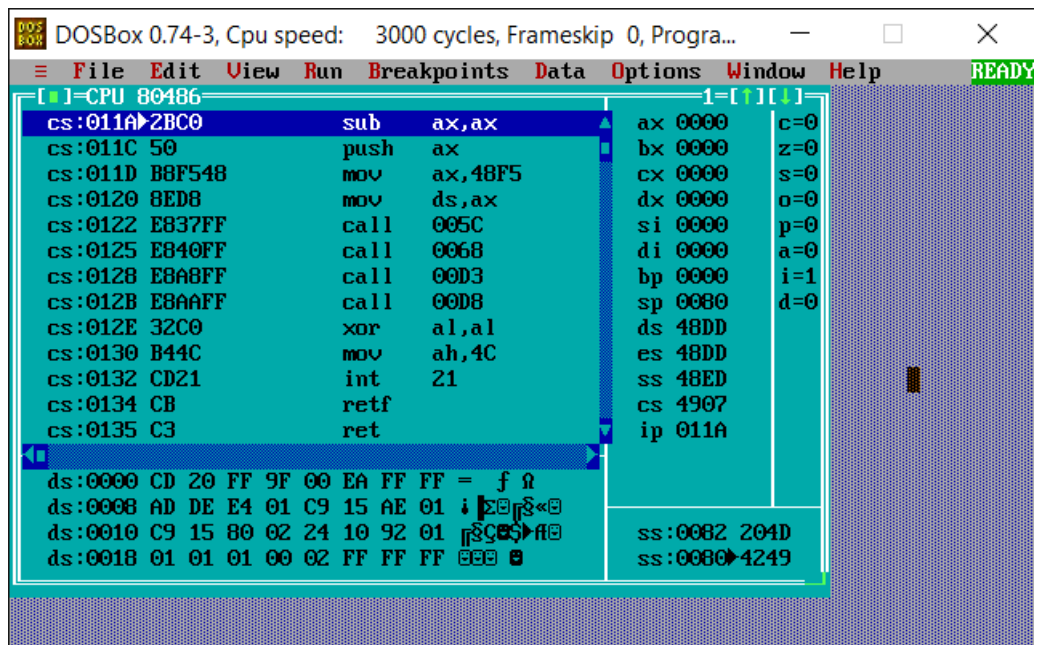


Рис. 4 - Загрузка «хорошего» EXE

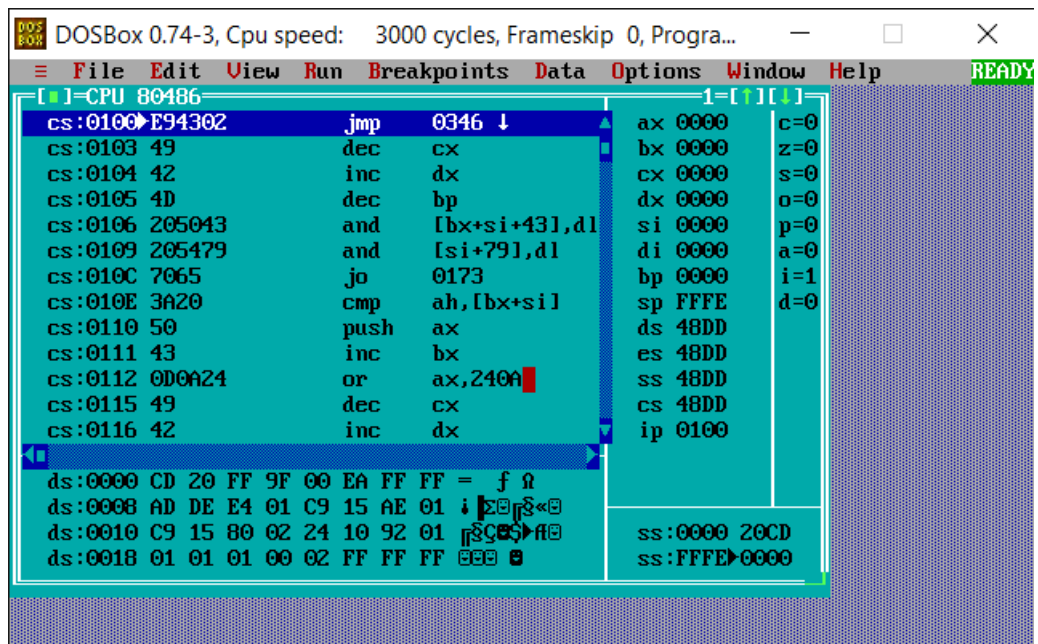


Рис. 5 - Загрузка COM

## ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД

### Файл lr1com.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN
; Данные
type1 db 'IBM PC Type: PC', 0DH, 0AH, '$'
type2 db 'IBM PC Type: PC/XT', 0DH, 0AH, '$'
type3 db 'IBM PC Type: AT', 0DH, 0AH, '$'
type4 db 'IBM PC Type: PS2 mod. 30', 0DH, 0AH, '$'
type5 db 'IBM PC Type: PS2 mod. 30/60', 0DH, 0AH, '$'
type6 db 'IBM PC Type: PS2 mod. 60', 0DH, 0AH, '$'
type7 db 'IBM PC Type: PCjr', 0DH, 0AH, '$'
type8 db 'IBM PC Type: PC Convertible', 0DH, 0AH, '$'
notype db 'Unknown type: ', 0DH, 0AH, '$'
dos_ver_string db 'MS-DOS Version: . ', 0DH, 0AH, '$'
oem_string db 'OEM Number: ', 0DH, 0AH, '$'
user_number_string db 'User serial number: ', 0DH, 0AH, '$'
; 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
;-----
WRITE_MSG PROC near
    mov AH, 09h
    int 21h
    ret
WRITE_MSG 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, в AH - младшая
    pop CX
    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 в 10чную с/с, SI - адрес 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
;-----
GET_PC_TYPE PROC near
;Вывод типа IBM PC
        push BX
        mov BX,0F000h
        mov ES,BX
        mov AL,ES:[0FFFEh]
        pop BX
        ret
GET_PC_TYPE ENDP
;-----
PRINT_PC_TYPE PROC near
        cmp AL, 0FFh
        je type1_found
        cmp AL, 0FEh
        je type2_found
        cmp AL, 0FCh
        je type3_found
        cmp AL, 0FAh
        je type4_found
        cmp AL, 0F8h
        je type5_found
        cmp AL, 0F8h
        je type6_found

```

```

        cmp AL, 0FDh
        je type7_found
        cmp AL, 0F9h
        je type8_found
        jmp notype_found
type1_found:
        mov DX, OFFSET type1
        call WRITE_MSG
        ret
type2_found:
        mov DX, OFFSET type2
        call WRITE_MSG
        ret
type3_found:
        mov DX, OFFSET type3
        call WRITE_MSG
        ret
type4_found:
        mov DX, OFFSET type4
        call WRITE_MSG
        ret
type5_found:
        mov DX, OFFSET type5
        call WRITE_MSG
        ret
type6_found:
        mov DX, OFFSET type6
        call WRITE_MSG
        ret
type7_found:
        mov DX, OFFSET type7
        call WRITE_MSG
        ret
type8_found:
        mov DX, OFFSET type8
        call WRITE_MSG
        ret
notype_found:
        mov DI, offset notype
        call BYTE_TO_HEX
        mov [DI + 13], AX
        mov DX, offset notype
        call WRITE_MSG
        ret
PRINT_PC_TYPE ENDP
;-----
LOAD_INFO PROC near
        mov AH, 30h
        int 21h
        ret
LOAD_INFO ENDP
;-----
PRINT_MS_DOS_VER PROC near
        ;Вывод версий DOS
        mov SI, OFFSET dos_ver_string
        add SI, 16
        call BYTE_TO_DEC

```

```

        add SI, 3
        mov AL, AH
        call BYTE_TO_DEC
        mov DX, OFFSET dos_ver_string
        call WRITE_MSG
        ;Вывод OEM
        mov SI, OFFSET oem_string
        add SI, 13
        mov AL, BH
        call BYTE_TO_DEC
        mov DX, OFFSET oem_string
        call WRITE_MSG
        ;Вывод серийного номера пользователя
        mov DI, OFFSET user_number_string
        add DI, 22
        mov AX, CX
        call WRD_TO_HEX
        mov AL, BL
        call BYTE_TO_HEX
        mov [DI + 4], AX
        mov DX, OFFSET user_number_string
        call WRITE_MSG
        ret
PRINT_MS_DOS_VER ENDP
;-----
; КОД
BEGIN:

        call GET_PC_TYPE
        call PRINT_PC_TYPE
        call LOAD_INFO
        call PRINT_MS_DOS_VER

;Выход в DOS

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

TESTPC ENDS
END START ; Конец модуля, start - точка входа

```

## Файл lrlexe.asm

```

ASTACK      SEGMENT          STACK
            DW 64 DUP(?)
ASTACK      ENDS

DATA SEGMENT
type1 db 'IBM PC Type: PC', 0DH, 0AH, '$'
type2 db 'IBM PC Type: PC/XT', 0DH, 0AH, '$'
type3 db 'IBM PC Type: AT', 0DH, 0AH, '$'
type4 db 'IBM PC Type: PS2 mod. 30', 0DH, 0AH, '$'
type5 db 'IBM PC Type: PS2 mod. 30/60', 0DH, 0AH, '$'
type6 db 'IBM PC Type: PS2 mod. 60', 0DH, 0AH, '$'
type7 db 'IBM PC Type: PCjr', 0DH, 0AH, '$'
type8 db 'IBM PC Type: PC Convertible', 0DH, 0AH, '$'
notype db 'Unknown type: ', 0DH, 0AH, '$'
dos_ver_string db 'MS-DOS Version: . ', 0DH, 0AH, '$'
oem_string db 'OEM Number: ', 0DH, 0AH, '$'
user_number_string db 'User serial number: ', 0DH, 0AH, '$'

```

```

DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:ASTACK
; 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
;-----
WRITE_MSG PROC near
    mov AH, 09h
    int 21h
    ret
WRITE_MSG 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, в AH - младшая
    pop CX
    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 в 10чную с/с, SI - адрес 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_l
        or AL,30h
        mov [SI],AL
end_l:    pop DX
        pop CX
        ret
BYTE_TO_DEC ENDP
;-----
GET_PC_TYPE PROC near
;Вывод типа IBM PC
        push BX
        mov BX,0F000h
        mov ES,BX
        mov AL,ES:[0FFFEh]
        pop BX
        ret
GET_PC_TYPE ENDP
;-----
PRINT_PC_TYPE PROC near
        cmp AL, 0FFh
        je type1_found
        cmp AL, 0FEh
        je type2_found
        cmp AL, 0FCh
        je type3_found
        cmp AL, 0FAh
        je type4_found
        cmp AL, 0FCh
        je type5_found
        cmp AL, 0F8h
        je type6_found
        cmp AL, 0FDh
        je type7_found
        cmp AL, 0F9h
        je type8_found
        jmp notype_found
type1_found:
        mov DX, OFFSET type1
        call WRITE_MSG
        ret
type2_found:
        mov DX, OFFSET type2
        call WRITE_MSG
        ret
type3_found:
        mov DX, OFFSET type3
        call WRITE_MSG
        ret

```



```

type4_found:
    mov DX, OFFSET type4
    call WRITE_MSG
    ret

type5_found:
    mov DX, OFFSET type5
    call WRITE_MSG
    ret

type6_found:
    mov DX, OFFSET type6
    call WRITE_MSG
    ret

type7_found:
    mov DX, OFFSET type7
    call WRITE_MSG
    ret

type8_found:
    mov DX, OFFSET type8
    call WRITE_MSG
    ret

notype_found:
    mov DI, offset notype
    call BYTE_TO_HEX
    mov [DI + 13], AX
    mov DX, offset notype
    call WRITE_MSG
    ret

PRINT_PC_TYPE ENDP
;-----
LOAD_INFO PROC near
    mov AH, 30h
    int 21h
    ret

LOAD_INFO ENDP
;-----
PRINT_DOS_VER PROC near
    ;Вывод версий DOS
    mov SI, OFFSET dos_ver_string
    add SI, 16
    call BYTE_TO_DEC
    add SI, 3
    mov AL, AH
    call BYTE_TO_DEC
    mov DX, OFFSET dos_ver_string
    call WRITE_MSG
    ;Вывод OEM
    mov SI, OFFSET oem_string
    add SI, 13
    mov AL, BH
    call BYTE_TO_DEC
    mov DX, OFFSET oem_string
    call WRITE_MSG
    ;Вывод серийного номера пользователя
    mov DI, OFFSET user_number_string
    add DI, 22
    mov AX, CX
    call WRD_TO_HEX

```

```

        mov AL, BL
        call BYTE_TO_HEX
        mov [DI + 4], AX
        mov DX, OFFSET user_number_string
        call WRITE_MSG
        ret
PRINT_DOS_VER ENDP
;-----
MAIN      PROC FAR
        sub     AX,AX
        push    AX
        mov     AX,DATA
        mov     DS,AX
        call    GET_PC_TYPE
        call    PRINT_PC_TYPE
        call    LOAD_INFO
        call    PRINT_DOS_VER
        xor     AL,AL
        mov     AH,4Ch
        int     21H
        ret
MAIN      ENDP
CODE      ENDS
        END MAIN

```