

**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**

Студентка гр. 0382

Здобнова К.Д.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

### **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 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».

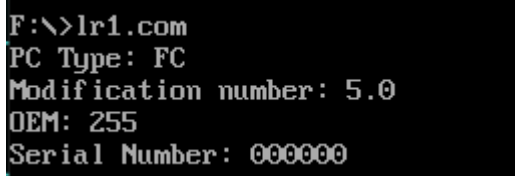
**Шаг 5.** Откройте отладчик TD.EXE и загрузите .COM. Ответьте на контрольные вопросы «Загрузка COM модуля в основную память». Представьте в отчете план загрузки модуля .COM в основную память.

**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.

### **Выполнение работы.**

**Шаг 1.** Пользуясь данным шаблоном, был реализован исходный код .COM модуля, который определяет тип PC и версию системы (см. прил. А). В результате данной реализации получились так называемые “хороший” .COM модуль, а так же “плохой” .EXE модуль. Далее были продемонстрированы выводы данных модулей.



```
F:\>lr1.com
PC Type: FC
Modification number: 5.0
OEM: 255
Serial Number: 000000
```

Рисунок 1. lr1.com

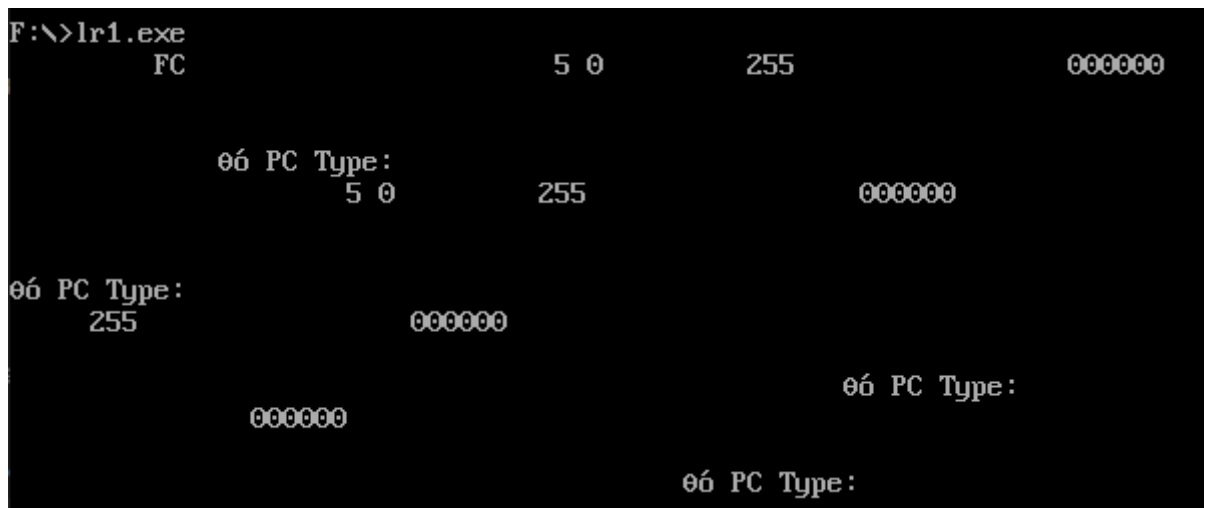


Рисунок 2. lr1.exe

**Шаг 2.** На основе ранее реализованного .COM модуля был реализован исходный код “хорошего” .EXE модуля. Далее был продемонстрирован вывод данного модуля.

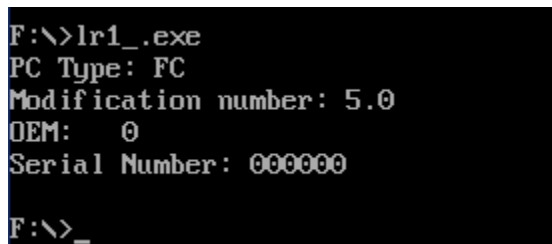


Рисунок 3. lr1 .exe

### Шаг 3. Ответы на контрольные вопросы:

- 1) Сколько сегментов должна содержать COM-программа? – COM-программа должна содержать только один сегмент.
- 2) EXE-программа? – EXE-программа может содержать любое количество.
- 3) Какие директивы должны обязательно быть в тексте COM-программы? – Директива ASSUME – сообщение транслятору о том, какие сегменты использовать в соответствии с какими регистрами, т.к. сегменты сами по себе равноправны. Директива ORG 100h, задающая смещение для всех адресов программы для префикса программного сегмента
- 4) Все ли форматы команд можно использовать в COM-программе? - Адрес сегмента до загрузки неизвестен, следовательно нельзя использовать такие команды, как: mov reg, seg. COM-программа не содержит описание адресов.

Так как адреса зависят от размещения загрузочного модуля в оперативной памяти, следовательно поэтому нельзя использовать команды, дающие доступ к началу сегментов.

#### Шаг 4. Ответы на контрольные вопросы:

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

– .COM файл состоит из одного сегмента и может быть размером не больше 64КБ, содержит команды и данные. Код начинается с 0h

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	E9	BC	01	54	79	70	65	20	6F	66	20	50	43	3A	20	24	Иж.Type of PC: \$
00000010	50	43	0D	0A	24	50	43	2F	58	54	0D	0A	24	41	54	0D	PC..\$PC/XT..\$AT.
00000020	0A	24	50	53	32	20	6D	6F	64	65	6C	20	33	30	0D	0A	.\$PS2 model 30..
00000030	24	50	53	32	20	6D	6F	64	65	6C	20	38	30	0D	0A	24	\$PS2 model 80..\$
00000040	50	43	6A	72	0D	0A	24	50	43	20	43	6F	6E	76	65	72	PCjr..\$PC Conver
00000050	74	69	62	6C	65	0D	0A	24	56	65	72	73	69	6F	6E	20	tible..\$Version
00000060	6F	66	20	4D	53	20	44	4F	53	3A	20	20	2E	20	20	0D	of MS DOS: . .
00000070	0A	24	53	65	72	69	61	6C	20	6E	75	6D	62	65	72	20	.\$Serial number
00000080	6F	66	20	4F	45	4D	3A	20	20	20	24	53	65	72	69	61	of OEM: \$Seria
00000090	6C	20	6E	75	6D	62	65	72	20	6F	66	20	75	73	65	72	l number of user
000000A0	3A	20	20	20	20	20	20	20	20	20	24	0D	0A	24	24	0F	: \$..\$\$.
000000B0	3C	09	76	02	04	07	04	30	C3	51	8A	E0	E8	EF	FF	86	<.v....0ГQЪаипят
000000C0	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53	8A	FC	E8	E9	FF	Д±.ТиижяУТСЪийя
000000D0	88	25	4F	88	05	4F	8A	C7	E8	DE	FF	88	25	4F	88	05	€%O€.OЪзиЮя€%O€.
000000E0	5B	C3	51	52	32	E4	33	D2	B9	0A	00	F7	F1	80	CA	30	[ГQR2л3ТН...чсЪКО
000000F0	88	14	4E	33	D2	3D	0A	00	73	F1	3C	00	74	04	0C	30	€.N3T=..sc<.t..0
00000100	88	04	5A	59	C3	BA	03	01	B4	09	CD	21	B8	00	F0	8E	€.ZYГе..r.H!ë.pђ
00000110	C0	26	A0	FE	FF	3C	FF	74	1C	3C	FE	74	1E	3C	FB	74	А& юя<ят.<ют.<ыт
00000120	1A	3C	FC	74	1C	3C	FA	74	1E	3C	F8	74	20	3C	FD	74	.<ьт.<ьт.<шт <ьт
00000130	22	3C	F9	74	24	BA	10	01	EB	25	90	BA	15	01	EB	1F	"<шт\$е..л%ђе..л.
00000140	90	BA	1D	01	EB	19	90	BA	22	01	EB	13	90	BA	31	01	ђе..л.ђе".л.ђел.
00000150	EB	0D	90	BA	40	01	EB	07	90	BA	47	01	EB	01	90	B4	л.ђе@.л.ђеG.л.ђг
00000160	09	CD	21	C3	B4	30	CD	21	50	BE	58	01	83	C6	13	E8	.H!ГгОН!PsX.ђЖ.и
00000170	70	FF	58	8A	C4	83	C6	03	E8	67	FF	BA	58	01	B4	09	ряХЪДђЖ.игяеX.г.
00000180	CD	21	BE	72	01	83	C6	16	8A	C7	E8	55	FF	BA	72	01	H!sr.ђЖ.ЪзиUяер.
00000190	B4	09	CD	21	BA	AB	01	B4	09	CD	21	BF	8B	01	83	C7	г.H!e«.г.H!i<.ђз
000001A0	1C	8B	C1	E8	24	FF	8A	C3	E8	0E	FF	83	EF	02	89	05	.<Ви\$яЪГи.яђп.%.
000001B0	BA	8B	01	B4	09	CD	21	BA	AB	01	B4	09	CD	21	C3	E8	е<.г.H!e«.г.H!Ги
000001C0	43	FF	E8	9F	FF	32	C0	B4	4C	CD	21						Сяиця2ArLH!

Рисунок 4. Структура «хорошего» COM-файла.

2) Какова структура файла «плохого» EXE? С какого адреса

располагается код? Что располагается с 0 адреса? – Данные и код содержатся в одном сегменте. Код – с адреса 300h. С адреса 0h располагается таблица настроек адресов, маркер MZ и заголовок.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	4D	5A	CB	00	03	00	00	00	20	00	00	00	FF	FF	00	00	MZЛ.....яя..
00000010	00	00	C3	56	00	01	00	00	1E	00	00	00	01	00	00	00	..ГV.....
00000020	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	.....

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000280	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000290	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	.....
000002B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002C0	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	.....
000002E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000300	E9	BC	01	54	79	70	65	20	6F	66	20	50	43	3A	20	24	йj.Type of PC: \$
00000310	50	43	0D	0A	24	50	43	2F	58	54	0D	0A	24	41	54	0D	PC..\$PC/XT..\$AT.
00000320	0A	24	50	53	32	20	6D	6F	64	65	6C	20	33	30	0D	0A	.\$PS2 model 30..
00000330	24	50	53	32	20	6D	6F	64	65	6C	20	38	30	0D	0A	24	\$PS2 model 80..\$
00000340	50	43	6A	72	0D	0A	24	50	43	20	43	6F	6E	76	65	72	PCjr..\$PC Conver
00000350	74	69	62	6C	65	0D	0A	24	56	65	72	73	69	6F	6E	20	tible..\$Version
00000360	6F	66	20	4D	53	20	44	4F	53	3A	20	20	2E	20	20	0D	of MS DOS: . .
00000370	0A	24	53	65	72	69	61	6C	20	6E	75	6D	62	65	72	20	.\$Serial number
00000380	6F	66	20	4F	45	4D	3A	20	20	20	24	53	65	72	69	61	of OEM: \$Seria
00000390	6C	20	6E	75	6D	62	65	72	20	6F	66	20	75	73	65	72	l number of user
000003A0	3A	20	20	20	20	20	20	20	20	20	24	0D	0A	24	24	0F	:
000003B0	3C	09	76	02	04	07	04	30	C3	51	8A	E0	E8	EF	FF	86	<.v....0GQBaипят
000003C0	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53	8A	FC	E8	E9	FF	Д±.ТиижяYTSЪийия
000003D0	88	25	4F	88	05	4F	8A	C7	E8	DE	FF	88	25	4F	88	05	€%O€.OлЪиЮя€%O€.
000003E0	5B	C3	51	52	32	E4	33	D2	B9	0A	00	F7	F1	80	CA	30	[ГQR2д3TН..чсЪК0
000003F0	88	14	4E	33	D2	3D	0A	00	73	F1	3C	00	74	04	0C	30	€.N3T=..sc<.t..0
00000400	88	04	5A	59	C3	BA	03	01	B4	09	CD	21	B8	00	F0	8E	€.ZYTe..r.H!ë.pH
00000410	C0	26	A0	FE	FF	3C	FF	74	1C	3C	FE	74	1E	3C	FB	74	A& юя<ят.<ят.<ят
00000420	1A	3C	FC	74	1C	3C	FA	74	1E	3C	F8	74	20	3C	FD	74	.<ът.<ът.<шт <ст
00000430	22	3C	F9	74	24	BA	10	01	EB	25	90	BA	15	01	EB	1F	"<шт\$e..л%е..л.
00000440	90	BA	1D	01	EB	19	90	BA	22	01	EB	13	90	BA	31	01	еe..л.еe".л.еel.
00000450	EB	0D	90	BA	40	01	EB	07	90	BA	47	01	EB	01	90	B4	л.еe@.л.еeГ.л.ђг'
00000460	09	CD	21	C3	B4	30	CD	21	50	BE	58	01	83	C6	13	E8	.H!Гr0H!PsX.ђЖ.и
00000470	70	FF	58	8A	C4	83	C6	03	E8	67	FF	BA	58	01	B4	09	ряХЪдђЖ.игяеX.г.
00000480	CD	21	BE	72	01	83	C6	16	8A	C7	E8	55	FF	BA	72	01	H!sr.ђЖ.ЪзиUяer.
00000490	B4	09	CD	21	BA	AB	01	B4	09	CD	21	BF	8B	01	83	C7	г.H!e«.г.H!i<.ђЗ
000004A0	1C	8B	C1	E8	24	FF	8A	C3	E8	0E	FF	83	EF	02	89	05	.<Би\$яЪГи.яђп.%.
000004B0	BA	8B	01	B4	09	CD	21	BA	AB	01	B4	09	CD	21	C3	E8	е<.г.H!e«.г.H!Ги
000004C0	43	FF	E8	9F	FF	32	C0	B4	4C	CD	21						Сяиця2ArLH!

Рисунок 5. Структура «плохого» EXE-файла.

3) Какова структура файла «хорошего» EXE? Чем он отличается от «плохого» EXE файла? – В “хорошем” .EXE файле стек и код разделены по сегментам, следовательно файл может быть больше, чем 64КБ. Код не использует директивы ORG 100h, следовательно память не идет под PSP модуль, а код начинается с адреса 300h.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	4D	5A	D6	00	03	00	01	00	20	00	00	00	FF	FF	00	00	MZЦ.....яя..
00000010	00	01	55	23	11	01	1B	00	1E	00	00	00	01	00	15	01	..U#.....
00000020	1B	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	.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000280	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000290	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	.....
000002B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002C0	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	.....
000002E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000300	54	79	70	65	20	6F	66	20	50	43	3A	20	24	50	43	0D	Type of PC: \$PC.
00000310	0A	24	50	43	2F	58	54	0D	0A	24	41	54	0D	0A	24	50	.\$PC/\$XT..\$AT..\$P
00000320	53	32	20	6D	6F	64	65	6C	20	33	30	0D	0A	24	50	53	S2 model 30..\$PS
00000330	32	20	6D	6F	64	65	6C	20	38	30	0D	0A	24	50	43	6A	2 model 80..\$PCj
00000340	72	0D	0A	24	50	43	20	43	6F	6E	76	65	72	74	69	62	r..\$PC Convertib
00000350	6C	65	0D	0A	24	56	65	72	73	69	6F	6E	20	6F	66	20	le..\$Version of
00000360	4D	53	20	44	4F	53	3A	20	20	2E	20	20	0D	0A	24	53	MS DOS: . .\$.S
00000370	65	72	69	61	6C	20	6E	75	6D	62	65	72	20	6F	66	20	erial number of
00000380	4F	45	4D	3A	20	20	20	24	53	65	72	69	61	6C	20	6E	OEM: \$Serial n
00000390	75	6D	62	65	72	20	6F	66	20	75	73	65	72	3A	20	20	umber of user:
000003A0	20	20	20	20	20	20	20	24	0D	0A	24	00	00	00	00	00	\$.\$. . . . .
000003B0	24	0F	3C	09	76	02	04	07	04	30	C3	51	8A	E0	E8	EF	\$.<.v....0GQБаип
000003C0	FF	86	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53	8A	FC	E8	ятДт.ТиижяУГЗЪи
000003D0	E9	FF	88	25	4F	88	05	4F	8A	C7	E8	DE	FF	88	25	4F	йя€%O€,OЪзиЮя€%O
000003E0	88	05	5B	C3	51	52	32	E4	33	D2	B9	0A	00	F7	F1	80	€. [ГQR2д3ТН. . .чсЪ
000003F0	CA	30	88	14	4E	33	D2	3D	0A	00	73	F1	3C	00	74	04	K0€.N3T=. .sc<.t.
00000400	0C	30	88	04	5A	59	C3	BA	00	00	B4	09	CD	21	B8	00	.0€,ZYГe..r.H!€.
00000410	F0	8E	C0	26	A0	FE	FF	3C	FF	74	1C	3C	FE	74	1E	3C	рhA& ня<ят.<ют.<
00000420	FB	74	1A	3C	FC	74	1C	3C	FA	74	1E	3C	F8	74	20	3C	ыт.<ът.<ът.<шт <
00000430	FD	74	22	3C	F9	74	24	BA	0D	00	EB	25	90	BA	12	00	ст"<шт\$e..л%ђе..
00000440	EB	1F	90	BA	1A	00	EB	19	90	BA	1F	00	EB	13	90	BA	л.ђе..л.ђе..л.ђе
00000450	2E	00	EB	0D	90	BA	3D	00	EB	07	90	BA	44	00	EB	01	..л.ђе=.л.ђеD.л.
00000460	09	B4	09	CD	21	C3	B4	30	CD	21	50	BE	55	00	83	C6	ђгг.Н!ГгОН!PsU.ђЖ
00000470	13	E8	70	FF	58	8A	C4	83	C6	03	E8	67	FF	BA	55	00	.иpяXЪДђЖ.иgяeU.
00000480	B4	09	CD	21	BE	6F	00	83	C6	16	8A	C7	E8	55	FF	BA	г.Н!so.ђЖ.ЪзиUяe
00000490	6F	00	B4	09	CD	21	BA	A8	00	B4	09	CD	21	BF	88	00	o.г.Н!eђ.г.Н!ie.
000004A0	83	C7	1C	8B	C1	E8	24	FF	8A	C3	E8	0E	FF	83	EF	02	ђЗ.<Ви\$яЪГи.яђп.
000004B0	89	05	BA	88	00	B4	09	CD	21	BA	A8	00	B4	09	CD	21	ћ.e€.г.Н!eђ.г.Н!
000004C0	C3	2B	C0	50	B8	10	00	8E	D8	E8	3B	FF	E8	97	FF	32	Г+AP€. .Ъши;яи-я2
000004D0	C0	B4	4C	CD	21	C3											ArLH!Г

Рисунок 6. Структура "хорошего" EXE-модуля.

## Шаг 5. Ответы на контрольные вопросы:

1) Какой формат загрузки модуля COM? С какого адреса располагается код? – Сначала определяется сегментный адрес участка ОП, способного вместить загрузку программы, затем создается блок памяти для PSP и программы, COM-файл считывается помещается в память с 100h. После сегментные регистры устанавливаются на начало PSP. SP устанавливается на конец PSP, 0000h помещается в стек, в IP записывается 100h. Код располагается с адреса 100h.

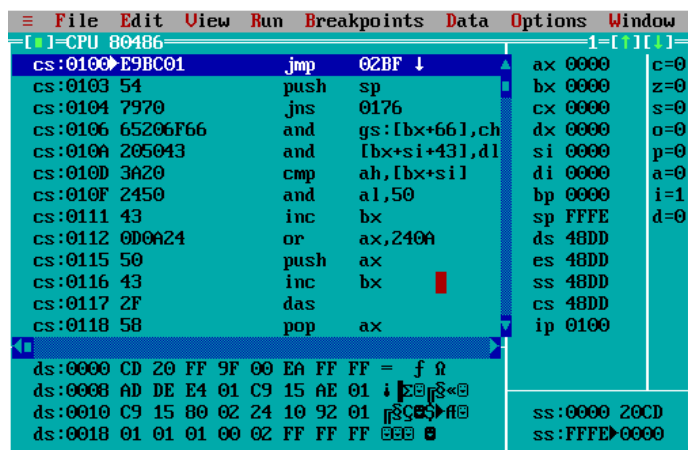




Рисунок 7. COM-модуль в начале загрузки в память.

- 2) Что располагается с адреса 0? – Префикс программного сегмента.
- 3) Какие значения имеют сегментные регистры? На какие области памяти они указывают? – Сегментные регистры имеют значения, соответствующие сегменту, в который был помещен модуль. 48DD(PSP), так как один и тот же сегмент памяти.
- 4) Как определяется стек? Какую область памяти он занимает? Какие адреса? – Указатель стека в конце сегмента, занимает неиспользованную память, адреса меняются от FFFh к 0000h.

#### Шаг 6. Ответы на контрольные вопросы:

- 1) Как загружается «хороший» EXE? Какие значения имеют сегментные регистры? – Создается префикс программного сегмента, в начальный сегмент записывается загрузочный модуль, таблица настройки – в рабочую память. Каждый сегмент прибавляет сегментный адрес начального сегмента данных. Далее определяются значения сегментных регистров. DS и ES – начало PSP(48DD), CS – Начало команд(4932). SS – начало стека(48ED).
- 2) На что указывают регистры DS и ES? - DS и ES – начало PSP(48DD).

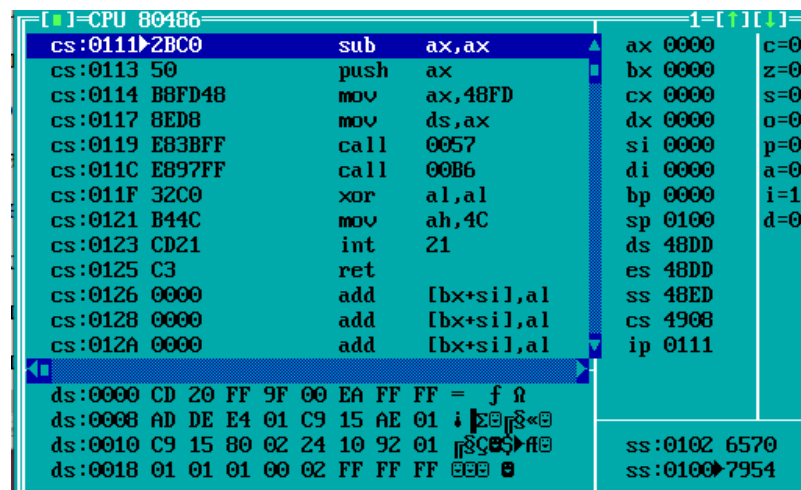


Рисунок 8. EXE-модуль в начале загрузки в память.

- 3) Как определяется стек? - В отличие от COM стек определяется при объявлении его сегмента, выделяется указанная память.



4) Как определяется точка входа? – директивой END с адресом, с которого начинается выполнение программы.

### **Вывод**

В ходе выполнения лабораторной работы была написана программа для определения типа и версии РС, изучены различия 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: lr1.asm

```
; Шаблон текста программы на ассемблере для модуля типа .COM
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN
; ДАННЫЕ
PC_Type          db    'PC Type: ', 0dh, 0ah, '$'
Mod_numb         db    'Modification number: . ', 0dh, 0ah, '$'
OEM              db    'OEM: ', 0dh, 0ah, '$'
S_numb           db    'Serial Number: ', 0dh, 0ah, '$'
; 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
;-----
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 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_l
    or AL,30h
    mov [SI],AL
end_l: pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP
;-----
; КОД
BEGIN:
;PC_Type
    push es
    push bx
    push ax
    mov bx, 0F000h
    mov es, bx
    mov ax, es:[0FFFEh]
    mov ah, al
    call BYTE_TO_HEX
    lea bx, PC_Type
    mov [bx+9], ax; смещение по колву символов, записали в PC_Type
по адресу
    pop ax
    pop bx
    pop es

    mov ah, 30h; Воспользуемся функцией получения информации о MS
DOS
    int 21h

;Mod_numb
    push ax
    push si
    lea si, Mod_numb; в si адрес Mod_numb
    add si, 21; Сместимся на 21 символ
    call BYTE_TO_DEC; al - Basic version number
    add si, 3; Еще на три
    mov al, ah
    call BYTE_TO_DEC; al - Modification number
    pop si
    pop ax

```

```

;OEM
    mov al, bh
    lea si, OEM
    add si, 7
    call BYTE_TO_DEC; al - OEM number

;S_numb
    mov al, bl
    call BYTE_TO_HEX; al - 24b number
    lea di, S_numb
    add di, 15
    mov [di], ax
    mov ax, cx
    lea di, S_numb
    add di, 20
    call WRD_TO_HEX

;Output
    mov AH, 09h
    lea DX, PC_Type
    int 21h
    lea DX, Mod_numb
    int 21h
    lea DX, OEM
    int 21h
    lea DX, S_numb
    int 21h

; Выход в DOS
    xor AL, AL
    mov AH, 4Ch
    int 21h
TESTPC ENDS
    END START ;конец модуля, START - точка входа

```

**Название файла: lr1\_.asm**

```

AStack SEGMENT STACK
AStack ENDS

```

```

DATA SEGMENT
PC_Type          db    'PC Type:  ', 0dh, 0ah, '$'
Mod_numb         db    'Modification number:  . ', 0dh, 0ah, '$'
OEM              db    'OEM:  ', 0dh, 0ah, '$'
S_numb           db    'Serial Number:      ', 0dh, 0ah, '$'
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

```

```

;-----
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 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
main:

```

```

        push ds
        sub ax, ax
        push ax
        mov ax, DATA
        mov ds, ax
;PC_Type
        push es
        push bx
        push ax
        mov bx, 0F000h
        mov es, bx
        mov ax, es:[0FFFEh]
        mov ah, al
        call BYTE_TO_HEX
        lea bx, PC_Type
        mov [bx+9], ax; смещение по колву символов, записали в PC_Type
по адресу
        pop ax
        pop bx
        pop es

        mov ah, 30h; Воспользуемся функцией получения информации о MS
DOS
        int 21h

;Mod_numb
        push ax
        push si
        lea si, Mod_numb; в si адрес Mod_numb
        add si, 21; Сместимся на 21 символ
        call BYTE_TO_DEC; al - Basic version number
        add si, 3; Еще на три
        mov al, ah
        call BYTE_TO_DEC; al - Modification number
        pop si
        pop ax

;OEM
        mov al, bh
        lea si, OEM
        add si, 7
        call BYTE_TO_DEC; al - OEM number

;S_numb
        mov al, bl
        call BYTE_TO_HEX; al - 24b number
        lea di, S_numb
        add di, 15
        mov [di], ax
        mov ax, cx
        lea di, S_numb
        add di, 20
        call WRD_TO_HEX

;Output
        mov AH, 09h
        lea DX, PC_Type
        int 21h

```

```
        lea DX, Mod_numb
        int 21h
        lea DX, OEM
        int 21h
        lea DX, S_numb
        int 21h

; Выход в DOS
        xor AL,AL
        mov AH,4Ch
        int 21H
CODE ENDS
        END main
```