

**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. 7381

\_\_\_\_\_

Лукашев Р.С.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2019

## **Цель работы.**

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

## **Постановка задачи.**

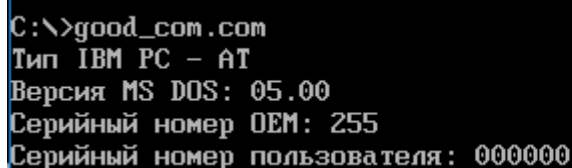
Разработать на языке Assembler программу, определяющую тип IBM PC, версию MS-DOS (в формате XX.YY, где XX – номер основной версии операционной системы, а YY – номер модификации), серийный номер OEM (Original Equipment Manufacturer) и 24-битовый серийный номер пользователя и выводящую эти данные на экран пользователя.

Далее необходимо отладить полученный исходный модуль и получить «хороший» .COM модуль, после чего построить «плохой» .EXE, полученный из исходного текста для .COM модуля.

Затем нужно написать текст исходного .EXE модуля, который выполняет те же функции, что и модуль .COM, сравнить исходные тексты для .COM и .EXE модулей, ответить на контрольные вопросы.

## **Ход работы.**

- 1) Был написан текст исходного .COM модуля. Результатом отладки исходного модуля был “хороший” .COM модуль. После этого был построен “плохой” .EXE модуль, полученный из исходного текста для .COM модуля. Результат работы хорошего .COM и плохого .EXE модуля представлены на рис. 1 и рис. 2 соответственно.



```
C:\>good_com.com
Тип IBM PC - AT
Версия MS DOS: 05.00
Серийный номер OEM: 255
Серийный номер пользователя: 000000
```

Рисунок 1 – результат работы “хорошего” .COM модуля

```
C:\>bad_exe.exe

                                     Тип IBM PC – PC
                                     Тип IBM PC – PC
05.00
  Тип IBM PC – PC
255M P0000002 модель 30
0000002 модель 30
```

Рисунок 2 – результат работы “плохого” .EXE модуля

- 2) Был написан текст исходного .EXE модуля, выполняющего те же функции, что и модуль в Шаге 1. Результатом отладки этого модуля был “хороший” .EXE модуль. Результат работы “хорошего” .EXE представлен на рисунке 3.

```
C:\>good_exe.exe
Тип IBM PC – AT
Версия MS DOS: 05.00
Серийный номер OEM: 0
Серийный номер пользователя: 000000
```

Рисунок 3 – результат работы “хорошего” .EXE модуля

## Ответы на контрольные вопросы.

### *Отличия исходных текстов .COM и .EXE программ*

- 1) Сколько сегментов может содержать COM программ?
  - Один сегмент.
- 2) EXE-программ?
  - EXE-программа может содержать любое количество сегментов, но не менее одного. Обычно по мере необходимости выделяют сегменты стека, данных и кода, при этом их может быть несколько, а обязательным является только сегмент кода.
- 3) Какие директивы должны обязательно быть в тексте COM-программы?
  - Так как адресация в COM-программах начинается со смещения 100h (256 байт) от начала PSP, то в программе должна присутствовать директива `ORG 100h` для резервации места. Также необходима директива `END` для

обозначения точки входа в программу. Пример простейшей .COM программы, которую можно отладить и собрать:

```
CODE    SEGMENT
    org 100h
begin:
    xor  al, al
    mov  ah, 4ch
    int  21h
CODE    ENDS
    END  begin
```

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

- Нет, не все. COM-программа подразумевает наличие только одного сегмента, поэтому в COM-программах запрещены far переходы. Также нельзя использовать команды, связанные с адресом сегмента. В EXE-программах существует таблица настроек, называемая Relocation Table (таблица разметки), отсутствующая в COM-программах. Во время запуска EXE-программы загрузчик определяет адреса сегментов, основываясь на информации о местоположении в файле полей адресов из Relocation Table. А так как в COM-программах такая таблица отсутствует, то такие команды, как “mov ax, DATA”, “mov ax, CODE”, “mov ax, seg CODE”, где DATA и CODE – имена сегментов, и другие, недопустимы. Также нельзя использовать команды размером больше размера сегмента, а также команды, работающие с 64 – битными регистрами.

*Отличия форматов файлов .COM и .EXE модулей*

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

- COM-файл состоит из одного сегмента и содержит команды и данные. Команды и данные начинаются с адреса 0h, но при загрузке модуля устанавливается смещение 100h. 16-ричное представление модуля представлено на рис. 4.

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Dump
00000000	E9	99	02	92	A8	AF	20	49	42	4D	20	50	43	20	2D	20	й™.'ëï IBM PC -
00000010	50	43	0D	0A	24	92	A8	AF	20	49	42	4D	20	50	43	20	PC..\$'ëï IBM PC
00000020	2D	20	50	43	2F	58	54	0D	0A	24	92	A8	AF	20	49	42	- PC/XT..\$'ëï IB
00000030	4D	20	50	43	20	2D	20	41	54	0D	0A	24	92	A8	AF	20	M PC - AT..\$'ëï
00000040	49	42	4D	20	50	43	20	2D	20	50	53	32	20	AC	AE	A4	IBM PC - PS2 →@»
00000050	A5	AB	EC	20	33	30	0D	0A	24	92	A8	AF	20	49	42	4D	Г«м 30..\$'ëï IBM
00000060	20	50	43	20	2D	20	50	53	32	20	AC	AE	A4	A5	AB	EC	PC - PS2 →@»Г«м
00000070	20	35	30	20	A8	AB	A8	20	36	30	0D	0A	24	92	A8	AF	50 ë«ë 60..\$'ëï
00000080	20	49	42	4D	20	50	43	20	2D	20	50	53	32	20	AC	AE	IBM PC - PS2 →@
00000090	A4	A5	AB	EC	20	38	30	0D	0A	24	92	A8	AF	20	49	42	»Г«м 80..\$'ëï IB

Рисунок 4 – структура файла COM

2) Какова структура файла “плохого” EXE? С какого адреса располагается код?

Что располагается с адреса 0?

- В “плохом” EXE-модуле данные и команды содержатся в одном сегменте, сначала данные, затем команды. Сегмента стека нет. С адреса 0h идет таблица разметки (Relocation table), а код располагается с адреса 300h (После таблицы разметки еще выделено командой org 100h 256 байт пустой памяти, поэтому на самом деле, код располагается с адреса 200h). 16-ричное представление модуля представлено на рис. 5.

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Dump
00000000	4D	5A	D6	01	03	00	00	00	20	00	00	00	FF	FF	00	00	MZЦ.....яя..
00000010	00	00	00	00	00	01	00	00	3E	00	00	00	01	00	FB	50	.....>.....ыр
00000020	6A	72	00	00	00	00	00	00	00	00	00	00	00	00	00	00	j r.....
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	.....
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000300	E9	99	02	92	A8	AF	20	49	42	4D	20	50	43	20	2D	20	й™.'ëï IBM PC -
00000310	50	43	0D	0A	24	92	A8	AF	20	49	42	4D	20	50	43	20	PC..\$'ëï IBM PC
00000320	2D	20	50	43	2F	58	54	0D	0A	24	92	A8	AF	20	49	42	- PC/XT..\$'ëï IB
00000330	4D	20	50	43	20	2D	20	41	54	0D	0A	24	92	A8	AF	20	M PC - AT..\$'ëï

Рисунок 5 – структура “плохого” EXE

3) Какова структура файла “хорошего” EXE-модуля? Чем он отличается от “плохого” EXE?

- В “хорошем” файле EXE содержится информация для загрузчика, сегменты стека, данных и кода расположены отдельно, сначала идет сегмент кода, затем данных, а затем стека. В отличие от “плохого” EXE-модуля, код явно располагается с адреса 200h. 16-ричное представление модуля представлено на рис. 6.

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Dump
00000000	4D	5A	E4	00	03	00	01	00	20	00	31	00	FF	FF	2F	00	МЗд..... .1.яя/.
00000010	00	03	00	00	00	00	00	00	3E	00	00	00	01	00	FB	50	.....>.....ыР
00000020	6A	72	00	00	00	00	00	00	00	00	00	00	00	00	00	00	jr.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	49	01	.....I.
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000200	E9	45	01	24	0F	3C	09	76	02	04	07	04	30	C3	51	8A	йЕ.\$.<.v....0ГQЪ
00000210	E0	E8	EF	FF	86	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53	аипя†Д±.ТиижяУГS
00000220	8A	FC	E8	E9	FF	88	25	4F	88	05	4F	8A	C7	E8	DE	FF	льийя€%€€.ользиюя
00000230	88	25	4F	88	05	5B	C3	51	52	32	E4	33	D2	B9	0A	00	€%€€. [ГQR2д3Т№..
00000240	F7	F1	80	CA	30	88	14	4E	33	D2	3D	0A	00	73	F1	3C	чсЪK0€.N3T=..sc<
00000250	00	74	04	0C	30	88	04	5A	59	C3	B4	09	CD	21	C3	50	.t...0€.ZYTr.n!ГР
00000260	52	1E	B8	00	F0	8E	D8	A1	FE	FF	1F	3C	FF	74	35	3C	R.ё.рЪШЮя.<ят5<
00000270	FE	74	37	3C	FB	74	33	3C	FC	74	35	3C	FA	74	37	3C	ют7<ът3<ът5<ът7<
00000280	F8	74	3F	3C	FC	74	35	3C	FD	74	3D	3C	F9	74	3F	BF	шт?<ът5<эт=<шт?i
00000290	2D	01	E8	79	FF	89	05	BA	C9	00	B4	09	CD	21	BA	2D	-.иуя%.ей.г.н!е-
000002A0	01	EB	31	90	BA	00	00	EB	2B	90	BA	12	00	EB	25	90	.л1.е..л+.е..л%.
000002B0	BA	27	00	EB	1F	90	BA	39	00	EB	19	90	BA	56	00	EB	е'.л..е9.л..еV.л
00000380	FE	B0	00	B4	4C	CD	21	00	00	00	00	00	00	00	00	00	ю°.гЛH!.....
00000390	92	A8	AF	20	49	42	4D	20	50	43	20	2D	20	50	43	0D	'Ёİ IBM PC - PC.

Рисунок 6 – структура “хорошего” EXE

### Загрузка COM модуля в основную память

- 1) Какой формат загрузки модуля COM? С какого адреса располагается код?
  - Отладчик с загруженной программой представлен на рис. 7. При загрузке программы в память, DOS определяет сегментный адрес участка памяти, у которого достаточно места для загрузки программы, после чего создается два блока памяти: для переменных среды, и для PSP и программы. Затем заполняются поля PSP в соответствии с характеристиками программы. После этого устанавливается адрес области Disk Transfer Data на вторую половину PSP. Анализируются параметры запуска программы на предмет наличия в первых двух параметрах идентификаторов дисковых устройств. По результатам

анализа устанавливается содержимое регистра AX при входе в программу. Если первый или второй параметры не содержат правильного идентификатора дискового устройства, то соответственно в регистры AL и AH записывается значение FF.

После этого происходит запись COM файла в эту память по адресу PSP:0100h. Затем сегментные регистры устанавливаются на начало PSP, регистр SP устанавливается на конец сегмента PSP, вся область памяти после PSP распределяется программе, и в стек записывается слово 0000h. После этого в регистр IP записывается значение 100h с помощью команды JMP 100h. Код располагается с адреса 100h.

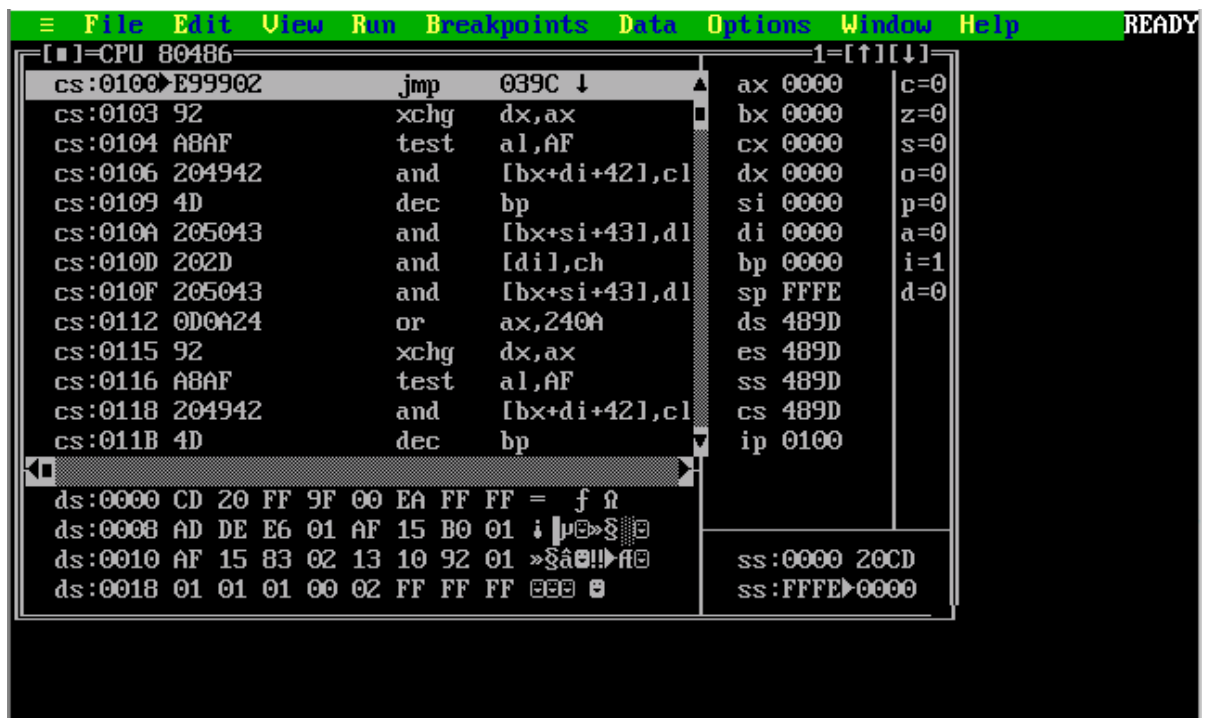


Рисунок 7 – загрузка хорошего модуля COM в память

- 2) Что располагается с адреса 0?
  - PSP
- 3) Какие значения имеют сегментные регистры? На какие области памяти они указывают?

- В нашем случае сегментные регистры имеют значение 489Dh. Они указывают на начало PSP.
- 4) Как определяется стек? Какую область памяти он занимает? Какие адреса?
- Стек определяется автоматически, в регистре SP находится относительный адрес конца программного сегмента, а в SS – адрес начала программного сегмента, т.е. программа занимает начало сегмента, а стек занимает конец сегмента. Адреса расположены в диапазоне 0000h – FFFEh.

#### *Загрузка “хорошего” EXE-модуля в основную память*

- 1) Как загружается «хороший» EXE? Какие значения имеют сегментные регистры?
- Отладчик с загруженной программой представлен на рис. 7. Загрузка файла происходит аналогично COM модулю, после чего во внутренний буфер DOS считывается форматированная часть заголовка файла, определяется размер загрузочного модуля, определяется смещение начала загрузочного модуля, вычисляется сегментный адрес для загрузки стартового сегмента, загрузочный модуль считывается в память по адресу стартового сегмента. Затем сканируются элементы таблицы перемещений, и для каждого элемента таблицы: считывается содержимое элемента таблицы как два двухбайтных слова (OFFSET, SEGMENT), вычисляется сегментный адрес ссылки перемещения REL\_SEG, выбирается слово по адресу REL\_SEG: OFFSET, к этому слову прибавляется значение стартового сегмента, затем эта сумма записывается обратно по тому же адресу. Далее заказывается память для программы исходя из значений в заголовке, инициализируются регистры, и программа запускается на выполнение.





Рисунок 8 – загрузка хорошего EXE-модуля в память

2) На что указывают регистры DS и ES?

- После загрузки регистры DS и ES указывают на начало PSP.

3) Как определяется стек?

- В SS записывается значение стартовый сегмент + смещение сегмента стека, смещение берется из заголовка, а в SP записывается значение смещения указателя, также из заголовка.

4) Как определяется точка входа?

- Для запуска программы в CS записывается значение стартовый сегмент + смещение сегмента кода, находящееся в заголовке, а в IP – отдельное значение из заголовка. Точка входа в коде определяется директивой END.

## Выводы.

В ходе выполнения работы были исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.