

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

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

Студент гр. 8382

Преподаватель

---

Торосян Т.А.

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

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

### **Необходимые сведения для составления программы.**

Байт типа IBM PC хранится по адресу 0F000:0FFFE, в предпоследнем байте ROM BIOS. Соответствие кода и типа в таблице:

<b>PC</b>	<b>FF</b>
<b>PC/XT</b>	<b>FE,FB</b>
<b>AT</b>	<b>FC</b>
<b>PS2 модель 30</b>	<b>FA</b>
<b>PS2 модель 50 или 60</b>	<b>FC</b>
<b>PS2 модель 80</b>	<b>F8</b>
<b>PCjr</b>	<b>FD</b>
<b>PC Convertible</b>	<b>F9</b>

Для определения версии MS DOS используется функция 30H прерывания 21H. Входным параметром является номер функции в AH:

**MOV AH, 30h**

**INT 21h**

Выходными параметрами являются:

**AL** – номер основной версии. Если 0, то <2.0;

**AH** – номер модификации;

**BH** – серийный номер OEM (Original Equipment Manufacturer);

**BL:CH** – 24-битовый серийный номер пользователя.

### **Постановка задачи.**

Требуется реализовать текст исходного .COM модуля, который определяет тип PC и версию системы. 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 (Original Equipment Manufacturer) и серийным номером пользователя. Полученные строки выводятся на экран.

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

Затем нужно написать текст «хорошего» .EXE модуля, который выполняет те же функции, что и модуль .COM, далее его построить, отладить и сравнить исходные тексты для .COM и .EXE модулей.

### **Процедуры используемые в программе.**

TETR\_TO\_HEX – Используется для перевода половины байта в шестнадцатеричную систему счисления.

BYTE\_TO\_HEX – Используется для перевода байта регистра AL в шестнадцатеричную систему счисления, помещая результат в AH.

WRD\_TO\_HEX – Используется для перевода двух байт регистра AX в шестнадцатеричную систему счисления, помещая результат в регистр DI.

BYTE\_TO\_DEC – Используется для перевода байта регистра AL в десятичную систему счисления, помещая результат в SI.

TYPE\_IBM\_PC – Определяет тип IBM PC.

VERS\_DOS – Определяет версию MS DOS.

OEM\_DOS – Определяет серийный номер OEM.

USER\_DOS - Определяет серийный номер пользователя.

PRINT – Вывод на экран.

### Структуры данных.

Таблица 1 – Структуры данных

Название поля данных	Тип	Назначение
type	db	Тип IBM PC
PC	db	PC
PC XT	db	PC/XT
AT	db	AT
PS2 30	db	PS2 model 30
PS2 50 60	db	PS2 model 50 or 60
PS2 80	db	PS2 model 80
PCjr	db	PCjr
PC Conv	db	PC Convertible
ver	db	Version number MSDOS
oem	db	Serial number OEM
user	db	Serial user s number

### Ход работы.

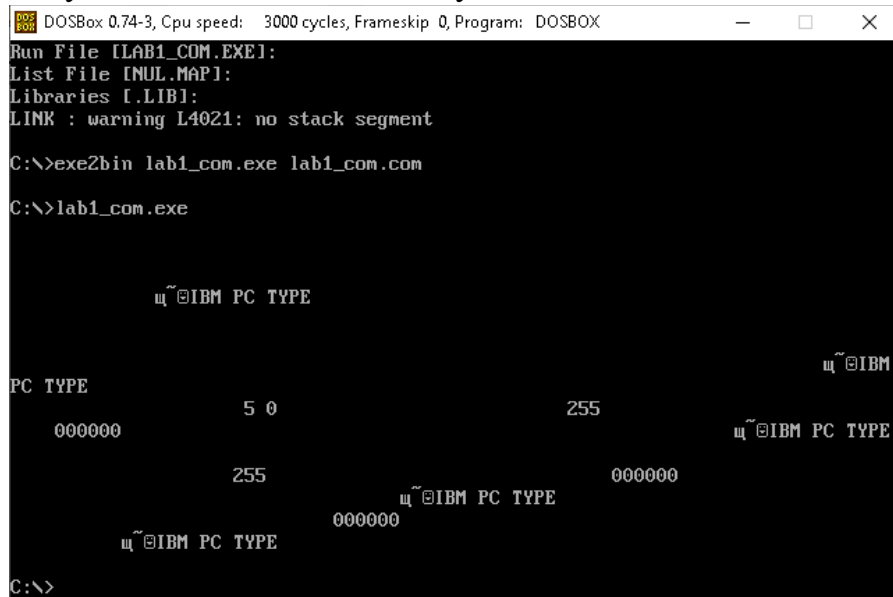
#### Шаг 1. Запуск модулей из COM-исходника

Запуск «хорошего» .COM модуля.

```
C:\>lab1_com.com
IBM PC TYPE
PC/XT
MSDOS version number:5.0
OEM serial number: 255
User s serial number: 000000
C:\>
```

Рисунок 1 – «Хороший» .COM модуль

### Запуск «плохого» .EXE модуля.



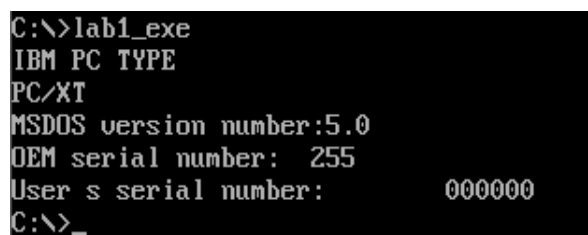
```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Run File [LAB1_COM.EXE]:
List File [NUL.MAP]:
Libraries [LIB1:
LINK : warning L4021: no stack segment

C:\>exe2bin lab1_com.exe lab1_com.com
C:\>lab1_com.exe

IBM PC TYPE
PC TYPE
5 0
255
000000
IBM PC TYPE
PC TYPE
255
000000
IBM PC TYPE
000000
IBM PC TYPE
C:\>
```

Рисунок 2 – «Плохой» .EXE модуль

### Шаг 2. Запуск «хорошего» EXE-модуля.



```
C:\>lab1_exe
IBM PC TYPE
PC/XT
MSDOS version number:5.0
OEM serial number: 255
User s serial number: 000000
C:\>_
```

Рисунок 3 – «Хороший» .EXE модуль

**Шаг 3.** Ответы на контрольные вопросы. Отличия исходных текстов COM и EXE программ.

1) **Сколько сегментов должна содержать COM-программа?**

Один сегмент.

2) **EXE программа?**

EXE программа может содержать больше одного сегмента.

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

Директива `ORG 100h` (смещение `100h`), так как при загрузке COM-файла в память DOS занимает первые 256 байт (`100h`) блоком данных PSP и

4) Все ли форматы команд можно использовать в СОМпрограмме?

#### Шаг 4. Содержимое модуля в шестнадцатеричном виде.

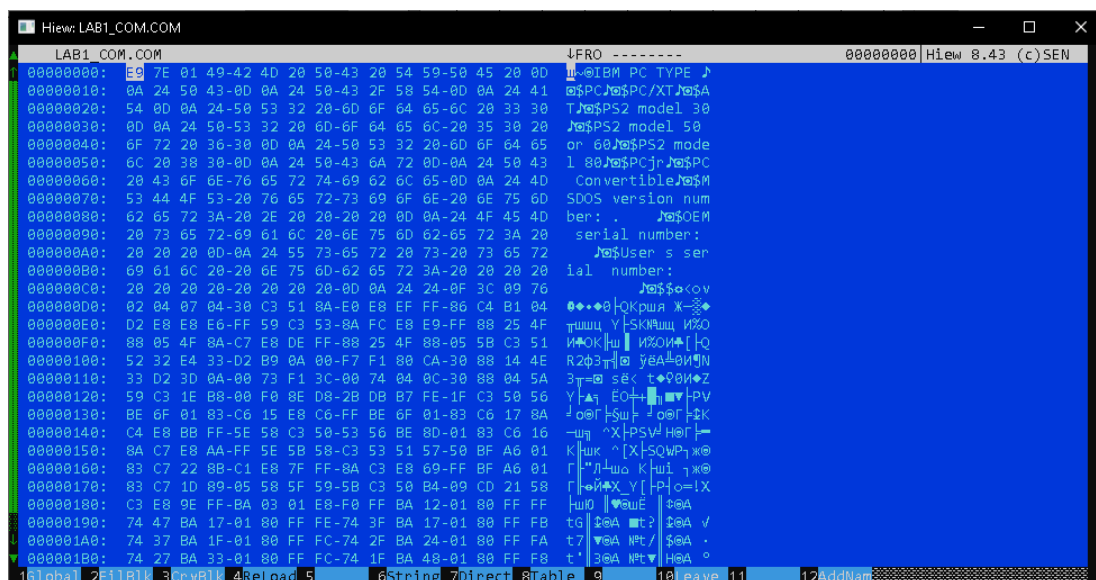


Рисунок 4 - .COM модуль в шестнадцатеричном виде

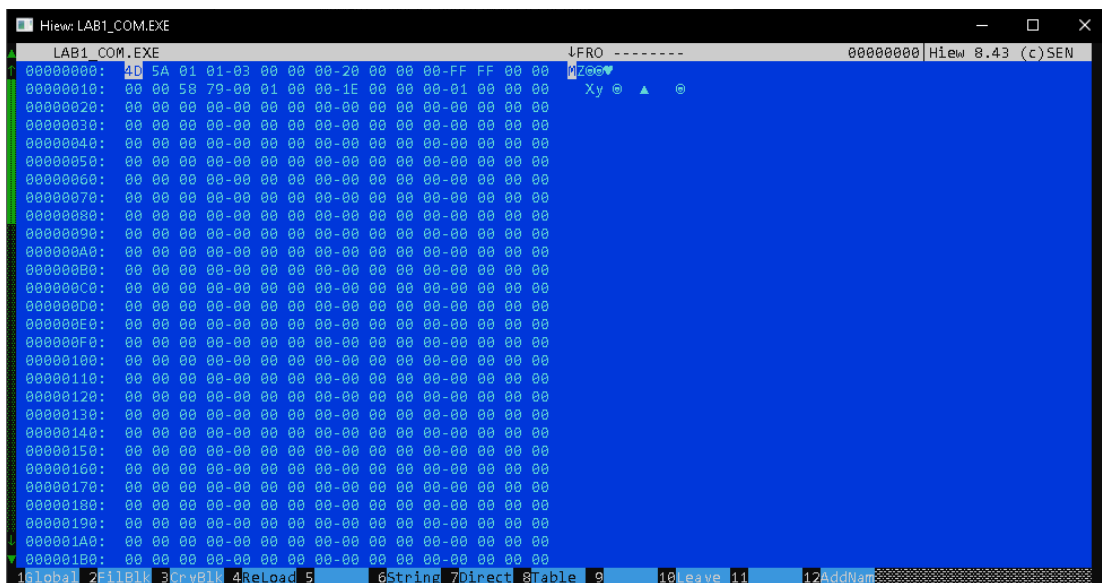


Рисунок 5 - «Плохой» .EXE модуль в шестнадцатеричном виде

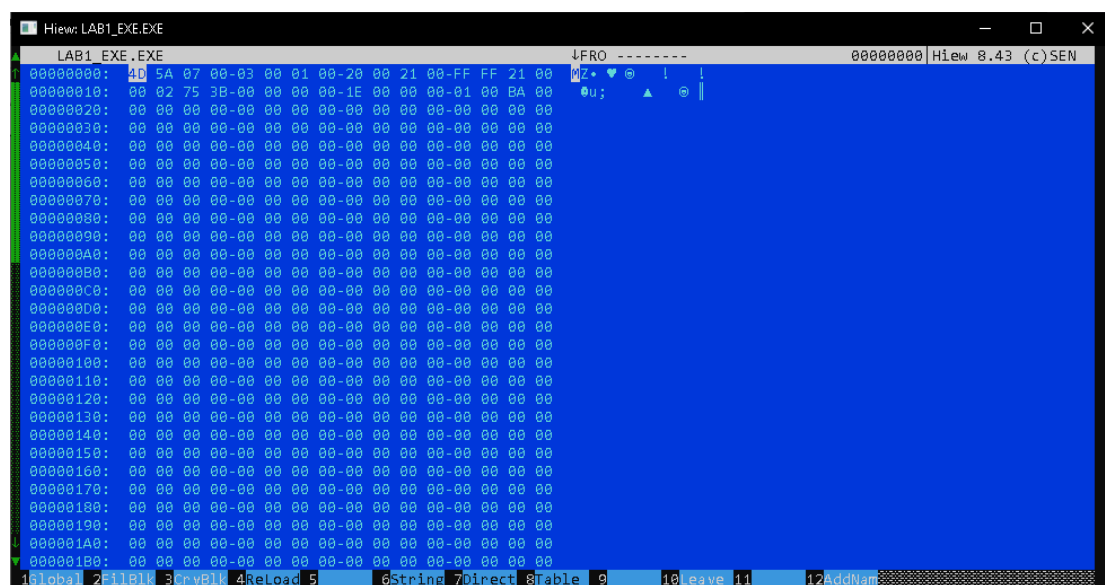


Рисунок 6 - «Хороший» .EXE модуль в шестнадцатеричном виде

## Ответы на контрольные вопросы. Отличия форматов файлов COM и EXE программ.

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

COM файл состоит из одного сегмента и содержит данные и машинные команды. Код начинается с адреса 0h, но при загрузке модуля устанавливается смещение в 100h.

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

В «плохом» EXE файле данные и код содержатся в одном сегменте. Код располагается с адреса 300h. С адреса 0h располагается Relocation Table (таблица разметки).

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

В «хорошем» файле EXE содержится информация для загрузчика, сегмент стека, сегмент данных и сегмент кода (3 сегмента вместо одного в «плохом» .EXE). Код располагается с адреса 200h в отличии от 300h в «плохом» .EXE файле.

### Шаг 5. Загрузка COM модуля в основную память.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	SI	CS	IP	Stack	Flags
0000	0000	119C	0100	+0 0000	0200
BX	DI	DS		+2 CD00	
CX	BP	ES	HS	+4 0420	OF DF IF SF ZF AF PF CF
DX	SP	SS	FS	+6 006A	0 0 1 0 0 0 0 0

CMD >

Address	Instruction	Disassembly	Hex
0100	E97E01	JMP 02B1	CD 20 04 6A 00 EA FD FF
0103	49	DEC CX	AD DE ED 04 92 01 00 00
0104	42	INC DX	18 01 10 01 18 01 92 01
0105	4D	DEC BP	FF FF FF FF FF FF FF FF
0106	205043	AND [BX+SI+43], DL	FF FF FF FF FF FF FF FF
0109	205459	AND [SI+59], DL	FF FF FF FF 96 11 E4 FF
010C	50	PUSH AX	92 01 14 00 18 00 9C 11
010D	45	INC BP	FF FF FF FF 00 00 00 00

Address	Hex	ASCII
DS:0000	CD 20 04 6A 00 EA FD FF	. . j . . . . .
DS:0010	18 01 10 01 18 01 92 01	. . . . .
DS:0020	FF FF FF FF FF FF FF FF	. . . . .
DS:0030	92 01 14 00 18 00 9C 11	. . . . .
DS:0040	05 00 00 00 00 00 00 00	. . . . .

1 Step 2 StepProc 3 Retrieve 4 Help 5 Set BRK 6 7 up 8 dn 9 le 0 ri



**Ответы на контрольные вопросы. Загрузка COM модуля в основную память.**

- После загрузки COM-программы в память сегментные регистры указывают на начало PSP. Код располагается с адреса 100h (ip = 0100h).

- Адрес начала PSP.

- 48DDh. Они указывают на начало PSP.

- Стек определяется автоматически, указатель стека устанавливается на конец сегмента. Если для программы размер сегмента в 64КБ является достаточным, то DOS устанавливает в регистре SP адрес конца сегмента – FFFh. Адреса расположены в диапазоне 0000h-FFFh.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD
AX 0000 SI 0000 CS 119C IP 0100 Stack +0 0000 FLAGS 0200
BX 0000 DI 0000 DS 119C +2 CD00
CX 0000 BP 0000 ES 119C HS 119C +4 A020 OF DF IF SF ZF AF PF CF
DX 0000 SP FFFD SS 119C FS 119C +6 007B 0 0 1 0 0 0 0 0

CMD >L lab1_exe.exe

0100 E97E01 JMP 0281
0103 49 DEC CX
0104 42 INC DX
0105 4D DEC BP
0106 205043 AND [BX+SI+431],DL
0109 205459 AND [SI+591],DL
010C 50 PUSH AX
010D 45 INC BP

0 1 2 3 4 5 6 7 8 9 A B C D E F
DS:0000 CD 20 A0 7B 00 EA FD FF AD DE ED 04 92 01 00 00 .f....
DS:0010 18 01 10 01 18 01 92 01 FF FF FF FF FF FF FF FF .....
DS:0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
DS:0030 92 01 14 00 18 00 9C 11 FF FF FF FF 00 00 00 00 .....
DS:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

1 Step 2StepProc 3Retrieve 4 Help 5Set BRK 6 7un 8dn 9le 0r

```

## Рисунок 8 – Загрузка «хорошего» EXE модуля в память

Ответы на контрольные вопросы. Загрузка «хорошего» EXE модуля в память.

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

В области памяти строится PSP, стандартная часть заголовка считывается в память, определяется длина тела загрузочного модуля, определяется начальный сегмент, загрузочный модуль считывается в начальный сегмент, таблица настройки считывается в рабочую память, определяются значения сегментных регистров. DS и ES устанавливаются на начало PSP, SS - на начало стека, CS - на начало сегмента кода.

### 2) **На что указывают регистры DS и ES?**

DS и ES указывают на начало PSP. После выполнения команд `mov ax, @data` и `mov ds, ax` регистре DS содержит адрес начала сегмента данных.

### 3) **Как определяется стек?**

В исходном коде модуля стек определяется при помощи директивы `STACK`, а при исполнении в регистры SS и SP записываются адрес начала сегмента стека и его вершины соответственно.

### 4) **Как определяется точка входа?**

При помощи команды `END`.

## **Вывод.**

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