**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. 6381 |  | Поляков Н. С. |
| Преподаватель |  | Губкин А. Ф. |

Санкт-Петербург

2018

**Цель работы.**

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

**Основные теоретические положения.**

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

**PC FF**

**PC/XT FE,FB**

**AT FC**

**PS2 модель 30 FA**

**PS2 модель 50 или 60 FC**

**PS2 модель 80 F8**

**PCjr FD**

**PC Convertible F9**

Для определения версии MS DOS следует воспользоваться функцией 30H прерывания 21H. Входным параметром является номер функции в AH:

**MOVAH,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:CX** – 24-битовый серийный номер пользователя.

**Описание функций.**

|  |  |
| --- | --- |
| Имя | Описание |
| TETR\_TO\_HEX | осуществляет перевод половины байта в AL в символ в 16-ричной с/c и помещает результат в AL |
| BYTE\_TO\_HEX | осуществляет перевод байта в AL, в два символа в 16-ричной c/c и помещает результат в AX |
| WRD\_TO\_HEX | осуществляет перевод байта в AL, в два символа в 16-ричной c/c и помещает результат в AX |
| BYTE\_TO\_DEC | осуществляет перевод байта в AL, в два символа в 10-тичной c/c и помещает результат в SI |
| PCTYPE | читает содержимое предпоследнего байта ROM BIOS для определения типа PC и выводит его на экран |
| VERNUM | определяет номера основной версии и модификации MS DOS, находящийся в регистре AL после вызова функции 30H прерывания 21H, и выводит их на экран |
| SERNUMM | определяет номера основной версии и модификации MS DOS, находящийся в регистре AL после вызова функции 30H прерывания 21H, и выводит их на экран |
| USERNUM | определяет 24-битовый серийный номер пользователя, находящийся в регистрах BL:CX после вызова функции 30H прерывания 21H, и выводит его на экран |

**Описание структур данных.**

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| TYPE\_OF\_PC | db | Строка “Type of IBM PC: ” |
| TYPE\_PC | db | Строка “PC” |
| TYPE\_PCXT | db | Строка “PC/XT” |
| TYPE\_AT | db | Строка “AT” |
| TYPE\_PS2m30 | db | Строка “PS2 model 30” |
| TYPE\_PS2m50 | db | Строка “PS2 model 50 or 60” |
| TYPE\_PS2m80 | db | Строка “PS2 model 80” |
| TYPE\_PCJR | db | Строка “PCjr” |
| TYPE\_PCC | db | Строка “PC Convertible” |
| TYPE\_CODE | db | Строка “PC Code” |
| DOS\_VER | db | Строка “MS DOS version: 00.00” |
| SERIAL\_NUM | db | Строка “Serial number OEM: ” |
| USER\_SER\_NUM | db | Строка “User serial number: 000000” |

**Ход работы.**

1. Результат работы «хорошего» .COM модуля (lab1c.com):



Рис. 1. Результат работы lab1c.com.

1. Результат работы «плохого» .EXE модуля (lab1c.exe):

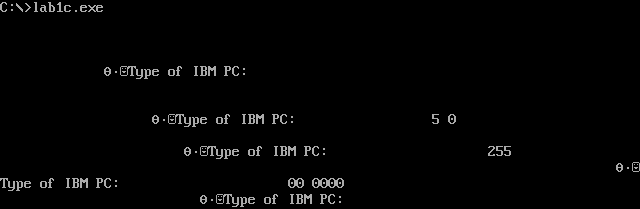


Рис. 2. Результат работы lab1c.exe.

1. Результат работы «хорошего» .EXE модуля (lab1e.exe):

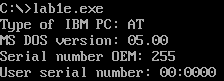


Рис. 3. Результат работы lab1e.exe.

**Ответы на контрольные вопросы.**

1. **Отличия исходных текстов COM и EXE программ.**
2. Сколько сегментов должна содержать COM-программа?

Ответ: COM-программа содержит один сегмент.

1. EXE-программа?

Ответ: EXE-программа может содержат один или более сегментов. В частности предусматриваются отдельные сегменты для кода, данных и стека.

1. Какие директивы должны обязательно быть в тексте COM-программы?

Ответ: Обязательно должна присутствовать директивы:

* ORG 100h, которая сдвигает адресацию, так как в первых 256 байтах находится управляющая информация;
* ASSUME, которая устанавливает соответствие между сегментными регистрами и сегментами в программе;
* END, с помощью которой определяется точка входа в программу.

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

Ответ: Нет, не все. COM-программа подразумевает наличие только одного сегмента, а значит, можно использовать только near-переходы. Так же нельзя использовать команды, связанные с адресом сегмента, так как в таких командах используется relocation table, в которой содержатся адреса сегментов. Такой таблицы нет в COM-программе.

1. **Отличия форматов файлов COM и EXE модулей.**
2. Какова структура файла COM? С какого адреса располагается код?

Ответ: Файл COM состоит из команд, процедур и данных, используемых в программе. Код начинается с нулевого адреса (0h).

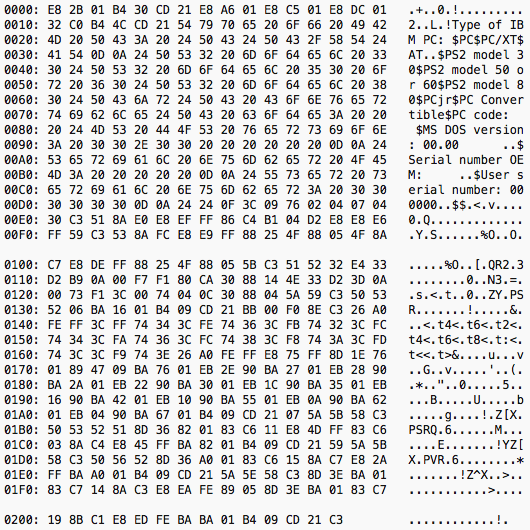


Рис. 4. Файл lab1c.com в 16-ричном виде.

1. Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с адреса 0?

Ответ: В «плохом» файле EXE данные и код содержатся в одном сегменте. Код располагается с адреса 300h. С нулевого адреса располагается информация для загрузчика.

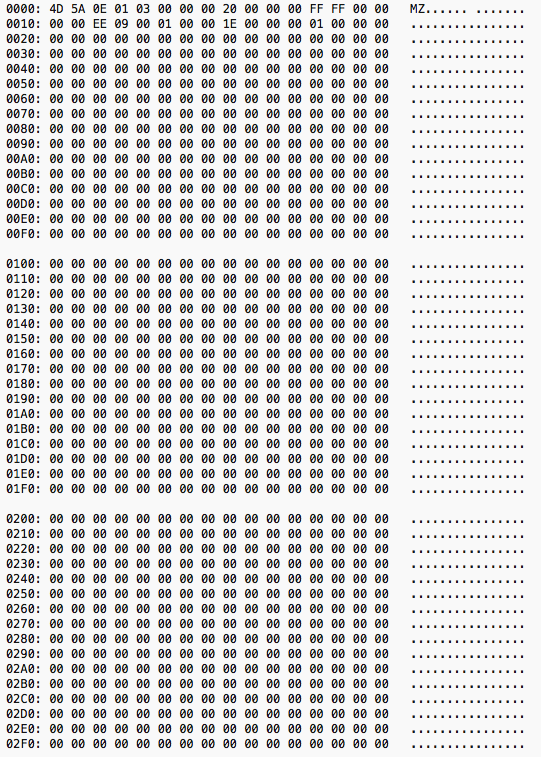
****

Рис. 5. Файл lab1с.exe в 16-ричном виде (плохой EXE).

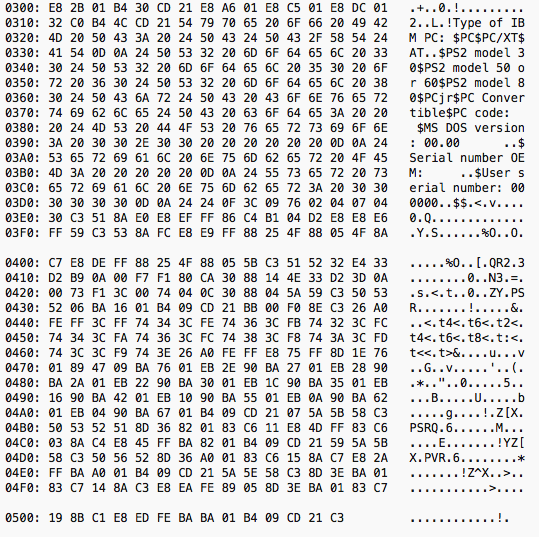
****

Рис. 6. Файл lab1с.exe в 16-ричном виде (плохой EXE, продолжение).

1. Какова структура файла «хорошего» EXE? Чем он отличается от файла «плохого» EXE?

Ответ: В «хорошем» файле EXE содержится информация для загрузчика (00h: символы MZ; 02h: количество байт на последней странице файла; 04h: количество страниц в файле; 08h: размер заголовка в параграфах; 0Ch: максимальное выделение памяти в параграфах; 12h: контрольная сумма; 14h: начальное значение регистра IP; 16h: начальное значение регистра CS), а также программа, разделённая на сегменты кода, данных и стека. В «плохом» EXE код, данные и стек находятся в одном сегменте, а также из-за использования директивы ORG 100h этот сегмент начинается с адреса 300h, а не с 200h как в хорошем EXE (200h: stack segment; 250h: data segment; 320h: code segment).

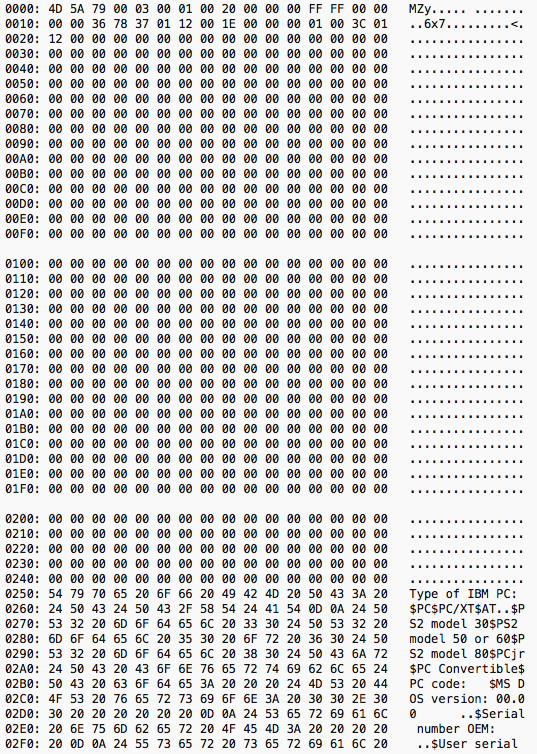


Рис. 7. Файл lab1e.exe в 16-ричном виде (хороший EXE).

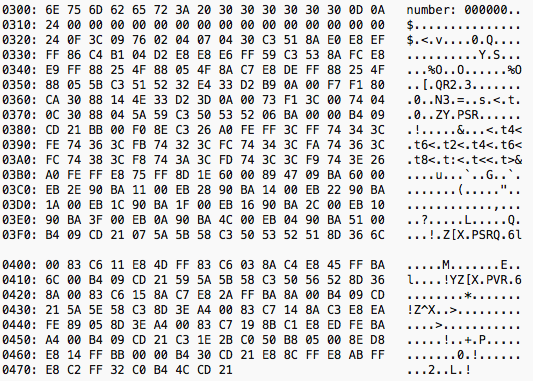


Рис. 8. Файл lab1e.exe в 16-ричном виде (хороший EXE, продолжение).

1. **Загрузка COM модуля в основную память.**
2. Какой формат загрузки модуля COM? С какого адреса располагается код?

Ответ: После загрузки COM-программы в память, все сегментные регистры указывают на начало выделенной для программы памяти. Код располагается через 100h байт от начала сегмента (IP=100).

1. Что располагается с адреса 0?

Ответ: С адреса 0 располагается PSP.

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

Ответ: Все сегментные регистры имеют значения 119С. Они указывают на PSP.

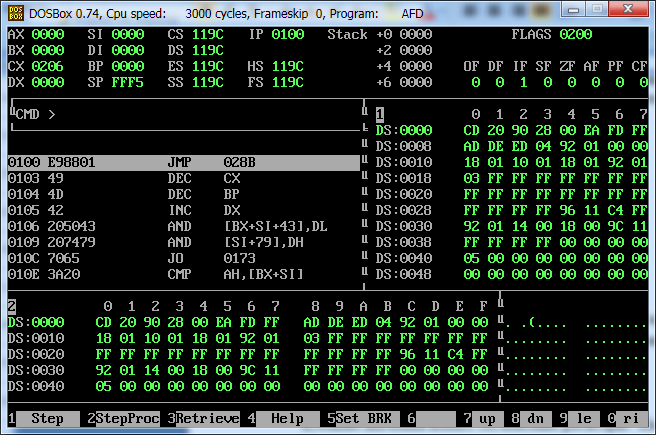
****

Рис. 9. Сегментные регистры lab1c.com.

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

Ответ: Сегмент стека генерируется в COM-файлах автоматически. Указатель стека устанавливается на конец 64 КБ сегмента. Следовательно, он занимает оставшуюся память, а адреса изменяются от FFFEh к 0000h.

1. **Загрузка «хорошего» EXE модуля в основную память.**
2. Как загружается «хороший» EXE? Какие значения имеют сегментные регистры?

Ответ: Сначала создаётся PSP, определяется длина тела загрузочного модуля, определяется начальный сегмент. Затем загрузочный модуль считывается в начальный сегмент, таблица настройки считывается в рабочую память, к полю каждого сегмента прибавляется сегментный адрес начального сегмента и определяются значения сегментных регистров. DS и ES указывают на начало сегмента PSP (119C), CS указывает на начало сегмента кода (11BE), а SS – на начало сегмента стека (11AC).



Рис. 10. Сегментные регистры lab1e.exe.

1. На что указывают регистры DS и ES?

Ответ: Регистры DS и ES указывают на начало PSP.

1. Как определяется стек?

Ответ: Для стека определяется сегмент стека указанного размера.

1. Как определяется точка входа?

Ответ: Точка входа в программу определяется с помощью директивы END “метка”.

**Вывод.**

В ходе лабораторной работы былиисследованы различия в структурах исходных текстов модулей **.COM** и **.EXE**, структур файлов загрузочных модулей и способов их загрузки в основную память.

**Приложение 1. Код программы.**