МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №1

по дисциплине «Операционные системы»

Тема: Исследование структур загрузочных модулей

Студент гр. 8382 _____ Кобенко В.П.

Преподаватель _____ 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.

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:

```
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:CX – 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 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 (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 AX.

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 «хорошего» .COM модуля.

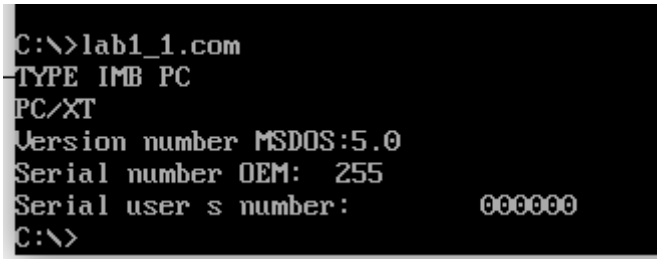

Рисунок 1 – «Хороший» .COM модуль

Запуск «плохого» .EXE модуля.

Рисунок 2 – «Плохой» .EXE модуль

Шаг 2. Запуск «хорошего» .EXE модуля.

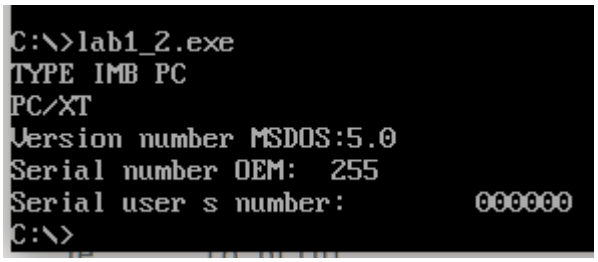

Рисунок 3 – «Хороший» .EXE модуль

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 располагает код программы только после этого блока. Директива ASSUME, ставящая в соответствие начало программы сегментам кода и данных.

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

Нет, не все, так как в отличие от EXE-программы, в которой существует таблица настроек (таблица разметки), называемая Relocation Table, COM-программа ею не располагает. Адреса сегментов определяются загрузчиком в момент запуска программы на основе информации о местоположении полей адресов в файле из Relocation Table. Следовательно, в связи с отсутствием этой таблицы в COM-программах, команды вида mov [регистр], seg [сегмент] недопустимы.

Шаг 4. .COM модуль в шестнадцатеричном виде.

Рисунок 4 - .COM модуль в шестнадцатеричном виде

«Плохой» .EXE модуль в шестнадцатеричном виде.

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

«Хороший» .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.

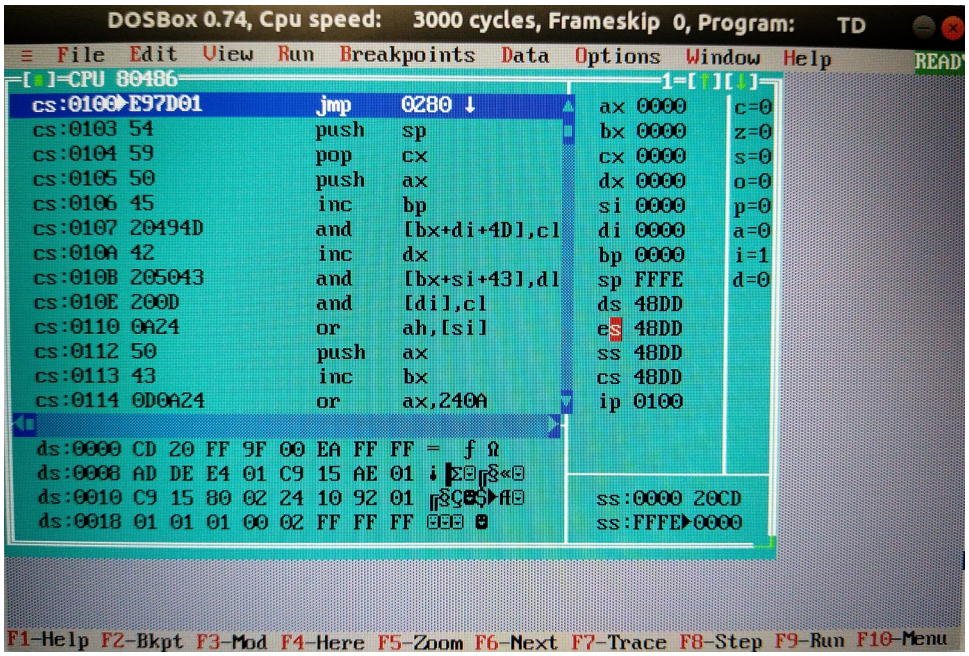

Рисунок 7 – Загрузка COM модуля в основную память

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

1) **Какой формат загрузки COM модуля? С какого адреса располагается код?**

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

2) **Что располагается с 0 адреса?**

Адрес начала PSP.

3) **Какие значения имеют сегментные регистры? На какие области памяти они указывают?**

48DDh. Они указывают на начало PSP.

4) **Как определяется стек? Какую область памяти он занимает? Какие адреса?**

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 – FFFEh. Адреса расположены в диапазоне 0000h-FFFEh.

Шаг 6. Загрузка «хорошего» EXE модуля в память.

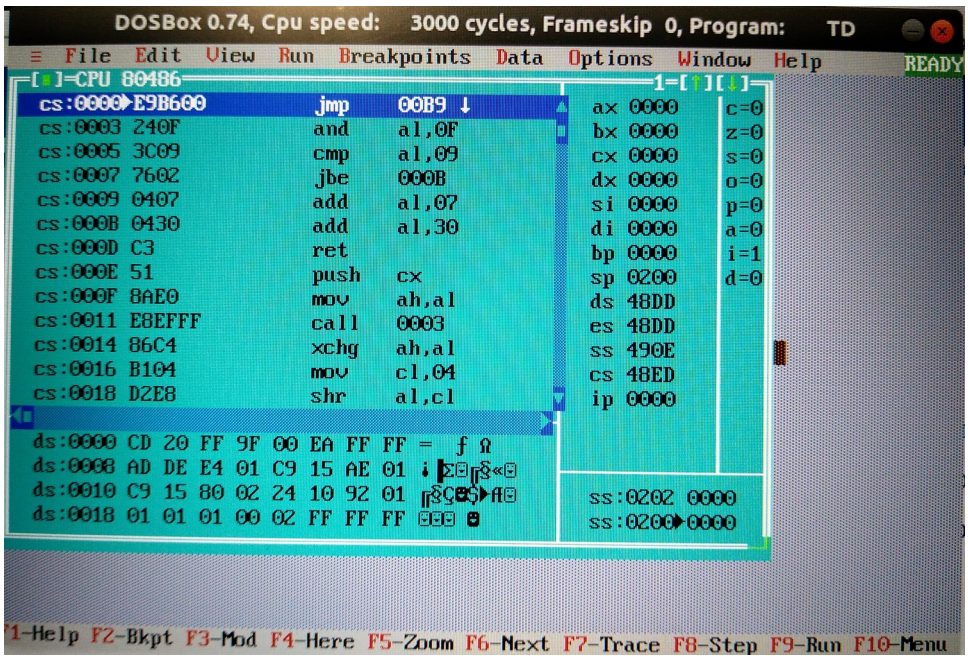

Рисунок 8 – Загрузка «хорошего» EXE модуля в память

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.