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

Кафедра Математического обеспечения электронно-вычислительных машин

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Операционные системы»

ТЕМА: ИССЛЕДОВАНИЕ СТРУКТУР ЗАГРУЗОЧНЫХ МОДУЛЕЙ.

Студентка гр. 0382 _____ Рубежова Н.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Изучить и исследовать различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

Задание.

1. Написать текст исходного .COM модуля, который определяет тип PC и версию системы. Результатом будет “хороший” .COM модуль, а также “плохой” .EXE, полученный из исходного текста для .COM модуля.

2. Написать текст исходного .EXE модуля, который выполняет те же функции, что и модуль в Шаге 1. Результатом будет “хороший” .EXE.

3. Сравнить исходные тексты для .COM и .EXE модулей. Ответить на контрольные вопросы “Отличия исходных текстов COM и EXE программ”.

4. Запустить FAR и открыть файл загрузочного модуля .COM и файл “плохого” .EXE в шестнадцатеричном виде. Затем открыть файл загрузочного модуля “хорошего” .EXE и сравнить его с предыдущими файлами. Ответить на контрольные вопросы “Отличия форматов файлов COM и EXE модулей”.

5. Открыть отладчик TD.EXE и загрузить .COM. Ответить на контрольные вопросы “Загрузка COM модуля в основную память”. Представить в отчете план загрузки .COM модуля в основную память.

6. Открыть отладчик TD.EXE и загрузить “хороший” .EXE. Ответить на контрольные вопросы “Загрузка “хорошего” EXE модуля в основную память”.

7. Оформить отчет в соответствии с требованиями. В отчете необходимо привести скриншоты. Для файлов их вид в шестнадцатеричном виде, для загрузочных модулей – в отладчике.

Ход выполнения.

1. Был написан исходный текст COM-модуля, который определяет тип PC и версию системы. Процедура PC_TYPE реализовывает определение типа PC, обращаясь к предпоследнему байту ROM BIOS и сравнивая коды по таблице, представленной в методическом пособии. В результате сравнения

выводится строка с названием модели. Процедура DOS_VER формирует текстовые строки с информацией о версии системы, серийным номером OEM и серийным номером пользователя, а затем полученные строки выводит на экран вызовом процедуры PRINT, которая отвечает за вывод строки на экран. В результате компиляции и сборки имеем «хороший» .COM модуль и «плохой» .EXE файл, полученный из исходного текста для .COM модуля. .EXE-модуль «плохой», поскольку имеются различия в структуре .COM и .EXE модулей. Например, в .COM-модуле код и данные должны быть определены в одном сегменте, в то время как в .EXE-модуле определяются отдельно сегмент данных, сегмент кода и сегмент стека.

Результаты выполнения см. на рисунках 1-2.

Рисунок 1 – Запуск «хорошего» .COM модуля: com_file.com

Рисунок 2 – Запуск «плохого» .EXE модуля: com_file.exe

2. Был построен «хороший» .EXE модуль, для этого определили сегменты данных, стека и кода, с помощью директивы ASSUME сопоставили сегменты и сегментные регистры выделили главную процедуру MAIN – точку старта выполнения программы.

Рисунок 3 – Запуск «хорошего» .EXE модуля: exe_file.exe

3. Сравним исходные тексты для .COM и .EXE модулей:

Контрольные вопросы по лабораторной работе:

1) Сколько сегментов должна содержать COM-программа?

Ответ: 1 сегмент, в .COM модуле код и данные определяются в ОДНОМ сегменте, а стек генерируется автоматически.

2) EXE-программа?

Ответ: один или больше, код и данные должны быть разделены на отдельные сегменты. В EXE-модуле обязательно определяется сегмент кода. Также по необходимости определяются сегменты данных и стека.

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

Ответ: Директива ORG 100h, так как когда COM-программа начинает работать, все сегментные регистры содержат адрес префикса программного сегмента (PSP), 256-байтового блока, который резервируется операционной системой DOS непосредственно перед COM или EXE программой в памяти. Так как адресация начинается с шест. смещения 100 от начала PSP, то в программе кодируется директива ORG 100H. А с помощью директивы ASSUME делаем так, чтобы сегмент данных и кода указывали на общий сегмент.

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

Ответ: Т.к. в COM-программе все сегментные регистры определяются в момент запуска программы, а не в момент компиляции, то невозможно использование команд с указанием сегментов.

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

Ответ: .COM модуль состоит из одного общего сегмента, который содержит и код, и данные. Код располагается с адреса 0100h, так как при загрузке COM-модуля директива ORG 100H (из-за блока PSP в 256 байт) устанавливает смещение 100h.

Модуль в шестнадцатеричном виде см. на рис. 4.

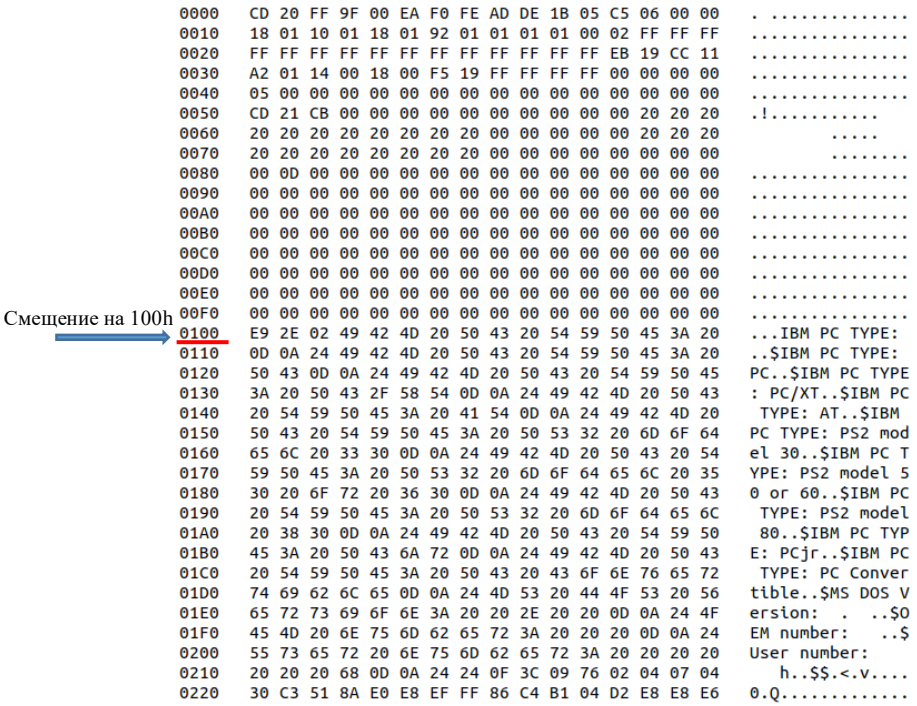

Рисунок 4 – 16-ый вид «хорошего» .COM модуля: com_file.com

6) Какова структура файла “плохого” EXE? С какого адреса располагается код? Что располагается с адреса 0?

Ответ: С адреса 0h располагается заголовок, relocation table, информация, используемая загрузчиком (всего 200h). Далее располагается код с адреса 300h(с учетом смещения 100h+200h), код, данные и стек находятся в одном сегменте.

Модуль в шестнадцатеричном виде представлен на рисунке 5.

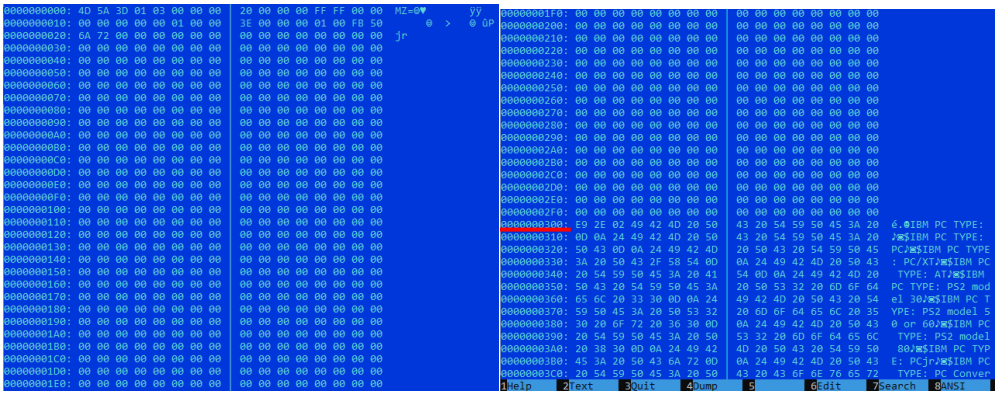

Рисунок 5 – “плохой” .EXE в 16-м виде

7) Какова структура файла “хорошего” EXE? Чем он отличается от файла “плохого” EXE?

Ответ: В «хорошем» .EXE модуле код, данные и стек разделены в отдельные сегменты, в отличие от «плохого» .EXE модуля. Сначала идут заголовок и relocation table (200h байт), затем сегмент стека, сегмент данных и сегмент кода.

Модуль в шестнадцатеричном виде представлен на рисунке 6.

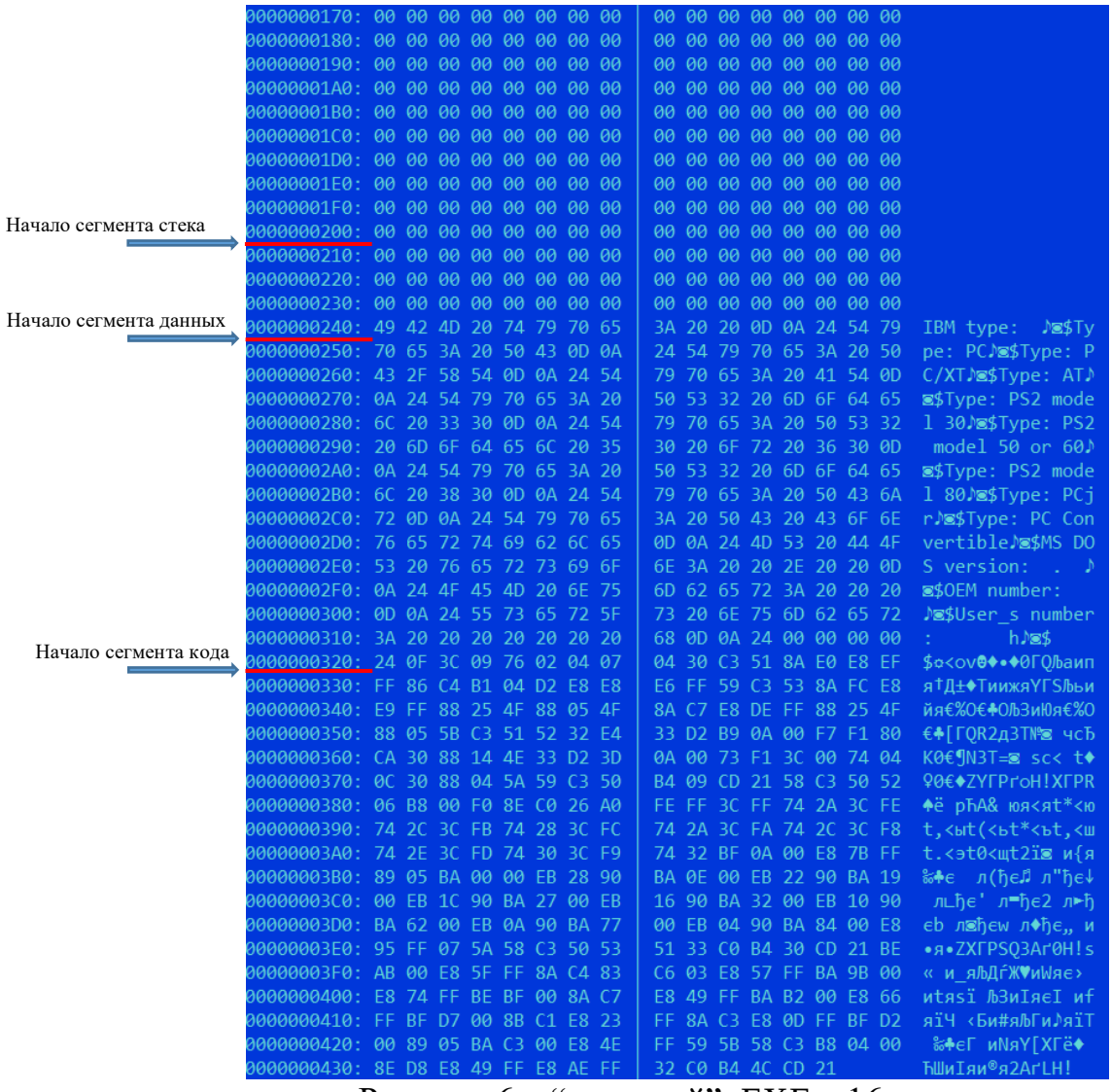

Рисунок 6 – “хороший” .EXE в 16-м виде

Результат загрузки .COM модуля в отладчик TD.EXE представлены на рисунке 7.

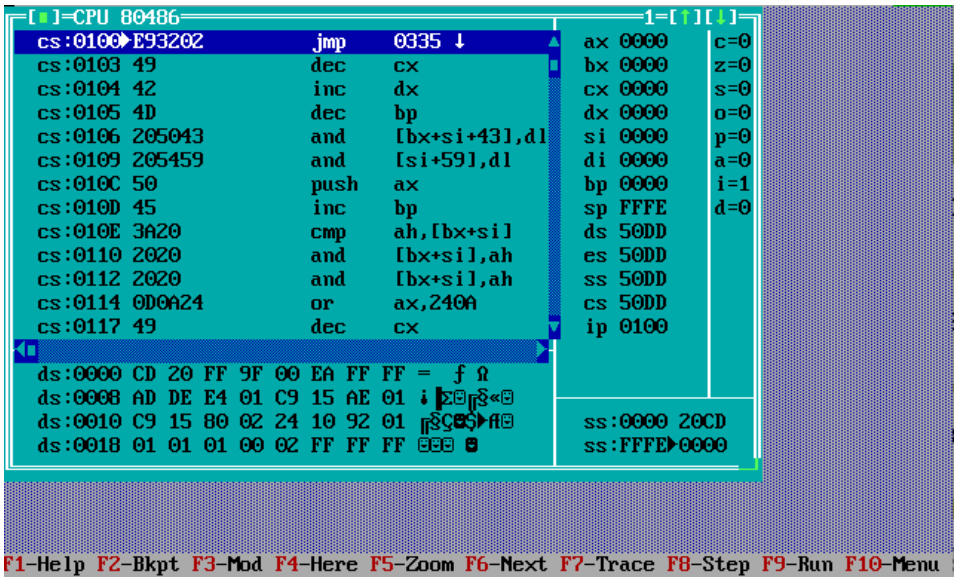

Рисунок 7 – Отладка “хорошего” .COM модуля

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

Ответ: Код располагается с адреса CS:100h. Для загрузки .COM модуля находится достаточный по размеру свободный участок ОП и определяется его сегментный адрес. Создаются блоки памяти, под информацию об окружении файла, PSP и программу. Выполняется чтение файла с диска и запись его в память с адреса PSP:100h.

9) Что располагается с адреса 0?

Ответ: Блок PSP(размером 256 байт).

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

Ответ: Сегментные регистры CS, DS, ES, SS устанавливаются на начало PSP (для нашей программы – значение 50DD, см. рис.8).

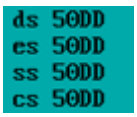

Рисунок 8 – Значения сегментных регистров

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

Ответ: Для .COM модуля стек генерируется автоматически. Регистр SS=0h указывает на начало стека, а SP=FFFEh – на конец стека. Адреса стека имеют диапазон [0000h, FFFEh].

Результат загрузки «хорошего» .EXE модуля в отладчик TD.EXE представлены на рисунке 9.

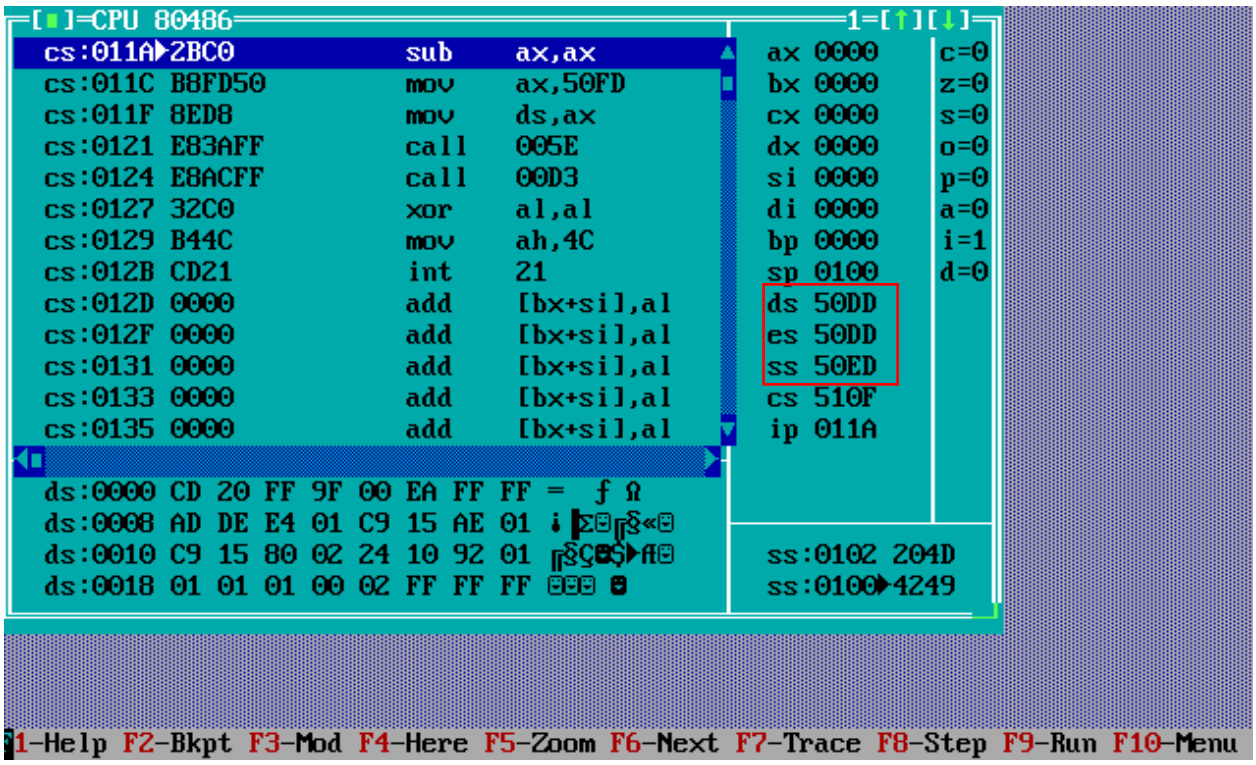

Рисунок 9 – загруженный в отладчик “хороший” .EXE модуль

12) Как загружается “хороший” EXE? Какие значения имеют сегментные регистры?

Ответ: Для загрузки .EXE модуля находится достаточный по размеру свободный участок ОП и определяется его сегментный адрес. Создаются блоки памяти, под информацию об окружении файла, PSP и программу. Затем, считывая информацию в заголовке, вычисляется размер загрузочного модуля, его смещение, сегментный адрес загрузки и далее считывается сам .EXE модуль. При инициализации DS, ES устанавливаются на начало PSP, в SP – из заголовка, SS указывает на сегментный адрес загрузки (начало PSP + 10h).

13) На что указывают регистры DS и ES?

Ответ: Регистры DS, ES указывают на начало блока PSP

14) Как определяется стек?

Ответ: Стек описывается в коде самой программы. А директива assume устанавливает сегментный регистр на сегмент стека. В итоге, SS будет указывать на начало сегмента стека, SP – на конец сегмента стека.

15) Как определяется точка входа?

Ответ: С помощью директивы END <метка>. Метку ставят в коде, непосредственно в месте, откуда начинается выполнение программы, т.е. метка – это точка входа.

Выводы.

Были изучены и исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.