**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. 0382 |  | Азаров М.С. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2021

## Цель работы.

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

## Задание.

**Шаг 1.** Напишите текст исходного .СОМ модуля, который определяет тип РС и версию системы. Это довольно простая задача и для тех, кто уже имеет опыт программирования на ассемблере, это будет небольшой разминкой. Для тех, кто раньше не сталкивался с программированием на ассемблере, это неплохая задача для первого опыта. За основу возьмите шаблон, приведенный в разделе «Основные сведения». Необходимые сведения о том, как извлечь требуемую информацию, представлены в следующем разделе. 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 и серийным номером пользователя. Полученные строки выводятся на экран. Отладьте полученный исходный модуль. Результатом выполнения этого шага будет «хороший» .СОМ модуль, а также необходимо построить «плохой» .ЕХЕ, полученный из исходного текста для .СОМ модуля.

**Шаг 2.** Напишите текст исходного .ЕХЕ модуля, который выполняет те же функции, что и модуль в Шаге 1 и постройте и отладьте его. Таким образом, будет получен «хороший» .ЕХЕ.

**Шаг 3.** Сравните исходные тексты для .СОМ и .ЕХЕ модулей. Ответьте на контрольные вопросы «Отличия исходных текстов СОМ и ЕХЕ программ».

**Шаг 4**. Запустите FAR и откройте (F3/F4) файл загрузочного модуля .СОМ и файл «плохого» .ЕХЕ в шестнадцатеричном виде. Затем откройте (F3/F4) файл загрузочного модуля «хорошего» .ЕХЕ и сравните его с предыдущими файлами. Ответьте на контрольные вопросы «Отличия форматов файлов СОМ и ЕХЕ модулей».

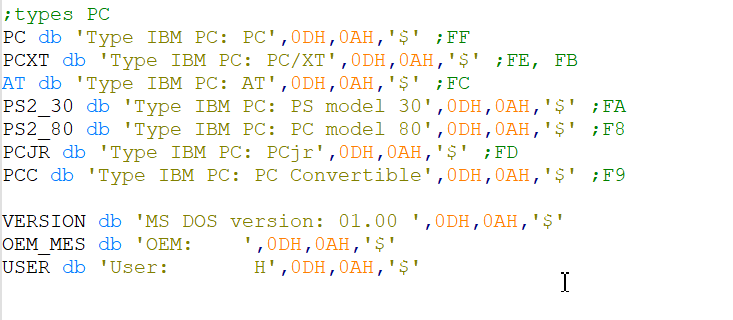
**Шаг 5**. Откройте отладчик TD.EXE и загрузите .СОМ. Ответьте на контрольные вопросы «Загрузка СОМ модуля в основную память». Представьте в отчете план загрузки модуля .СОМ в основную память.

**Шаг 6**. Откройте отладчик TD.EXE и загрузите «хороший» .ЕХЕ. Ответьте на контрольные вопросы «Загрузка «хорошего» ЕХЕ модуля в основную память».

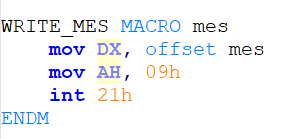
**Шаг 7**. Оформление отчета в соответствии с требованиями. В отчете необходимо привести скриншоты. Для файлов их вид в шестнадцатеричном виде, для загрузочных модулей – в отладчике.

## Ход работы

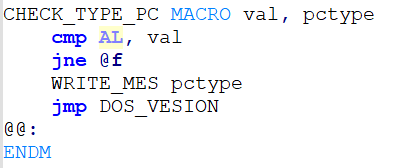
1. Создаем исходник для будущего COM файла.
   1. Берем за основу исходник для СОM из методички.
   2. Сохраняем в программе все сообщения, которые она может вывести.



* 1. Создаем макрос **WRITE\_MES** для вывода сообщений.

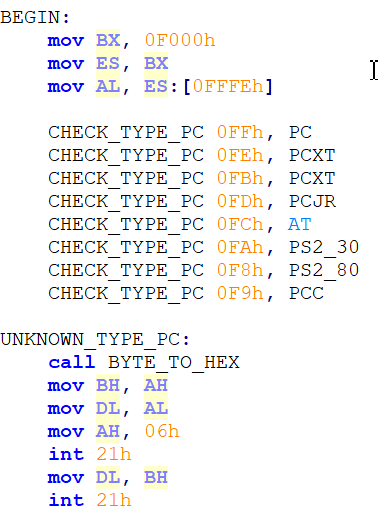


* 1. Создаем макрос для удобного определения типа системы.



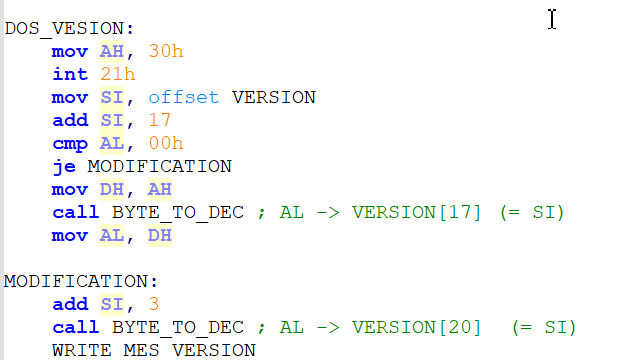
**val** – значение байта которому соответствует определённая система.

* 1. Выполняем первое задание по определению типа системы.

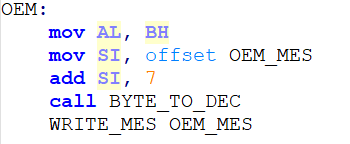


Раздел **UNKNOWN\_TYPE\_PC** нужен для случаев, когда определяемого типа системы нет в известном списке.

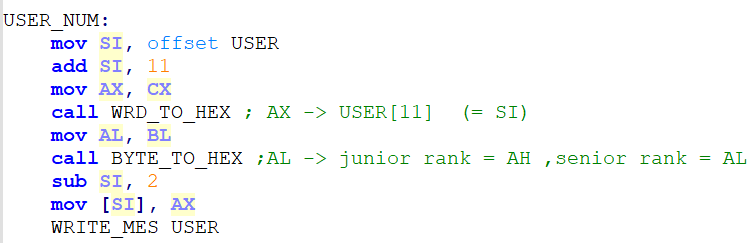
* 1. Определяем версию системы:



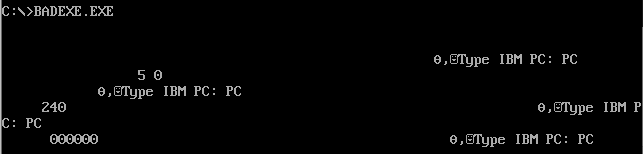
* 1. Определяем серийный номер OEM:



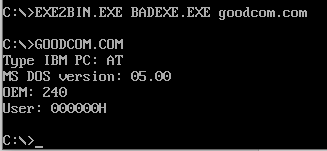
* 1. Определяем номер пользователя:



1. Создаем «плохой» EXE. Результат работы плохого EXE:



1. Создаем COM файл из «плохого» EXE с помощью программы EXE2BIN.EXE:



Программа работает корректно.

1. Создаем исходник для будущего EXE файла.
2. Создаем загрузочный модуль хорошего EXE. Проверяем корректность:



Программа работает корректно.

**Ответы на контрольные вопросы.**

Отличия исходных текстов . COM и . EXE программ:

1) Сколько сегментов должна содержать .COM программа?

**Ответ:** Программа для COM файла должна содержать только один сегмент – он используется, как сегмент кода, так и сегмент данных. Сегмент стека создается автоматически в этом же сегменте.

2) .EXE программа?

**Ответ:** От одного в который можно заключить и код и данные и стек, до четырех – сегмент кода , сегмент данных , стек и дополнительный сегмент.

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

**Ответ:** Необходимо директива ORG 100h, для смещения сегмента, чтобы не попасть в область PSP. Также необходимо использовать ASSUME (CS:TESTPC, DS:TESTPC, etc), чтобы связать сегмент с сегментными регистрами.

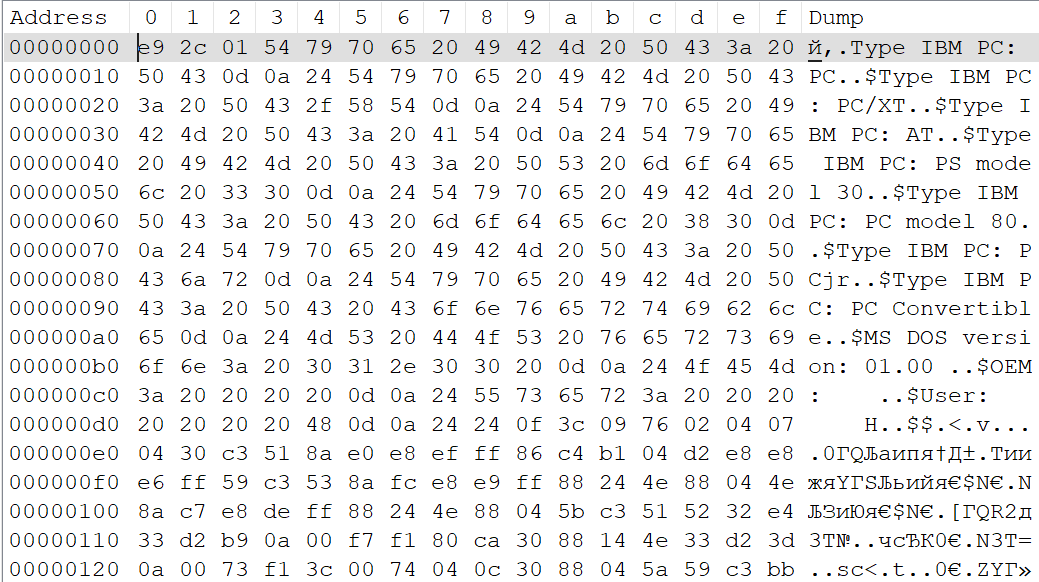
4) Все ли форматы команд можно использовать в .COM программе?

**Ответ:** Нет, команды с указанием сегментов не могут быть выполнены. В момент ассемблирования и редактирования связей сегментное значение для сегмента неизвестно. Оно определяется только при загрузке программы. Поскольку файл типа .COM не может предоставить загрузчику перечня всех сегментных ссылок (информация для перемещения), то в данном случае программа будет выполняться неправильно.

Отличия форматов файлов . COM и . EXE модулей

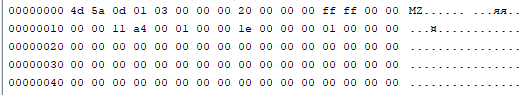
1. Какова структура файла COM? С какого адреса располагается код?

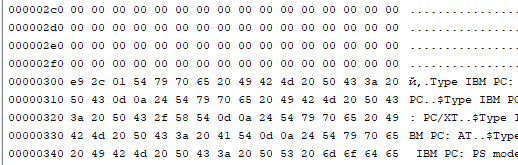
**Ответ:** COM файл состоит из одного сегмента в котором находятся и код и данные. Также в этом сегменте автоматически создается стек. Код располагается с адреса 0h , но из-за директивы ORG 100h будет загружен в ОП со смещением начиная с адреса 100h.



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

**Ответ:** Плохой EXE содержит только один сегмент. Причем сначала располагается заголовок, потом смещение в 100h из-за директивы org 100h и потом основной сегмент начиная с 300h. Как раз-таки с адреса 0h располагается заголовок EXE файла.

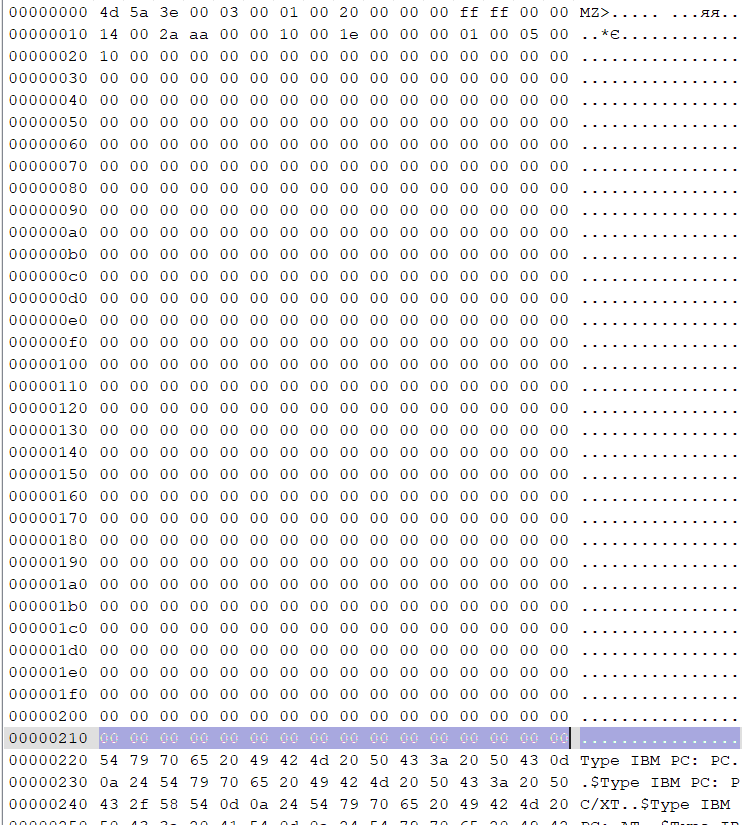




3) Какова структура файла «хорошего» EXE? Чем он отличается от файла «плохого» EXE?

**Ответ:** В хорошем EXE сначала располагается заголовок. Заголовок содержит необходимую информацию для загрузки программы в память и специальную таблицу, необходимую для настройки ссылок на дальние сегменты программы. Затем идут сегменты в очерёдности их объявления в исходнике.

~~Ну, во-первых, главное отличие хорошего от плохого EXE в том, что хороший работает корректно, а плохой нет.~~ Оба начинаются с заголовка, но в «плохом» после заголовка идет просто смещение на 100h и затем единственный сегмент, а в «хорошем» после заголовка идет сегмент стека (с 210h), затем сегмент данных и потом сегмент кода.



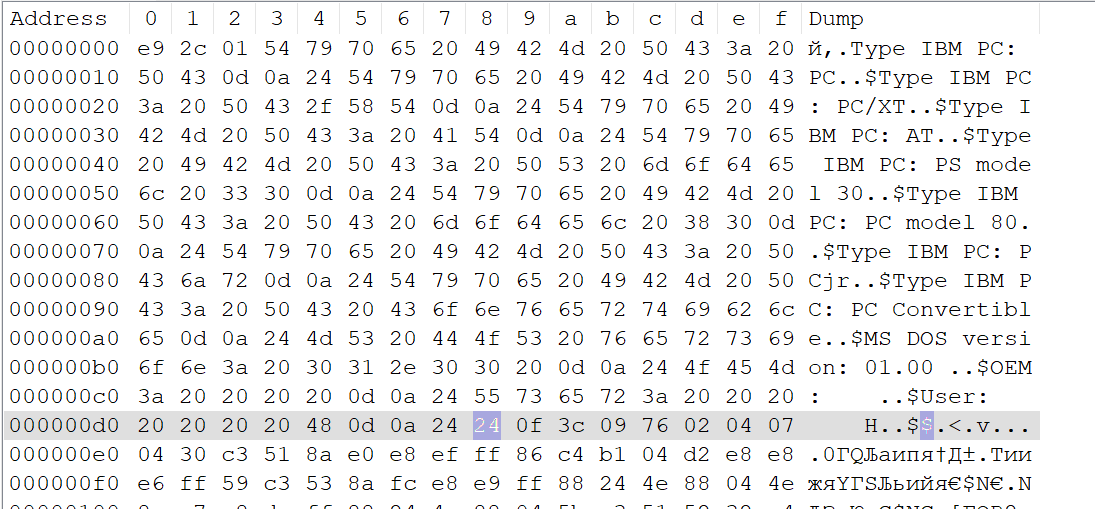
4) Как определяется стек? Какую область памяти он занимает? Какие адреса?

**Ответ:** Стек находится между заголовком и сегментом данных и занимает 210h-21Fh

Загрузка .COM модуля в основную память

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

**Ответ:** COM файл состоит из одного сегмента в котором вначале прописаны данные, затем код а вершина стека находится в конце сегмента и растет вверх. Вроде как с адреса d8h.



1. Что располагается с адреса 0?

**Ответ**: Если имеется ввиду ОП то PSP, если же загрузочный

модуль то сегмент.

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

**Ответ**: Сегментные регистры CS, DS, ES и SS указывают на сегмент кода, сегмент данных, дополнительные данные и стек соответственно. В начале выполнения программы регистры указывают на начало PSP

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

**Ответ**: Стек генерируется автоматически при создании .COM программы в конце сегмента. Адреса стека расположены в диапазоне FFFEh – 0h.

Загрузка «хорошего» ЕХЕ модуля в основную память

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

**Ответ**: Загрузочный модуль считывается в начальный сегмент. Таблица настройки порциями считывается в рабочую память. Для каждого элемента таблицы настройки к полю сегмента прибавляется сегментный адрес начального сегмента. В результате элемент таблицы указывает на слово в памяти, к которому прибавляется сегментный адрес начального сегмента. Когда таблица настройки адресов обработана, в регистры SS и SP записываются значения, указанные в заголовке, а к SS прибавляется сегментный адрес начального сегмента. В ES и DS записывается сегментный адрес начала PSP. Управление передается по адресу, указанному в заголовке

2)На что указывают регистры DS и ES?

**Ответ**: В начале выполнения программы они указывают на PSP.

3) Как определяется стек?

**Ответ**: Стек определяется с помощью директивы .stack, или вручную с помощью директивы segment.

1. Как определяется точка входа?

**Ответ**: Директивой END.

## Вывод.

В ходе работы были изучены различия в структурах исходных текстовых модулей типов **.COM** и **.EXE**, а также различия в загрузочных модулей этих типов и способов их загрузки в основную память.