**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. 6304 |  | Виноградов К.А. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Постановка задачи**

**Цель работы.**

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

**Функции управляющей программы.**

|  |  |
| --- | --- |
| **Название функции** | **Назначение функции** |
| Write\_msg | Вывод сообщения на экран |
| TETR\_TO\_HEX | Перевод числа из 2-ой в 16-ую с/с (1/2 байта) |
| BYTE\_TO\_HEX | Перевод числа из 2-ой в 16-ую с/с (1 байт) |
| WRD\_TO\_HEX | Перевод числа из 2-ой в 16-ую с/с (2 байта) |
| BYTE\_TO\_DEC | Перевод числа из 2-ой в 10-ую с/с (1 байт) |
| GET\_PC\_TYPE | Определяет тип IBM PC |
| GET\_SYS\_VER | Определяет версию системы |
| GET\_OEM\_NUM | Определяет OEM |
| GET\_SERIAL\_NUM | Определяет серийный номер пользователя |

**Структуры данных управляющей программы.**

|  |  |
| --- | --- |
| **Название структуры** | **Назначение структуры** |
| temp | Хранит индекс названия типа IBM PC |
| PC\_TYPE | Хранит тип IBM PC |
| SYSTEM\_VERSION | Хранит версию системы |
| OEM\_NUMBER | Хранит номер OEM |
| SERIAL\_NUMBER | Хранит серийный номер пользователя |
| PC\_TYPE\_NUMBERS | Хранит числовые значения типов IBM PC |
| PC\_TYPE\_NAMES | Хранит названия типов IBM PC |

**Последовательность действий, выполняемых программой.**

Программа определяет тип IBM PC, версию системы, OEM номер и серийный номер пользователя, переводит эти значения в необходимую систему счисления, сохраняет их в соответствующие переменные, а затем выводит эти значения на экран.

**Результаты**

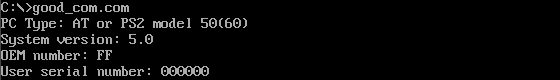


Рисунок 1 – Результат работы COM-файла.



Рисунок 2 – Результат работы «Плохого» EXE-файла.

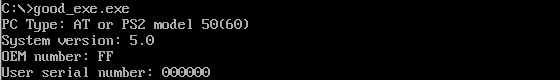


Рисунок 3 – Результат работы «Хорошего» EXE-файла.

**Описание результатов**

**Ответы на контрольные вопросы:**

**1.Отличия исходных текстов COM и EXE файлов.**

А) В COM-программе должен быть один сегмент, содержащий и код и данные.

Б) EXE-программа должна содержать три сегмента – сегмент кода, сегмент данных и сегмент стека.

В) COM-программа должна содержать следующие директивы: SEGMENT – для выделения сегмента программы, ASSUME – для установки относительных адресов с начала сегмента (при неиспользовании этой директивы DOS устанавливает значения такие, чтобы они указывали на тот сегмент, в который загружают программу), ORG – для установления относительного адреса начала выполнения программы (обычно используется ORG 100h т.к. 256 байт резервируется под PSP).

Г) В COM-программе можно использовать только команды удовлетворяющие ограничениям модели памяти TINY.

===============================================================

***Примечание [1] – Ответ:*** В com-файле все процедуры и переменные по умолчанию адресованы как NEAR объекты и поэтому FAR обращения недопустимы при использовании TINY модели памяти.

======================================================

Откроем исполняемые файлы в FAR-менеджере в шестнадцатеричном виде и сравним их:



Рисунок 4 – COM-файл



Рисунок 5 – «Плохой» EXE-файл



Рисунок 6 – «Хороший» EXE-файл

**2.Отличия форматов файлов COM и EXE модулей.**

А) COM-файл не содержит заголовка, а только код и данные программы, что позволяет ему быть более компактным, чем EXE-файл. Размер COM-файла ограничен 64 кб, т.е. размером одного сегмента. Код программы располагается с нулевого адреса.

Б) EXE-файл состоит из двух частей – заголовка и тела программы. В теле программы находятся код и данные, а в заголовке – информация о размере выполняемого модуля, области загрузки в памяти, адресе стека и относительных смещениях, которые должны заполнить машинные адреса в соответствии с относительными шестнадцатеричными позициями. Код «плохого» EXE-файла располагается с адреса 300h т.к. 200h занимает заголовок файла и а директива ORG сдвигает адрес выполнения программы еще на 100h.

======================================================

***Примечание [2] – Ответ:*** Области загрузки в памяти как таковой нет в заголовке программы, т.к. разместить в памяти программу и построить PSP для программы это задача загрузчика ОС. В заголовке есть относительные смещения от начала образа задачи, которые заменяются на конкретные с помощью информации из таблицы о размере длины файла и длины заголовка в процессе загрузки программы в память.

======================================================

В) «Хороший» EXE-файл более компактен по сравнению с плохим, т.к. отсутствует директива ORG. Также можно заметить различия в заголовочной части из-за изменения тела программы.

Отладим COM-файл и «хороший» EXE-файл с помощью TD.exe:

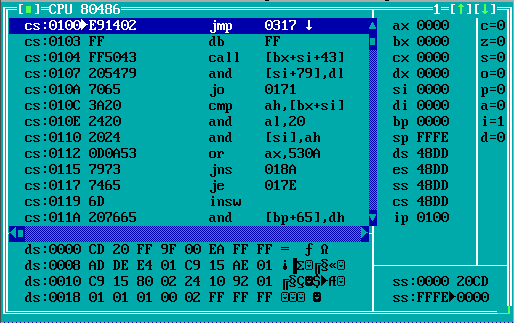


Рисунок 7 – COM-файл в отладчике

**3.Загрузка COM-модулей в основную память.**

А) Порядок загрузки COM-файла: PSP, данные и код, стек. Код начинается с адреса 100h.

Б) С нулевого адреса располагается PSP.

В) Все сегментные регистры имеют значение 48DD. Они указывают на начало PSP.

Г) Для COM-файла DOS автоматически определяет стек. Если для программы размер сегмента в 64 кб достаточен, то DOS устанавливает в регистре SP адрес конца сегмента – FFFE. Это будет вершина стека. Если в сегменте недостаточно места ля стека, то DOS устанавливает стек в конце памяти. В обоих случаях DOS записывает в стек нулевое слово.

======================================================

***Примечание [3] – Ответ:*** Расположение стека в конце сегмента команд не приводит к каким-либо неприятностям, пока размер программы далек от граничной величины 64 К. В этом случае начало сегмента команд занимают коды команд, а конец — стек. Если, однако, размер программы приближается к 64 К, то для стека остается все меньше места. При интенсивном использовании стека в программе может получиться, что по мере занесения в стек новых данных, стек дорастет до последних команд сегмента команд и начнет затирать эти команды. Очевидно, что этого нельзя допускать. В то же время система не проверяет, что происходит со стеком и никак не реагирует на затирание команд или данных. Таким образом, оценка размеров собственно программы, данных и стека является важным этапом разработки программы. SP-регистр движется от FFFE адреса в сторону уменьшения, т.е. к вершине программы, к ее коду и данным.

======================================================

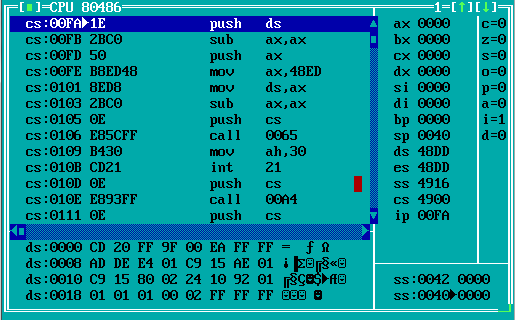


Рисунок 8 – «хороший» EXE-файл в отладчике

**4.Загрузка «хорошего» EXE-модуля в основную память.**

А) Порядок загрузки EXE-файла: PSP, сегмент кода, сегмент данных, сегмент стека. Сегментные регистры имеют значения: DS=48DD, ES=48DD, SS=4916, CS=4900.

Б) В начальный момент времени в DS=ES лежит адрес начала PSP. После выполнения команд 00FE и 0101 в DS помещается адрес сегмента данных.

В) Стек определяется директивой DW 32 DUP(?) в сегменте стека.

Г) Точка входа определяется директивой END Main, где Main – имя процедуры, с которой должна начинаться программа.

**Вывод**

В ходе данной лабораторной работы мы провели сравнение исполняемых файлов типа COM и EXE, в ходе которого выявили сходства и различия в их исходных кодах, а также в способе их загрузки в оперативную память.