**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.7383 |  | Рудоман В.А. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2019

**Цель работы:** Исследовать различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

**Функции и структуры данных программы:**

|  |  |
| --- | --- |
| **Название** | **Назначение** |
| 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\_TYPE\_OF\_PC | Определение типа IBM PC |
| GET\_VER\_OF\_SYS | Определение версии системы |
| GET\_NUM\_OF\_OEM | Определение OEM |
| GET\_SERIAL\_NUM | Определение серийного номера пользователя |

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Назначение** |
| PC\_TYPE | db | 'PC type: ',0dh,0ah,'$' |
| SYSTEM\_VERSION | db | 'System version: . ',0dh,0ah,'$' |
| OEM\_NUMBER | db | 'OEM number: ',0dh,0ah, |
| SERIAL\_NUMBER | db | 'User serial number: ',0dh,0ah,'$' |

**Действия, выполняемые программой:**

Программа определяет:

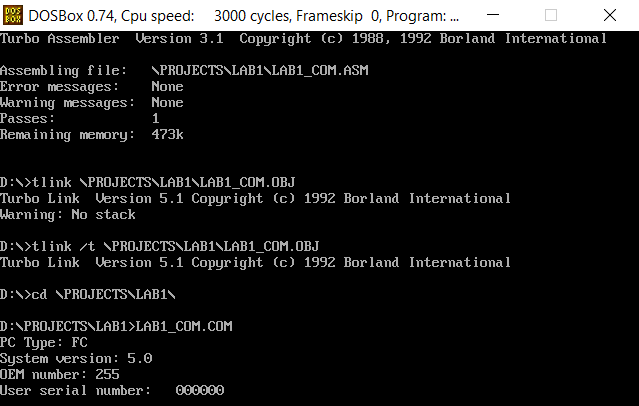
1. Тип IBM PC;
2. Версию системы;
3. OEM номер;
4. Серийный номер пользователя.

Сохраняет их в соответствующие переменные, а затем – выводит эти значения на экран при помощи функции 09H и вызова прерывания int 21h.

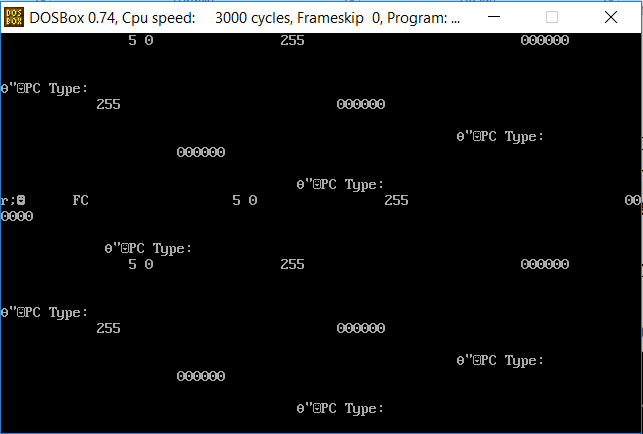
**Ход работы:**

1)Написание текста исходного LAB1\_COM.ASM файла, компиляция и компоновка LAB1\_COM.COM модуля, который определяет тип PC, версию системы, OEM и серийный номер пользователя.

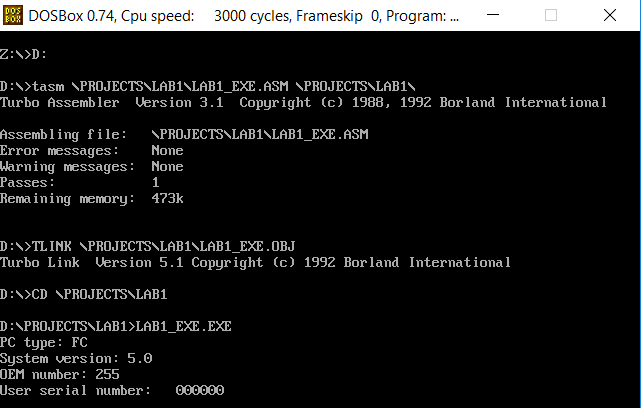
Результат работы LAB1\_COM.COM:



Так же был скомпилирован «плохой » LAB1\_COM.EXE. Результат запуска данного модуля представлен ниже:

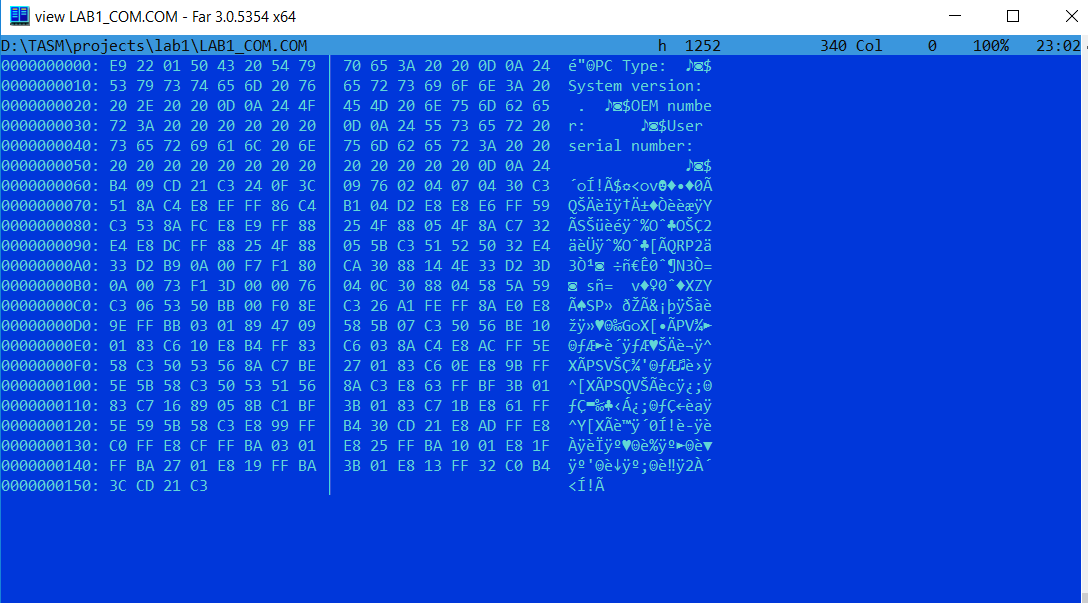


2)После этого был написан LAB1\_EXE.ASM файл, построен и отлажен. В итоге получился «хороший» LAB1\_EXE.EXE. Результат запуска данного модуля представлен ниже:

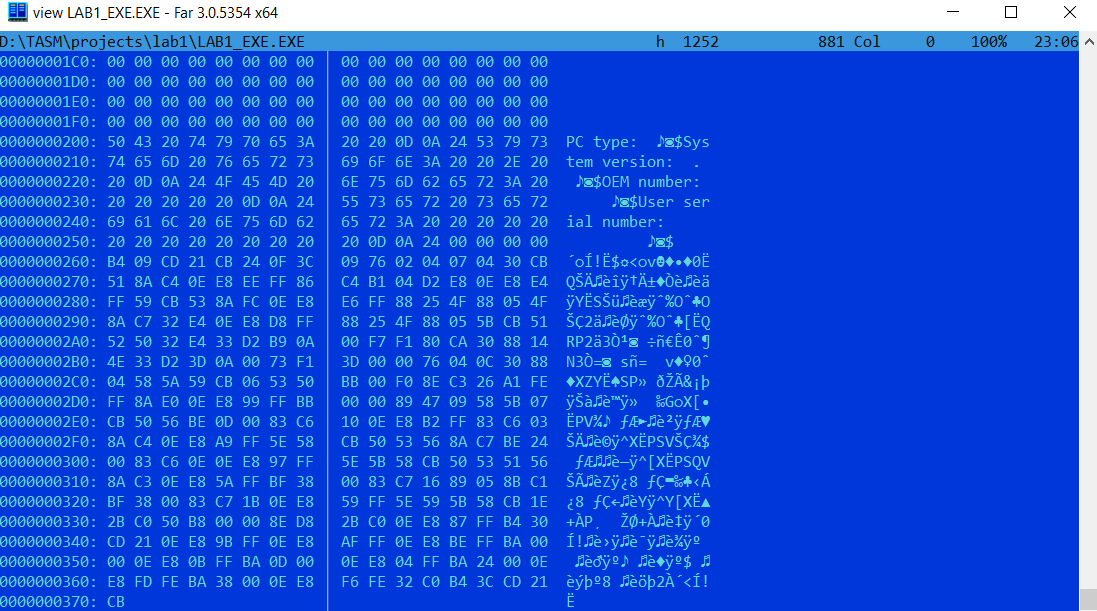


3)С помощью Far откроем все файлы вышеперечисленных модулей в шестнадцатеричном виде.

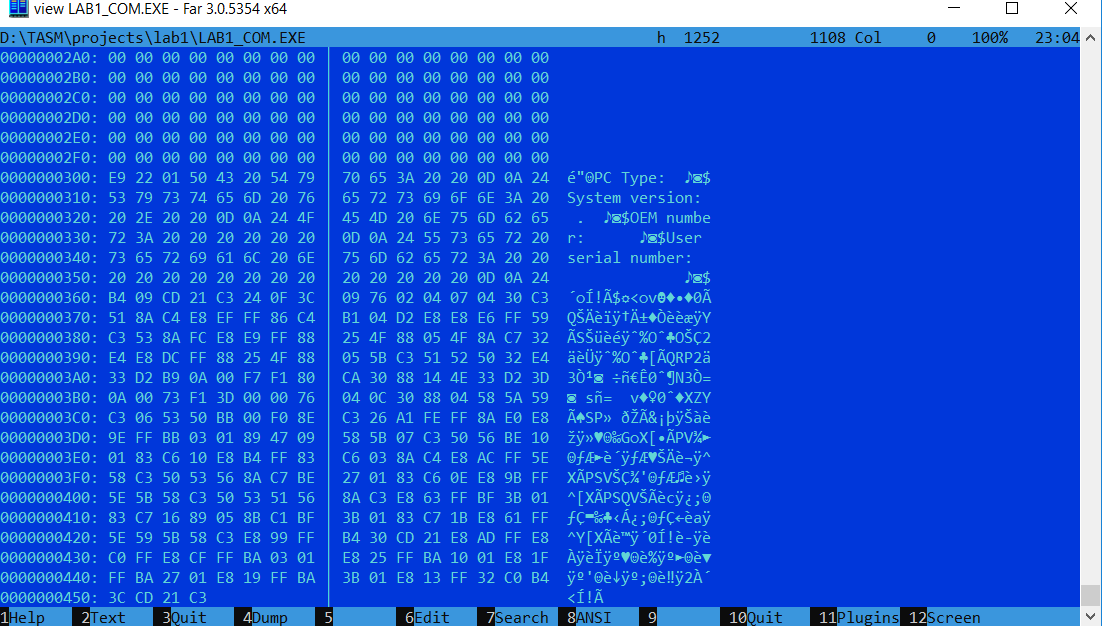
Представление LAB1\_COM.COM в шестнадцатеричном виде:



Представление «хорошего» LAB1\_EXE.EXE:

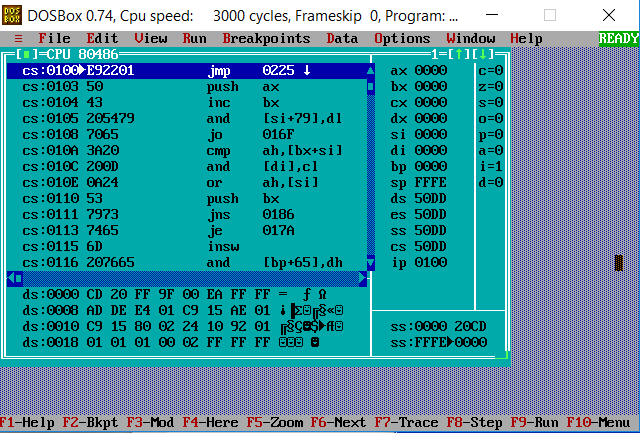


Представление «плохого» LAB1\_COM.EXE:

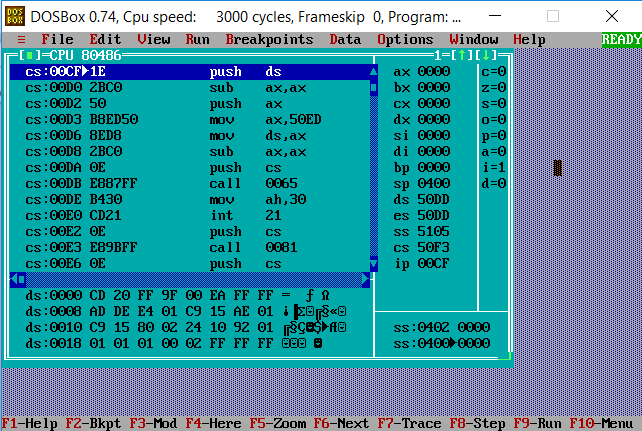


4)Открытие LAB1\_COM.COM и LAB1\_EXE.EXE в отладчике TD

LAB1\_COM.COM:

**

LAB1\_EXE.EXE:

**

**Ответы на контрольные вопросы:**

* **Отличия исходных текстов COM и EXE программ:**

1. *Сколько сегментов должна содержать COM-программа?*

COM-программа должна содержать один сегмент – сегмент кода

1. *Сколько сегментов должна содержать EXE-программа?*

EXE-программа может содержать 1 и более сегментов: сегментстека, сегмент данных и сегмент кода.

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

В тексте COM-программы обязательно следующие директивы:

1. ASSUME, которая должна указывать на то, что сегмент кода и данных начинается с одного и того же места.
2. ORG, которая устанавливает счётчик положения в сегменте равным заданной величине, которая передаётся как параметр.
3. *Все ли форматы команд можно использовать в COM-программе?*

В COM-программе все сегментные регистры определяются в момент запуска программы, а не в момент компиляции (ассемблирования), поэтому в ней нельзя использовать команды вида mov<регистр>, seg<имя сегмента>.

В COM файле нет relocation table (таблицы настройки), поэтому он не может получить информацию об адресе сегмента. Следовательно нельзя использовать команды, которые используют адрес сегмента и дальнюю адресацию.

* **Отличия форматов файлов COM и EXE модулей:**

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

COM-файл содержит данные и машинные команды. Структура очень компактна (сам файл весит гораздо меньше). Код начинается с адреса 0h (но при загрузке модуля устанавливается смещение в 100h).

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

В «плохом» EXE код и данные не разделены по сегментам, а перемешаны. Код располагается с адреса 300h, так как заголовок занимает 200h байт, а команда ORG 100h сдвигает код еще на 100h. С нулевого адреса располагается заголовок. В первых двух байтах можно увидеть символы MZ, означающие, что формат файла – 16-битный и его следует запускать в соответствии со структурой .EXE файлов. После заголовка идет таблица

настройки. Без которой, файл загружался бы в память как .COM файл.

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

В «хорошем» EXE данные, стек и код разделены по сегментам. Структура несколько компактнее, чем структура «плохого» файла, так как в этом файле отсутствует директива ORG 100h, резервирующая пространство для заголовка. Из-за этого код располагается с адреса 200h, а не с 300h, как в «плохом» EXE-файле.

* **Загрузка COM-модуля в основную память:**

1. *Какой формат загрузки модуля СОМ? С какого адреса располагается код?*
2. *Определяется сегментный адрес участка ОП с достаточным местом для загрузки программы*
3. *Создается блок памяти для PSP и программы;*
4. *COM файл начинает свою загрузку с адреса 100h;*
5. *Сегментные регистры СS, DS, ES, SS устанавливаются на начало PSP(oh), а регистр SP на конец PSP;*
6. *В стек записывается – 0000, в регистр IP – 100h.*

Порядок загрузки модуля COM: PSP, данные и код, стек. Код начинается с адреса 100h.

1. *Что располагается с адреса 0?*

С нулевого адреса располагается PSP.

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

Все сегментные регистры имеют значение «50DD» и указывают на начало PSP.

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

Стек занимает всё свободное пространство до конца сегмента памяти (размер .COM файла не может превышать 64 кб), которое осталось после загрузки данных и кода. В данном случае значение регистра SP=FFFE.

* **Загрузка “хорошего” EXE-модуля в основную память:**

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

Порядок загрузки EXE-модуля:

1. PSP;
2. Сегмент кода;
3. Сегмент данных;
4. Сегмент стека.

Сегментные регистры на момент загрузки программы имеют значения: ES=50DD, CS=50F3, DS=50DD, SS=5105.

В начале выполнения программы, значения ES и DS совпадают, так как не были выполнены команды “mov ax, data; mov ds, ax”, т.е. в регистр данных не был занесен адрес сегмента данных.

1. *Б) На что указывают регистры DS и ES?*

ES - на начало PSP;

DS - на начало данных.

После выполнения команд (“mov ax, data; mov ds, ax”), значение DS= 50ED.

1. *Как определяется стек?*

Стек определяется c помощью директивы «DW 512 DUP(?)» в описании сегмента стека.

1. *Как определяется точка входа?*

Точка входа определяется директивой END, после которой следует адрес, куда переходит программа при запуске.

**Вывод:** В ходе лабораторной работы было проведено сравнение текстов COM и EXE программ, их структуры, определены отличия файлов COM и EXE модулей, а так же исследованы способы загрузки их в память.