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

ОТЧЕТ

по лабораторной работе № 1

по дисциплине «Операционные системы»

Тема: Исследование структур загрузочных модулей

Студент гр. 9382 _____ Герасев Г.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

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

Постановка задачи.

Требуется написать текст исходного .COM модуля, который определяет тип PC и версию системы. Программа делает это читая содержимое предпоследнего байта ROM BIOS, по таблице, сравнивая коды, тем самым определяя тип PC и выводит 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.

Таблица 1 – Процедуры в программе

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа
BYTE_TO_HEX	Перевод байта в 16-ной с/с в символьный код
WRD_TO_HEX	Перевод слова в 16-ной с/с в символьный код
BYTE_TO_DEC	Перевод байта в 16-ной с/с в символьный код в 10-ной с/с
PRINT	Вывод строки на экран
PC_TYPE	Определение типа PC
OS_VER	Определение характеристик OS

Выполнение работы.

Были объявлены строки для вывода информации:

- TYPE_PC db 'Type: PC',0DH,0AH,'$';
- TYPE_PC_XT db 'Type: PC/XT',0DH,0AH,'$';
- TYPE_AT db 'Type: AT',0DH,0AH,'$';
- TYPE_PS2_M30 db 'Type: PS2 модель 30',0DH,0AH,'$';
- TYPE_PS2_M50_60 db 'Type: PS2 модель 50 или 60',0DH,0AH,'$';
- TYPE_PS2_M80 db 'Type: PS2 модель 80',0DH,0AH,'$';
- TYPE_PC_JR db 'Type: PCjr',0DH,0AH,'$';
- TYPE_PC_CONV db 'Type: PC Convertible',0DH,0AH,'$';
- VERSIONS db 'Version MS-DOS: . ',0DH,0AH,'$';
- SERIAL_NUMBER db 'Serial number OEM: ',0DH,0AH,'$';

- USER_NUMBER db 'User serial number: H $'.

Были составлены функция для определения типа ПК PC_TYPE в соответствии с таблицей:

Модель	Значение
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

А также функция для определения характеристик ОС OS_VER:

- номер основной версии системы и её модификации;
- номер OEM;
- серийный номер пользователя.

В результате выполнения были получены следующие значения(рис.1-3):

Рисунок 1 – «хороший» .COM модуль

Рисунок 2 – «плохой» .EXE модуль

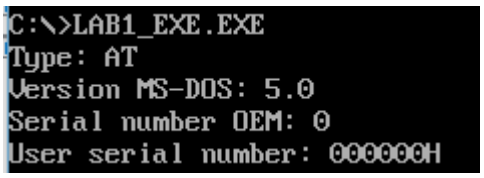

Рисунок 3 – «хороший» .EXE модуль

Выводы.

В ходе лабораторной работы были исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

ПРИЛОЖЕНИЕ А
ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

Отличия исходных текстов COM и EXE программ:

1. Сколько сегментов должна сдержать COM-программа?

COM-программа должна содержит ровно один сегмент. Код и данные находятся в одном сегменте, стек генерируется автоматически.

2. EXE-программа?

EXE-программа должна содержит не менее одного сегмента. Сегменты кода, данных и стека описываются отдельно друг от друга. Можно не описывать сегмент стека, в таком случае будет использоваться стек созданый по умолчанию.

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

Обязательна директива ORG 100h, так как требуется учесть смещение автоматически созданного сегмента, поэтому адресация имеет смещение в 256 байт от нулевого адреса. Также необходима ASSUME для того, чтобы сегмент данных и сегмент кода указывали на один общий сегмент. (ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING)

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

Нельзя использовать команды вида mov, из-за отсутствия таблицы настроек.

Отличия форматов файлов .COM и.EXE программ:

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

COM-файл состоит из одного сегмента, состоящего из сегмент кода и сегмент данных, сегмент стека генерируется автоматически при создании COM-программы. COM-файл ограничен размером одного сегмента и не

превышает
64 Кб

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 адреса 0?

Данные и код располагаются в одном сегменте, что для EXE файла не допустимо, так как код и данные должны быть разделены на отдельные сегменты. Код располагается с адреса 300h, а с адреса 0h идёт таблица настроек.

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

В EXE-программе код, данные и стек поделены на сегменты. Программа в формате EXE может иметь любой размер. EXE-файл имеет заголовок, который используется при его загрузке. Заголовок состоит из форматированной части, содержащей сигнатуру и данные, необходимые для загрузки EXE-файла, и таблицы для настройки адресов. В отличии от «плохого» EXE в «хорошем» EXE присутствуют три сегмента: сегмент кода, сегмент данных и сегмент стека, а «плохой» EXE содержит один сегмент, совмещающий код и данные. Также в «плохом» EXE адресация кода начинается с 300h, так как он получается из .COM файла, в котором изначально сегмент кода смещён на 100h, а при создании «плохого» EXE к этому смещению добавляется размер PSP модуля(200h). А в «хорошем» EXE присутствует только смещение для PSP модуля, поэтому код начинается с 200h. В данной случае смещение кода 400h так как выделяется память под стек (200h), память под стек находится между PSP и кодом.

Загрузка COM модуля в основную память:

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

Определяется сегментный адрес участка ОП, у которого достаточно места для загрузки программы, образ COM-файла считывается с диска и помещается в память, начиная с PSP:0100h. После загрузки двоичного образа COM-программы сегментные регистры CS, DS, ES и SS указывают на PSP(в данном случае сегментные регистры указывают на 48DD), SP указывает на конец сегмента PSP(обычно FFFE), слово 00H помещено в стек, IP содержит 100H в результате команды JMP PSP:100H.

2. Что располагается с адреса 0?

Программный сегмент PSP, размером 256 байт (100h), зарезервируемый операционной системой.

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

Сегментные регистры CS, DS, ES и SS указывают на PSP и имеют значения 48DD.

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

Стек генерируется автоматически при создании COM-программы. SS – на начало (0h), регистр SP указывает на конец стека (FFFEh), Адреса стека расположены в диапазоне 0h – FFFEh (FFFEh, – последний адрес, кратный двум).

Загрузка «хорошего» EXE модуля в основную память:

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

EXE-файл загружается, начиная с адреса PSP:0100h. В процессе загрузки считывается информация заголовка (PSP) EXE в начале файла и выполняется перемещение адресов сегментов, то есть DS и ES устанавливаются на начало сегмента PSP(DS=ES=48DD), SS(SS=48ED) – на начало сегмента стека, CS(CS=490D) – на начало сегмента команд. В IP загружается смещение точки входа в программу, которая берётся из метки после директивы END. Причём дополнительный программный сегмент (PSP) присутствует в каждом EXE-файле.

2. На что указывают регистры DS и ES?

Регистры DS и ES указывают на начало сегмента PSP.

3. Как определяется стек?

Стек определяется с помощью директивы .stack, после которой задаётся размер стека. При исполнение регистр SS указывает на начало сегмента стека, а SP на конца стека(его смещение).

4. Как определяется точка входа?

Точка входа определяется при помощи директивы END.