

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

ОТЧЕТ
по лабораторной работе №7
по дисциплине «Операционные системы»
Тема: Построение модуля оверлейной структуры

Студент гр. 7381

Павлов А.П.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследование возможности построение загрузочного модуля оверлейной структуры. Исследуется структура оверлейного сегмента и способ загрузки и выполнения оверлейных сегментов. Для запуска вызываемого оверлейного модуля используется функция 4B03h прерывания int 21h. Все загруженные и оверлейные модули находятся в одном каталоге.

В этой работе также рассматривается приложение, состоящее из нескольких модулей, поэтому все модули помещаются в один каталог и вызываются с использованием полного пути.

Необходимые сведения для составления программы.

Для организации программы, имеющей оверлейную структуру, используется функция 4B03h прерывания 21h. Эта функция позволяет в отведённую область памяти, начинающуюся с адреса сегмента, загрузить программу, находящуюся в файле на диске. Передача управления загруженной программе этой функцией не осуществляется и префикс сегмента программы (PSP) не создаётся.

Если флаг переноса $CF = 1$ после выполнения функции, то произошли ошибки и регистр AX содержит код ошибки. Значение регистра AX характеризует следующие ситуации, представленные в табл. 1.

Таблица 1 – Возможные ошибки при выполнении функции 4B03h.

Код ошибки	Описание
1	Несуществующая функция
2	Файл не найден
3	Маршрут не найден
4	Слишком много открытых файлов
5	Нет доступа
8	Мало памяти
10	Неправильная среда

Если флаг переноса $CF = 0$, то оверлей загружен в память.

Перед загрузкой оверлея вызывающая программа должна освободить память по функции 4Ah прерывания 21h. Затем определить размер оверлея. Это можно сделать с помощью функции 4Eh прерывания 21h. Перед обращением к функции необходимо определить область памяти размером в 43 байта под буфер DTA, которую функция заполнит, если файл будет найден.

Функция использует следующие параметры: *CX* – значение байта атрибутов, которое для файла имеет значение 0; *DS:DX* – указатель на путь к файлу, который записывается в формате строки ASCIIZ.

Если флаг переноса *CF* = 1 после выполнения функции, то произошли ошибки и регистр *AX* содержит код ошибки. Значение регистра *AX* характеризует ситуации, представленные в табл. 2.

Таблица 2 – Возможные ошибки при выполнении функции 4Eh.

Код ошибки	Описание
2	Файл не найден
3	Маршрут не найден

Если *CF* = 0, то в области памяти буфера DTA со смещением 1Ah будет находиться младшее слово размера файла, а в слове со смещением 1Ch – старшее слово размера памяти в байтах.

Полученный размер файла следует перевести в параграфы, причём следует взять большое целое числа параграфов. Затем необходимо отвести память с помощью функции 48h прерывания 21h. После этого необходимо сформировать параметры для функции 4B03h и выполнить её.

После отработки оверлея необходимо освободить память с помощью функции 49h прерывания 21h.

Оверлейный сегмент не является загрузочным модулем типов .COM или .EXE. Он представляет собой кодовый сегмент, который оформляется в ассемблере как функция с точкой входа по адресу 0 и возврат осуществляется командой retf. Это необходимо сделать, потому что возврат управления должен быть осуществлён в программу, выполняющую оверлейный сегмент. Если

использовать функции выхода 4Ch прерывания 21h, то программа закончит свою работу.

Описание функций и структур данных.

Таблица 1 – функции управляющей программы.

Название функции	Назначение
PRINT	Печатает строку на экран.
FreeMemory	Освобождает память, занимаемую вызывающий программой.
BYTE_TO_HEX	Переводит число AL в коды символов 16 с/с, записывая получившиеся в AL и AH.
TETR_TO_HEX	Вспомогательная функция для работы BYTE_TO_HEX
MAIN	Основная функция программы.
FIND_PATH	Ищет путь до оверлея
FIND_OVL_SIZE	Считывает размера файла оверлея и запрашивает нужный для загрузки объем памяти
CALL_OVL	Вызывает оверлейной программу и после освобождает память.

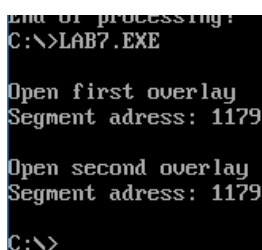
Ход работы.

Была написана программа, которая выполняет следующие действия:

1. Освобождение памяти для загрузки оверлеев.
2. Чтение размера файла оверлея и выделение памяти, достаточной для его загрузки.
3. Загрузка и выполнение оверлейного сегмента.
4. Освобождение памяти, отведённой для оверлейного сегмента.
5. Повторение пунктов 1-5 для второго оверлейного сегмента.

Результат работы.

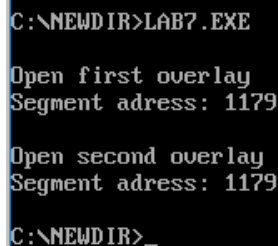
Была запущена программа, когда оба оверлейных модуля находятся в текущем каталоге. Результат работы программы представлен на рис. 1.



```
End of processing!  
C:\>LAB7.EXE  
  
Open first overlay  
Segment address: 1179  
  
Open second overlay  
Segment address: 1179  
  
C:\>
```

Рисунок 1 – результат работы программы, оба оверлейных модуля в текущем каталоге.

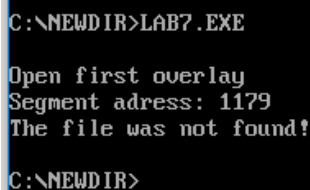
Была запущена программа, когда оба модуля находятся не в текущем каталоге. Результат работы программы представлен на рис. 2.



```
C:\NEWDIR>LAB7.EXE  
  
Open first overlay  
Segment address: 1179  
  
Open second overlay  
Segment address: 1179  
  
C:\NEWDIR>
```

Рисунок 2 – результат работы программы, оба модуля не в текущем каталоге.

Была запущена программа, когда один оверлей находится не в текущем каталоге. Результат работы программы представлен на рис. 5.



```
C:\NEWDIR>LAB7.EXE  
  
Open first overlay  
Segment address: 1179  
The file was not found!  
  
C:\NEWDIR>
```

Рисунок 5 – результат работы программы, один оверлей находится не в текущем каталоге.

Выводы.

В процессе выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры.

Ответы на контрольные вопросы.

1. Как должна быть устроена программа, если в качестве оверлейного сегмента использовать .COM модули?

Ответ: в начале выделенной памяти необходимо поместить PSP и увеличить смещение оверлейного сегмента на 256 байт, так как PSP запускаемо оверлея при таком запуске сформирован не будет.