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

ОТЧЕТ

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

Студент гр. 0382 _____ Азаров М.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Исследование возможности построения 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 загрузочные 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.

Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет функции:

1) Освобождает память для загрузки оверлеев.
2) Читает размер файла оверлея и запрашивает объем памяти, достаточный для его загрузки.
3) Файл оверлейного сегмента загружается и выполняется.
4) Освобождается память, отведенная для оверлейного сегмента.
5) Затем действия 1)-4) выполняются для следующего оверлейного сегмента.

Шаг 2. Также необходимо написать и отладить оверлейные сегменты. Оверлейный сегмент выводит адрес сегмента, в который он загружен.

Шаг 3. Запустите отлаженное приложение. Оверлейные сегменты должны загружаться с одного и того же адреса, перекрывая друг друга.

Шаг 4. Запустите приложение из другого каталога. Приложение должно быть выполнено успешно.

Шаг 5. Запустите приложение в случае, когда одного оверлея нет в каталоге. Приложение должно закончиться аварийно.

Шаг 6. Занесите полученные результаты в виде скриншотов в отчет. Оформите отчет в соответствии с требованиями.

Ход работы

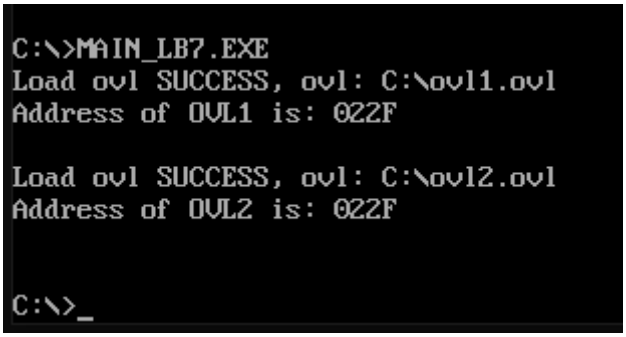
1. Для выполнения первого шага понадобилось разработать модуль типа .EXE , который содержит следующие процедуры:

Название процедуры	Что делает
`Main`	Главная процедура выполняющая поставленную задачу.
`Free_mem`	Освобождает память, которую не использует программа с помощью процедуры `Malloc` и обрабатывает результат этой операции.
`Malloc`	Процедура которая непосредственно освобождает память, используется в процедуре `Free_mem`
`Set_file_path`	Определяет путь к файлу запущенной программы и меняет название конечного файла на заданное имя в BX. Результат заносится в file_path.
`Request_mem_ovl`	Зная путь до файла оверлея (из переменой file_path , кот. определяется проц. `Set_file_path`) вычисляет размер файла и запрашивает для него память отдельным сегментом. Подготавливает PARAM_BLOCK для дальнейшей загрузки оверлея.
`Load_ovl`	Загружает оверлей зная путь к файлу оверлея file_path и PARAM_BLOCK. Запускает оверлей и после завершения очищает память выделенную под оверлей.
`Handle_er_load_ovl`	Обработка ошибок при загрузке оверлея.

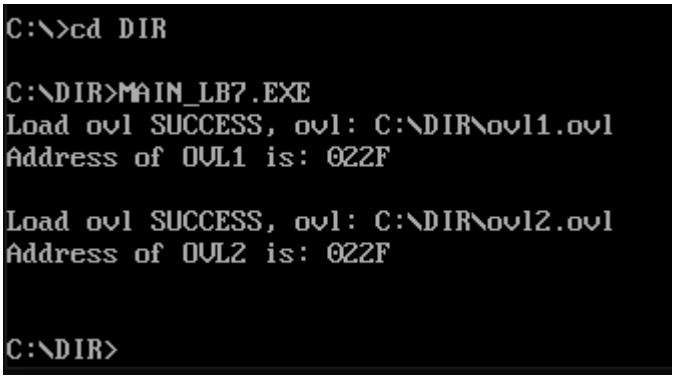
2. Были написаны два оверлейных сегмента ovl1.asm и avl2.asm, которые идентичны по содержанию. Их задача заключается в выводе адрес сегмента, в который они были загружены. Реализуется это с помощью процедуры

WRD_TO_HEX , взятой из методички. Затем они были скомпилированы как ovl1.ovl и ovl2.ovl.

3. Запустив модуль видим, что оверлейные сегменты были успешно запущены , причем с одного адреса, перекрывая друг друга:

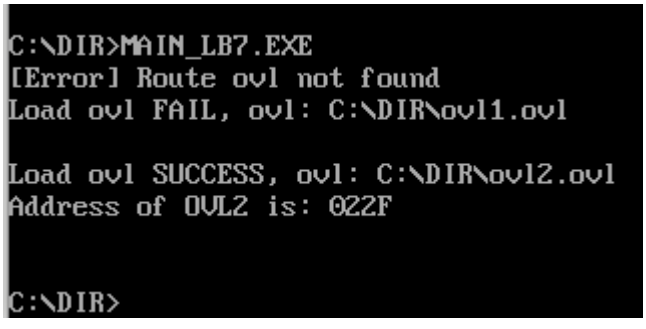


4. Попробуем запустить модуль из другого каталога. Теперь в каталоге «Dir» находится главный модуль и оверлейные сегменты. После запуска наблюдаем:



Как видим это никак не повлияло на корректность работы модуля.

5. Теперь попробуем запустить программу когда одного оверлея нет в каталоге:



Как видим в таком случае модуль не может найти один оверлейный сегмент и выводит соответствующее сообщение. Тем не менее второй оверлей был найден и успешно обработан.

Ответы на контрольные вопросы.

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

Ответ: Так как в .COM файле первые 100h выделяются под PSP, то точку входа в оверлей нужно будет сместит на 100h от начала сегмента. Также нужно будет сохранить регистры и восстанавливать их в конце работы оверелея.

Вывод.

В ходе выполнения лабораторной работы была изучена возможность построения загрузочного модуля оверлейной структуры. Также был исследован интерфейс между модулем вызывающим оверлей и самим оверлеем. Была разработана программа, состоящая из нескольких модулей, которая поочерёдно загружала и выполняла оверлейные сегменты.