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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Операционные сети»

Тема: «Исследование организации управления основной памятью»

Студентка гр. 0381 Сарычева А.А.

Преподаватель Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованный в компьютере и способ организации, принятый в ОС. В лабораторной работе рассматривается не страничная память и способ управления динамическими разделами. Для реализации управления памятью в этом случае строится список занятых и свободных участков памяти. Функции ядра, обеспечивающие управление основной памятью, просматривают и преобразуют этот список.

В лабораторной работе исследуются структуры данных и работа функций управления памятью ядра операционной системы.

Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

1) Количество доступной памяти.

2) Размер расширенной памяти.

3) Выводит цепочку блоков управления памятью.

Адреса при выводе представляются шестнадцатеричными числами. Объем памяти функциями управления памятью выводится в параграфах. Необходимо преобразовать его в байты и выводить в виде десятичных чисел. Последние восемь байт МСВ выводятся как символы, не следует преобразовывать их в шестнадцатеричные числа.

Запустите программу и внимательно оцените результаты. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 2. Измените программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah прерывания 21h (пример в разделе «Использование функции 4AH»). Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущем шаге. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 3. Измените программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H. Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущих шагах. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 4. Измените первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти. Обязательно обрабатывайте завершение функций ядра, проверяя флаг CF. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Выполнение работы.

Шаг 1. В файле lb3_1.asm написан исходный код .COM модуля. В начале данного модуля прописаны строки для вывода запрашиваемой информации, для вывода которых была создана процедура OUTPUT. Кроме того, были использованы процедуры TETR_TO_HEX, BYTE_TO_HEX, WRD_TO_HEX из предоставленного шаблона, а также для удобства организации кода были созданы процедуры OUTPUT_SYMBOL для посимвольного вывода строк в терминал, OUTPUT_ENTER для вывода переноса строки, OUTPUT_INT для вывода числа в десятичной системе исчисления.

Написана процедура OUTPUT_AVAILABLE_MEMORY, которая выводит количество доступной памяти в байтах. Для нахождения данного значения вызывается функция 48h прерывания 21h с заведомо невозможным размером выделения памяти, в результате чего получаем необходимое значение в регистре ax в параграфах, далее вызываем процедуру OUTINT, которая выведет данное значение в байтах в виде десятичного числа. Затем вызываются команды для вывода формата значения и процедура OUTPUT_ENTER.

Затем была написана процедура OUTPUT_MEMORYSIZE, которая выводит размер расширенной памяти. Для этого происходит обращение к ячейкам 30h и 31h памяти CMOS. В результате, которого в регистре ax хранится

требуемое значение в Кбайтах, далее вызываем процедуру OUTINT, которая выведет данное значение в байтах в виде десятичного числа. Затем вызываются команды для вывода формата значения и процедура OUTPUT_ENTER.

Следующая процедура OUTPUT_MSB выводит цепочку блоков управления памятью. Для каждого блока выводится его порядковый номер, тип, адрес самого блока, сегментный адрес PSP либо один из кодов информации, а также размер участка в байтах и системные данные. Для этого с помощью функции 52h прерывания 21h получаем адрес самого первого блока, от которого и начинаем выводить информацию до тех пор, пока тип блока не покажет, что данный элемент последний.

Результат работы программы:

Выводится размер доступной памяти равный 64 байтам, т.к. программа занимает весь доступный объем памяти, однако есть свободный участок памяти по сегментному адресу 0171h размером 64 байта. Размер расширенной памяти – 585728 байт.

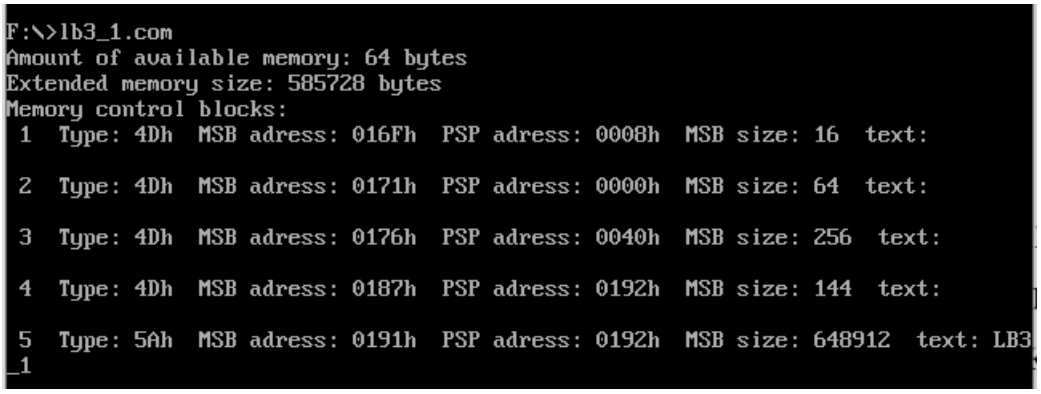

Рисунок 1 – Вывод модуля lb3_1.com

Шаг 2. В файле lb3_2.asm написан модифицированный код исходного .COM модуля, в котором теперь освобождается неиспользуемая память. В результате чего объем свободной памяти стал больше и равен 583360 байтам. Блок отведённый нашей программе занимает теперь меньшее количество памяти, поэтому появляется свободный блок размером 583360 байт.

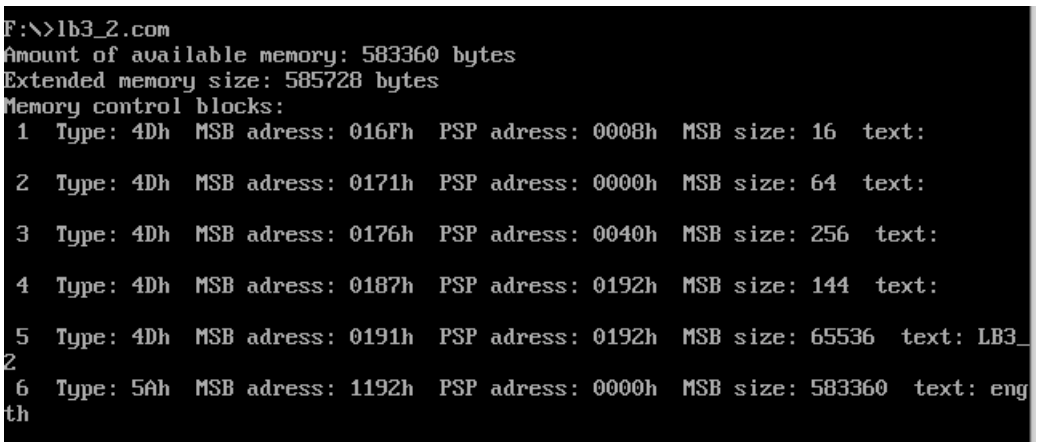

Рисунок 2 – Вывод модуля lb3_2.com

Шаг 3. В файле lb3_3.asm теперь после освобождения памяти, программа запрашивает еще 64 Кб памяти. В результате чего появился еще один блок, объем свободной памяти остается прежним и теперь равен сумме размеров последних двух блоков.

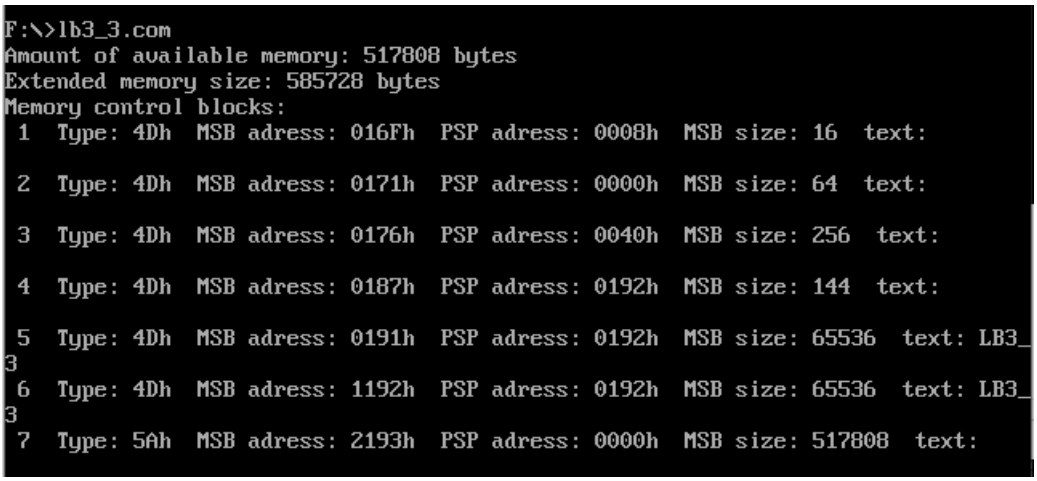

Рисунок 3 – Вывод модуля lb3_3.com

Шаг 4. В файле lb4_4.asm происходит запрос на выделение 64Кб памяти без предварительного ее освобождения. В результате чего функция 48h int 21h устанавливает флаг CF и выделения не происходит. А в результате его проверки выводится сообщение об невозможности данного действия, а все остальные данные совпадают с данными из пункта 1.

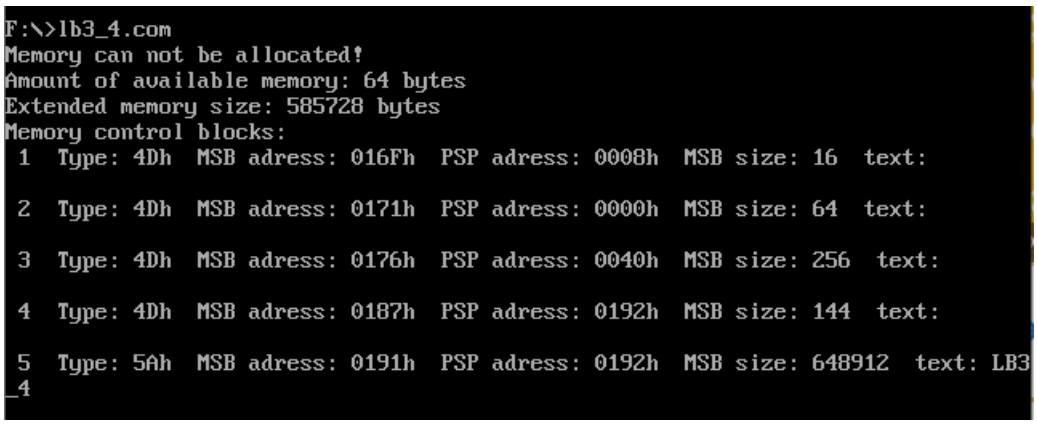

Рисунок 4 – Вывод модуля lb3_3.com

Выводы.

Была исследована структура данных и работа функций управления памятью ядра операционной системы.

Ответы на контрольные вопросы.

1) Что означает "доступный объем памяти"?

Доступный объем памяти – это свободный участок памяти максимального размера, который может запросить программа (может состоять из нескольких подряд идущих блоков).

2) Где MCB блок Вашей программы в списке?

В первом случае это последний 5 блок.

Во втором случае блок программы предпоследний и имеет порядковый номер 5.

В третьем случае программе принадлежат 5 и 6 блок памяти.

В четвертом случае аналогично первому программе принадлежит 5 блок.

3) Какой размер памяти занимает программа в каждом случае?

В первом случае программа занимает все возможное пространство равное 648912 байтам.

Во втором случае программа занимает пространство равное 65536 байтам.

В третьем случае программа занимает пространство размером 131072 байт.

В четвертый случай аналогично первому - 648912 байтам.