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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Операционные системы»

ТЕМА: ИССЛЕДОВАНИЕ ОРГАНИЗАЦИИ УПРАВЛЕНИЯ ОСНОВНОЙ ПАМЯТЬЮ

Студент(ка) гр. 0381 Ионина К.С.

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 нестраничная 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 MCB 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.

Шаг 5. Оцените результаты, полученные на предыдущих шагах. Ответьте на контрольные вопросы и оформите отчет.

Исходные данные.

Для выполнения работы были использованы данные из таблицы «Структура MCB»:

Смещение	Длина поля (байт)	Содержимое поля
00h	1	Тип MCB: 5Ah, если последний в списке; 4Dh, если не последний
01h	2	Сегментный адрес PSP владельца участка памяти,либо 0000h - свободный участок, 0006h - участок принадлежит драйверуOS XMS UMB 0007h - участок является исключенной

		верхнейпамятью драйверов 0008h - участок принадлежит MS DOS FFFAh - участок занят управля-ющим блоком386MAX UMB FFFDh - участок заблокирован 386MAX FFFEh - участок принадле-жит 386MAX UMB
03h	2	Размер участка в параграфах
05h	3	Зарезервирован
08h	8	"SC" - если участок принадлежит MS DOS, то внем системный код "SD" - если участок принадлежит MS DOS, то внем системные данные

Выполнение работы.

Шаг 1.

Сначала были объявлены строки для вывода информации:

```
AVAILABLE_MEMORY db 0Dh,0Ah,'Size of available memory:       $'
EXTENDED_MEMORY db 0Dh,0Ah,'Size of extended memory:   $'
BYTES_STR db ' byte $'
MCB_TYPE db 'MCB:   h  $'
ADRESS db 'Adress:     h  $'
OWNER_UNKNOWN db 'Owner:     h   $'
OWNER_FREE db 'Owner: Free    $'
OWNER_OSXMS db 'Owner: OS XMS UMD  $'
OWNER_DRIVER db 'Owner: High driver memory  $'
OWNER_MSDOS db 'Owner: MS DOS  $'
OWNER_BLOCKUMB db 'Owner: Occupied by 386MAX UMB  $'
OWNER_BLOCKED386 db 'Owner: Blocked by 386MAX  $'
OWNER_OWNED386MAX db 'Owner: 386MAX UMB  $'
```

AREA_SIZE db 'Area size: $'

MSB_SCSD db 'SC/SD: $'

NEWLINE db 0Dh,0Ah,'$'

Далее была реализована процедура AVAILABLE_MEM, которая выводит на экран размер доступной памяти в байтах, используя функцию 48h прерывания 21h. При вызове процедуры CONV происходит перевод параграфов в байты и полученное значение выводится на экран.

В процедуре CONV регистр bx хранит основание системы счисления, cx – количество цифр числа. Принцип работы процедуры следующий: число делится на основание системы счисления, остаток от деления сохраняется в стек, cx увеличивается, далее от частного отделяется каждая цифра справа до тех пор, пока не останется ноль. В тот момент, когда процедура получила ноль и слева тоже только нули, во втором регистре тоже ноль, из стека извлекается очередная цифра и переводится в символ, далее происходит вывод на экран. Цикл продолжается столько раз, сколько цифр подсчитано в cx.

Процедура EXTENDED_MEM выводит объем расширенной памяти. Происходит обращение к ячейкам памяти CMOS 30h и 31h, далее с помощью процедуры CONV, программа переводит значение в байты и выводит значение на экран.

Процедура MCB_CHAIN выводит цепочку блоков управления памятью, используя функции 52h, int21h программа получает адрес самого первого сегмента MCB. На экран выводится содержимое каждого блока, считывание цепочки завершается, когда программа дошла до последнего блока в списке.

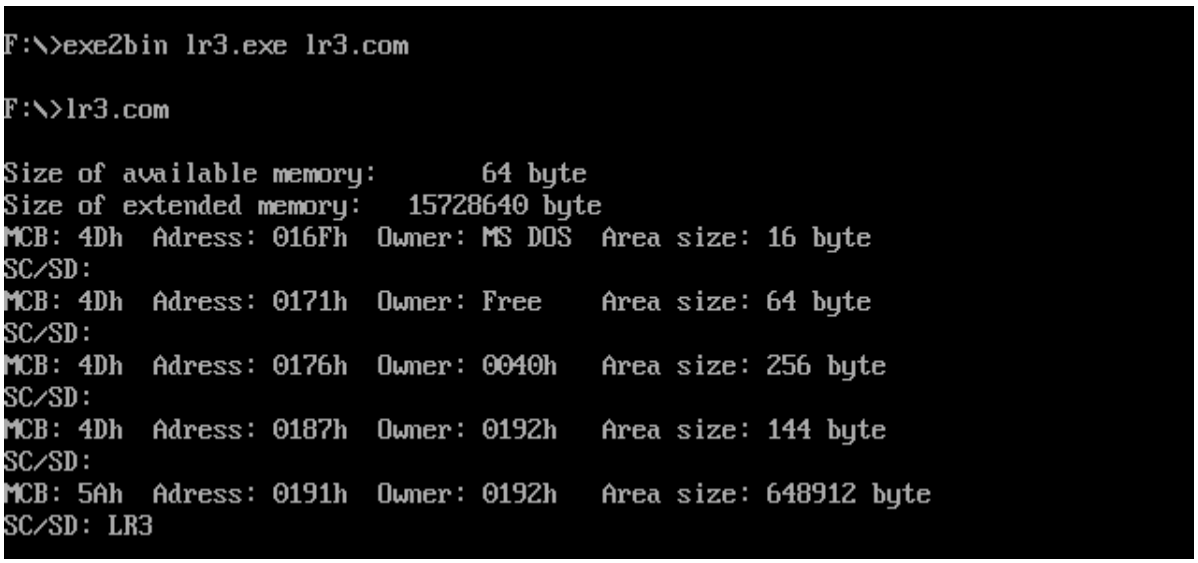

Рисунок 1. Результат работы программы. Шаг 1.

Шаг 2.

Код был модифицирован таким образом, что программа освобождает память, которую не использует. Следовательно, объем свободной памяти увеличивается, а в выводе цепочки блоков управления памятью появляется новая строка – свободный блок.

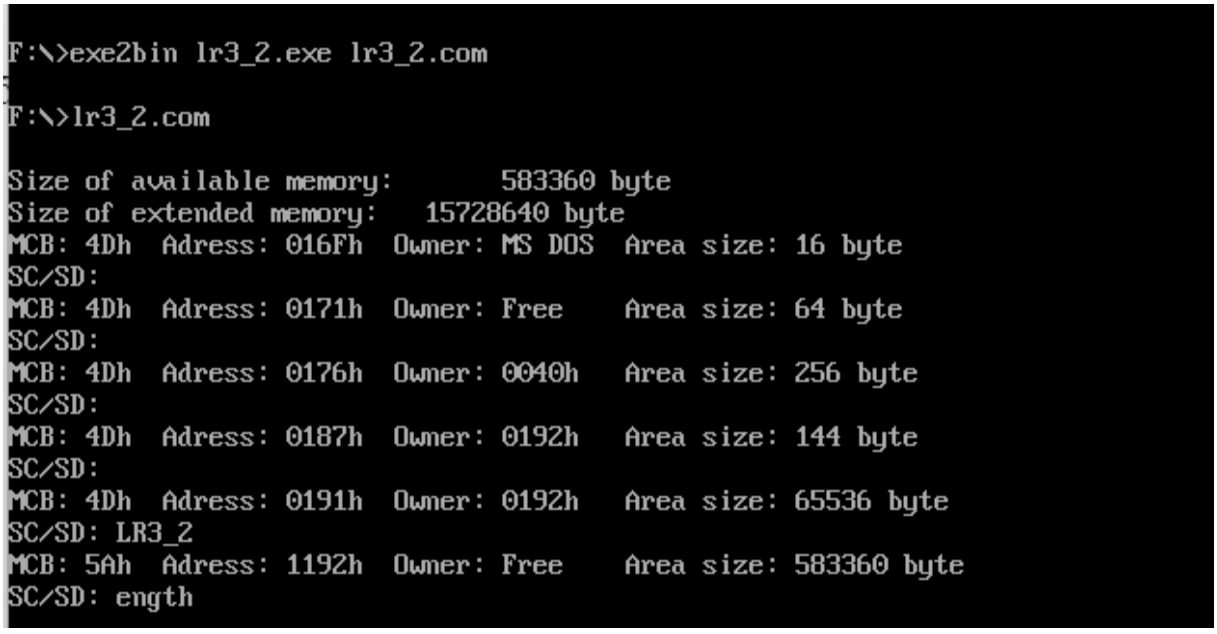

Рисунок 2. Результат работы программы. Шаг 2.

Шаг 3.

Код был изменен так, чтобы после освобождения памяти, программа запрашивала ещё 64 Кб памяти. В выводе цепочки блоков управления памятью

появляется ещё один блок размером 64 Кб. Объём свободной памяти не изменился по сравнению со вторым шагом.

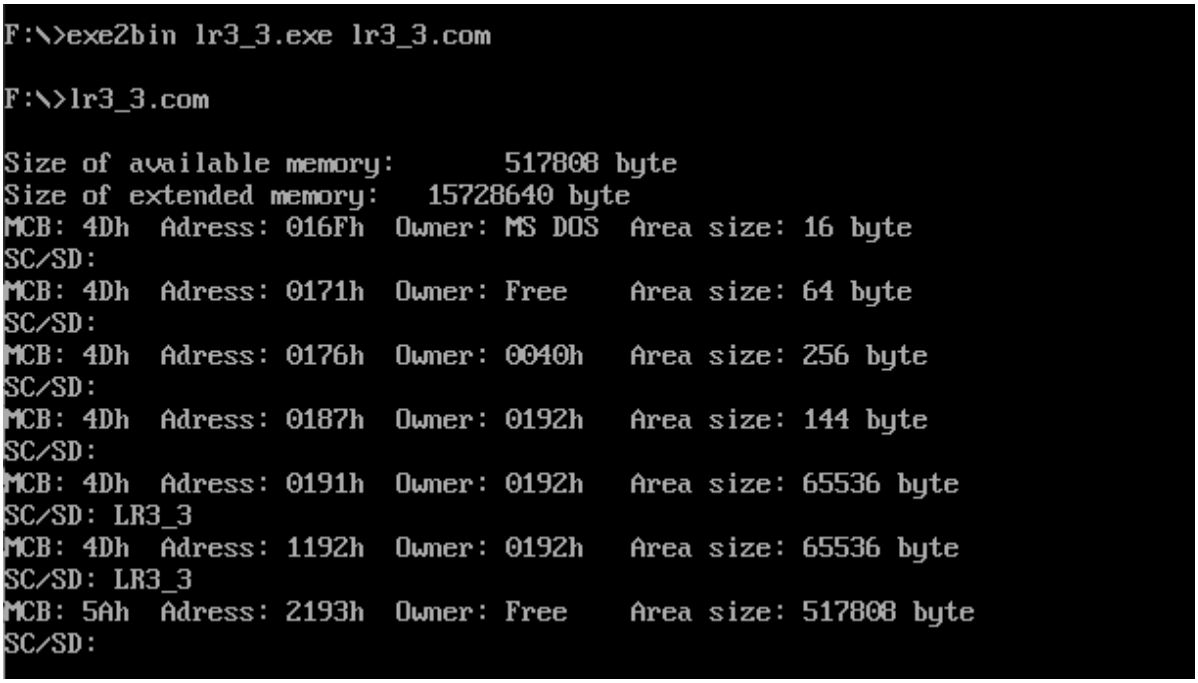

Рисунок 3. Результат работы программы. Шаг 3.

Шаг 4.

Данная версия программы запрашивает дополнительную память, перед этим её не освободив. Функция 48h устанавливает флаг CF – невозможно выделить память. Программа выводи на экран сообщение об этом.

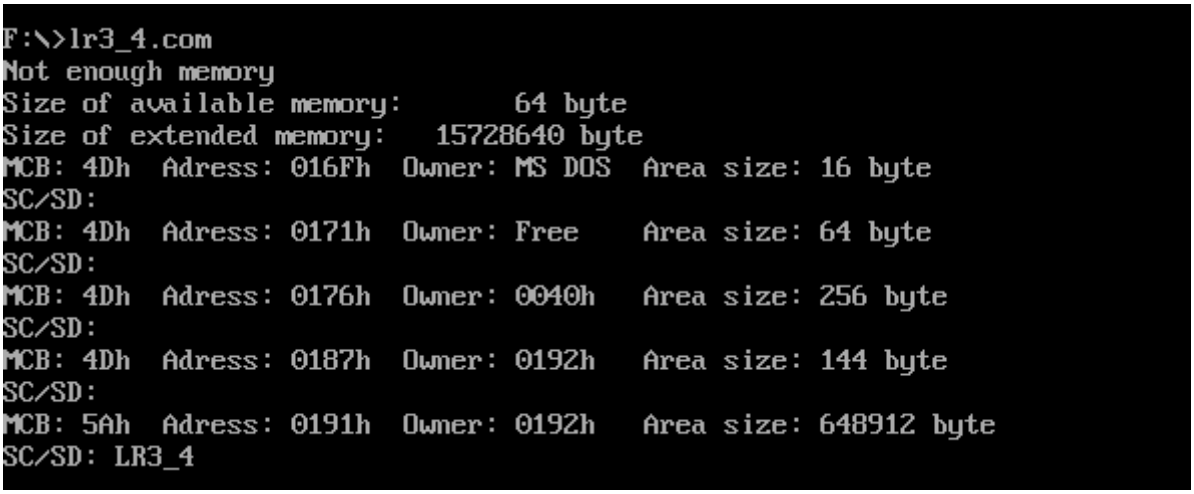

Рисунок 4. Результат работы программы. Шаг 4.

Вывод.

В ходе лабораторной работы были исследованы организации управления памятью и рассмотрены нестраничная память и способ управления динамическими разделами, реализовано управление памятью.

Контрольные вопросы по лабораторной работе №3.

1) Что означает "доступный объем памяти"?

Доступный объем памяти – это объем основной свободной памяти, который может использоваться для загрузки прикладных и системных программ.

2) Где MCB блок Вашей программы в списке?

В первой версии программы (шаг 1) это последний блок №5.

Во второй версии программы (шаг 2) это предпоследний блок №5, так как после него следует свободный блок.

В третьей версии программы (шаг 3) это блок №5 и блок №6.

В четвертой версии программы (шаг 4) это последний блок №5.

3) Какой размер памяти занимает программа в каждом случае?

В первом случае программа занимает все доступное пространство 648912 байт.

Во втором случае программа занимает 65536 байт.

В третьем случае программа занимает два блока по 65536 байт. 65536+65536 = 131072 байт.

В четвёртом случае программа выделяет 648912 байт, так как дополнительную память выделить невозможно.