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

Кафедра Математического обеспечения электронно-вычислительных машин

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Операционные системы»

ТЕМА: ИССЛЕДОВАНИЕ ОРГАНИЗАЦИИ УПРАВЛЕНИЯ ОСНОВНОЙ ПАМЯТЬЮ.

Студентка гр. 0382 _____ Рубежова Н.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 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 операционной системы.

Задание.

Шаг 1. 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.

Запустить программу и внимательно оценить результаты.

Шаг 2. Изменить программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого нужно использовать функцию 4Ah прерывания 21h. Запустить модифицированную программу. Сравнить выходные данные с результатами, полученными на предыдущем шаге.

Шаг 3. Изменить программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H. Повторить эксперимент, запустив модифицированную

программу. Сравнить выходные данные с результатами, полученными на предыдущих шагах.

Шаг 4. Изменить первоначальный вариант программы (с шага 2), запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти. Обязательно обрабатывать завершение функций ядра, проверяя флаг CF.

Шаг 5. Оценить результаты, полученные на предыдущих шагах. Ответить на контрольные вопросы.

Ход выполнения.

Для каждого шага был реализован программный модуль типа .COM, всего таких модуля – четыре, каждый из них выводит требуемую информацию и выполняет требования задания.

1. Для 1го модуля были написаны следующие процедуры:

AVAIL_MEM_PRINT – реализует вывод информации о количестве доступной памяти с помощью функции 4Ah прерывания 21h .

EXT_MEM_PRINT – реализует вывод информации о размере расширенной памяти, используя данные из ячеек CMOS.

MCB_PRINT – реализует вывод цепочки блоков управления памятью MCB с вызовом 52h прерывания 21h.

PRINT – вспомогательная процедура вывода строки через 09h прерывания 21h.

Результаты выполнения первого модуля см. ниже. Видим, что программа занимает 648912 байт. Также отображается и выводится информация о других блоках MCB.

```
C:\>var1.com
Available memory: 648912 bytes
Extended memory: 15728640 bytes
1_MCB  Address:016F   PSP_address: 0008  Size:    16  SD/SC:
2_MCB  Address:0171   PSP_address: 0000  Size:    64  SD/SC: DPMILOAD
3_MCB  Address:0176   PSP_address: 0040  Size:   256  SD/SC:
4_MCB  Address:0187   PSP_address: 0192  Size:   144  SD/SC:
5_MCB  Address:0191   PSP_address: 0192  Size:648912  SD/SC: VAR1
```

Рисунок 1 – Запуск первого модуля var1.com

2. Для 2го модуля была добавлена к предыдущим следующая процедура:

MEM_FREE – реализует освобождение памяти, которое не используется программой, с помощью функции 4Ah прерывания 21h .

Результаты выполнения второго модуля см. ниже. Видим, что блок с программой стал занимать меньше – всего 816 байт. Неиспользуемая память выделяется в отдельный, 6ой блок MCB.

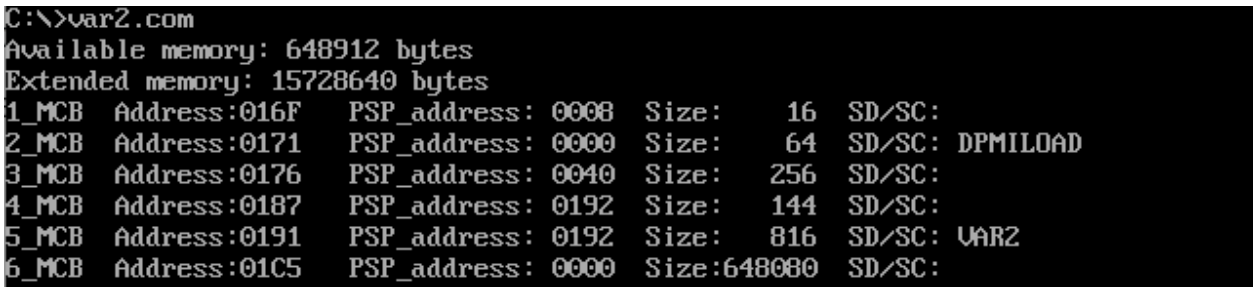

Рисунок 2 – Запуск второго модуля var2.com

3. Для 3го модуля была добавлена следующая процедура:

MEM_REQUEST – реализует запрос 64Кб памяти, с помощью функции 48h прерывания 21h. В случае, если данный объем памяти выделить невозможно, программа выводит сообщение: «Memory request is unavailable». Данная процедура вызывается после MEM_FREE - освобождения памяти, неиспользуемой программой. Затем вызывается MCB_PRINT.

Результаты выполнения третьего модуля см. ниже. Видим, что блок с неиспользуемой памятью выделился в 6ой блок MCB, а так как программа запросила еще 64Кб, этот блок ушел под программу. Запрос памяти выполнен успешно, без сообщений об ошибке.

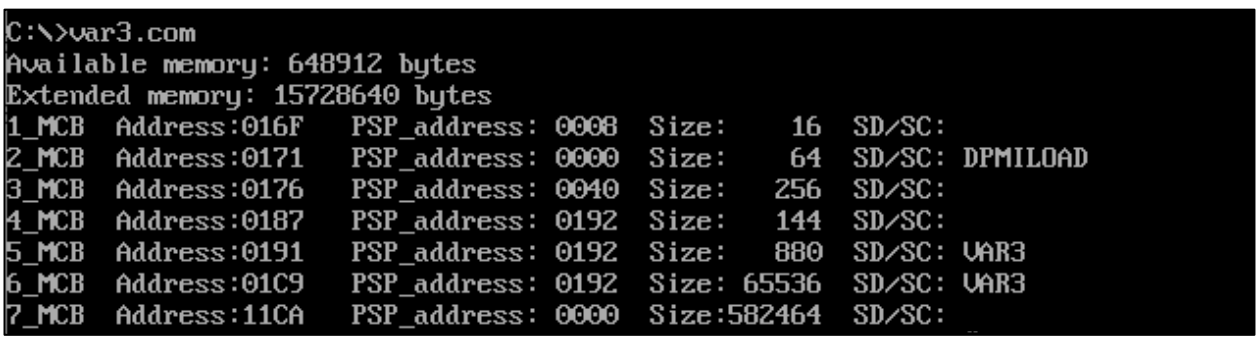

Рисунок 3 – Запуск третьего модуля var3.com

4. В четвертом модуле в отличие от третьего, меняется порядок вызова процедур: сначала запрашиваются 64Кб памяти (вызывается MEM_REQUEST), и уже затем вызывается процедура MEM_FREE - процедура освобождения неиспользуемой памяти.

Результаты выполнения четвертого модуля см. ниже. Видим, что возникла ошибка при запросе 64Кб памяти, поскольку до освобождения неиспользуемой памяти программа занимает большое количество байт и этих 64Кб свободных просто нет, поэтому запрос не выполняется. Затем вызывается MEM_FREE и блок с программой занимает уже гораздо меньше байт(5ый MCB блок), а неиспользуемая память высвобождается в отдельный блок(6ой MCB блок).

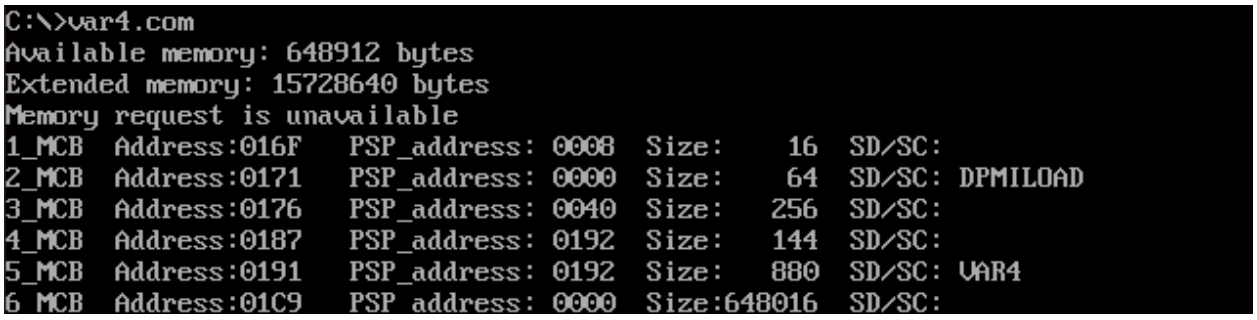

Рисунок 4 – Запуск четвертого модуля var4.com

Контрольные вопросы.

1. Что означает "доступный объем памяти"?

Ответ: Доступный объём памяти – максимальный размер оперативной памяти, который может быть использован программой.

2. Где MCB блок Вашей программы в списке?

Ответ: MCB блок программы на каждом шаге можно найти в списке MCB блоков там, где в столбце SD/SC значение: VAR#, где # - номер шага.

3. Какой размер памяти занимает программа в каждом случае?

Ответ: Для того, чтобы определить размер памяти, занимаемый программой, нужно посчитать сумму всех блоков MCB, помеченных в SD/SC

“VAR#”. То есть на первом шаге – 648912 байт, на втором – 816 байт, на третьем 66416 байт, на четвертом – 880 байт.

Выводы.

Была изучена организация управления памятью, а также рассмотрена нестраничная память и способ управления динамическими разделами. Были исследованы структуры данных и работа функций управления памятью ядра операционной системы.