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 гр. 0382 _____ Азаров М.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.

Ход работы

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

Название процедуры	Что делает
`print_avail_mem`	Получает кол-во доступной памяти и выводит его на экран.
`print_extended_mem`	Получает размер расширенной памяти и выводит его на экран.
`print_all_MCB`	Выводит цепочку блоков управления памятью
`parag_to_dec`	Переводит размер памяти в параграфах в байты , а затем в строчное представление в 10-ной с.с.
`print_MCB`	Выводит текущий блок управления памяти, адрес которого находится в ES.
`print_number_MCB`	Выводит номер текущего блока управления

	памяти
print_MCB_address	Выводит адрес текущего блока управления памяти
print_addr_psp_owner	Выводит сегментный адрес PSP владельца участка памяти
print_size_MCB	Выводит размер участка в параграфах
print_scsd	Выводит последние восемь байт MCB как символы

Также были использованы некоторые из данных в методичке процедур. В совокупности все эти процедуры выполняют поставленную задачу в **Шаг 1.**

Результат работы программы:

```
Size available memory:   648912 bytes
Size extended memory:    245760 bytes

MCB #01
Address MCB: 016F
Address PSP owner: 0008
Size:      16 bytes
SC/SD:

MCB #02
Address MCB: 0171
Address PSP owner: 0000
Size:      64 bytes
SC/SD:

MCB #03
Address MCB: 0176
Address PSP owner: 0040
Size:     256 bytes
SC/SD:
```

```
MCB #04
Address MCB: 0187
Address PSP owner: 0192
Size:     144 bytes
SC/SD:

MCB #05
Address MCB: 0191
Address PSP owner: 0192
Size:  648912 bytes
SC/SD: LAB3_1
```

2. Для модификации программного модуля понадобилось разработать дополнительную функцию:

Название процедуры	**Что делает**
`free_mem`	Освобождает память, которую текущая программа не занимает

Результат работы программы:

```
Size available memory:   648912 bytes
Size extended memory:    245760 bytes

MCB #01
Address MCB: 016F
Address PSP owner: 0008
Size:      16 bytes
SC/SD:

MCB #02
Address MCB: 0171
Address PSP owner: 0000
Size:      64 bytes
SC/SD:
```

```
MCB #03
Address MCB: 0176
Address PSP owner: 0040
Size:     256 bytes
SC/SD:

MCB #04
Address MCB: 0187
Address PSP owner: 0192
Size:     144 bytes
SC/SD:

MCB #05
Address MCB: 0191
Address PSP owner: 0192
Size:     864 bytes
SC/SD: LAB3_2

MCB #06
Address MCB: 01C8
Address PSP owner: 0000
Size:  648032 bytes
SC/SD:  #яt#&ѓ
```

Как мы видим, теперь наша программа занимает только необходимую ей память.

3. Также для требуемой модификации в **Шаг 3**, понадобилось разработать еще одну процедуру:

Название процедуры	**Что делает**
`malloc`	Запрашивает выделение памяти для программы в размере BX байт

Результат работы программы:

```
Size available memory:   648912 bytes
Size extended memory:    245760 bytes

MCB #01
Address MCB: 016F
Address PSP owner: 0008
Size:      16 bytes
SC/SD:

MCB #02
Address MCB: 0171
Address PSP owner: 0000
Size:      64 bytes
SC/SD:

MCB #03
Address MCB: 0176
Address PSP owner: 0040
Size:     256 bytes
SC/SD:

MCB #04
Address MCB: 0187
Address PSP owner: 0192
Size:     144 bytes
SC/SD:

MCB #05
Address MCB: 0191
Address PSP owner: 0192
Size:     976 bytes
SC/SD: LAB3_3

MCB #06
Address MCB: 01CF
Address PSP owner: 0192
```

```
Size:   65536 bytes
SC/SD: LAB3_3


MCB #07
Address MCB: 11D0
Address PSP owner: 0000
Size:  582368 bytes
SC/SD: .#tgt s
```

Так как мы сначала освободили память , то появился блок памяти №5 принадлежащий программе . После запроса на 64 Кб образовался еще один блок памяти №6 принадлежащий программе в размере 64 Кб.

4. Теперь сначала запросим 64 Кб, а потом освободим память.

Результат работы программы:

```
Size available memory:   648912 bytes
Size extended memory:    245760 bytes
Error: requested memory more then available free memory.


MCB #01
Address MCB: 016F
Address PSP owner: 0008
Size:      16 bytes
SC/SD:


MCB #02
Address MCB: 0171
Address PSP owner: 0000
Size:      64 bytes
SC/SD:


MCB #03
Address MCB: 0176
Address PSP owner: 0040
Size:     256 bytes
SC/SD:
```

```
MCB #04
Address MCB: 0187
Address PSP owner: 0192
Size:     144 bytes
SC/SD:


MCB #05
Address MCB: 0191
Address PSP owner: 0192
Size:     976 bytes
SC/SD: LAB3_4


MCB #06
Address MCB: 01CF
Address PSP owner: 0000
Size:  647920 bytes
SC/SD:                                                          Ъ|
,u#я
```

Как и ожидалось операция запроса памяти завершилась с ошибкой, так как свободной памяти отсутствует (вся память принадлежит текущей программе), поэтому мы и не можем запросить еще память.

Ответы на контрольные вопросы.

1) Что означает "доступный объем памяти"?

Ответ: Доступная память – это память, занимаемая программой

2) Где MCB блок Вашей программы в списке?

Ответ: MCB блоку запускаемой программы соответствует тот MCB блок который в поле SD/SC имеет название запускаемой программы , в нашем случае это LAB3_? , где вместо ? может стоять «1», «2», «3» , «4».

3) Какой размер памяти занимает программа в каждом случае?

Ответ: На первом шаге программа занимает - 648912 байт, на втором - 864 байт, на третьем — 976 + 65536 = 66512 байт, и на четвертом - 976 байт.

Вывод.

В ходе работы были изучены структуры данных и работа функций управления памятью ядра операционной системы. Разработана программа которая выводит сведения о текущем состоянии памяти ОС и умеет освобождать и запрашивать требуемое нами количество памяти.