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»

Студентка гр. 0381 _____ Степанова Е.М.

Преподаватель _____ Губкин А.Ф.

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.

Задания.

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.

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

Основные сведения.

В работе используются следующие процедуры:

TETR_TO_HEX — переводит 10-ые цифры в символьный код.

BYTE_TO_HEX — переводит байт в 16 системе счисления в символьный код(данная функция используется, когда при определении типа PC ни один код не совпал с рассматриваемыми значениями).

WRD_TO_HEX – переводит слова, представленные в 16 сс, в символьный код.

BYTE_TO_DEC – переводит байт, представленный в 16 сс, в символьный код, представленный в 10-ой сс.

Данные процедуры были взяты из раздела «общие сведения». Так же для корректной работы программы были написаны следующие процедуры:

PRINT – выводит значения в консоль.

CONVERT – переводит число, расположенное в двух регистр в 10-ую систему счисления и выводит результат на экран.

GET_EXTENDED_MEM – выводит количество доступной памяти.

GET_AVAILABLE_MEM – выводит размер расширенной памяти.

GET_BLOCK_CHAIN – выводит цепочку блоков управления памятью.

Так же были объявлены строки для вывода информации:

- AVAILABLE_MEM db 13,10,'Amount of available memory: $'
- EXTENDED_MEM db 13,10,'Size of extended memory: $'
- STR_BYTES db ' bytes $'
- MCB_TYPE db 13,10,'MCB:0 $'
- ADRESS db 'MCB Adress: $'
- OWNER db 'PSP owner address: $'
- AREA_SIZE db 'Area size: $'
- MCB_SD_SC db 'SD/SC: $'
- NEWLINE db 0DH,0AH,'$'
- MEM_PROBLEMS db 'Not enough memory...$'

Выполнение работы.

Шаг 1.

При запуске программы сначала вызывается процедура GET_AVAILABLE_MEM, которая выводит доступную память в консоль. Для этого используется функция 48h прерывания 21h. Программа запрашивает большой объём памяти(больше, чем свободно в системе), это позволяет вернуть размер свободной памяти в параграфах. С помощью процедуры CONVERT параграфы переводятся в байты и значение выводится в консоль.

В процедуре CONVERT в регистре cx хранится количество цифр в

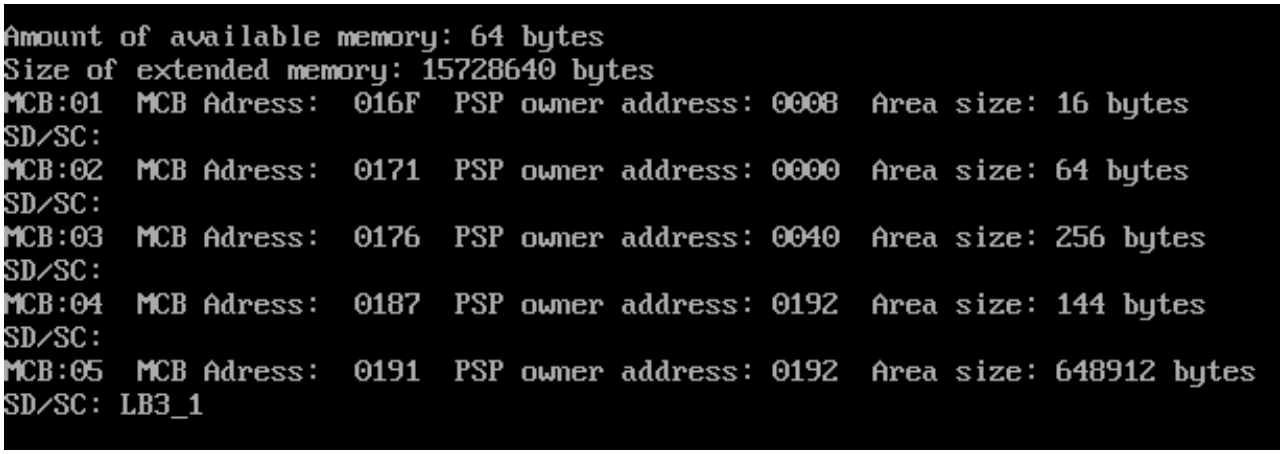
числе, bx хранится основание системы счисления. Далее программа делит число на основание системы счисления, получая в остатке последнюю цифру. Сразу же выводить её нельзя, поэтому она сохраняется в стеке. В cx увеличивается количество цифр. А с частным программа повторяет то же самое, отделяя от него очередную цифру справа, пока не останется ноль, что будет обозначать, что далее слева только нули — конец. Когда программа дошла до этого момента и при этом во втором регистре так же ноль, из стека извлекается очередная цифра и переводится в символ, после чего выводится. Это повторяется ровно столько раз, сколько цифр подсчитано в cx.

Далее вызывается процедура GET_EXTENDED_MEM, которая выводит объем расширенной памяти. Для этого программа обращается к ячейкам 30h и 31h памяти CMOS. Полученное значение является размером расширенной памяти, поэтому после этого программа лишь переводит значение в байты с помощью процедуры CONVERT и выводит в консоль результат в 10-ой системе счисления.

Затем программа вызывает процедуру GET_BLOCK_CHAIN, которая выводит цепочку блоков управления памятью. С помощью функции 52h int 21h программа получает адрес самого первого MCB. Далее постепенно выводится в консоль содержимое каждого блока, каждый раз переходя к следующему блоку. Считывание цепочки блоков завершается, когда достигнут последний блок в списке.

Результат работы программы:

Программа выводит размер доступной памяти, равный 64 байтам — доступная память занимает весь объем памяти, но есть свободный блок после блока DOS, длина которого равна 64 байтам. Размер расширенной памяти равен 15728640 байт, то есть 15 МБ.



Шаг 2.

В данной версии программа освобождает память, которую не использует, поэтому объем свободной памяти увеличивается, а в выводе цепочки блоков управления памятью появляется новая строка — свободный блок.

Шаг 3.

В данной версии программа после освобождения памяти, запрашивает ещё 64 Кб памяти, поэтому в выводе цепочки блоков управления памятью появляется ещё один блок размером 64 Кб памяти. Объем свободной памяти останется таким же, как и на шаге 2.

```
Amount of available memory: 517808 bytes
Size of extended memory: 15728640 bytes
MCB:01  MCB Adress:  016F  PSP owner address: 0008  Area size: 16 bytes
SD/SC:
MCB:02  MCB Adress:  0171  PSP owner address: 0000  Area size: 64 bytes
SD/SC:
MCB:03  MCB Adress:  0176  PSP owner address: 0040  Area size: 256 bytes
SD/SC:
MCB:04  MCB Adress:  0187  PSP owner address: 0192  Area size: 144 bytes
SD/SC:
MCB:05  MCB Adress:  0191  PSP owner address: 0192  Area size: 65536 bytes
SD/SC: LB3_3
MCB:06  MCB Adress:  1192  PSP owner address: 0192  Area size: 65536 bytes
SD/SC: LB3_3
MCB:07  MCB Adress:  2193  PSP owner address: 0000  Area size: 517808 bytes
SD/SC:
```

Шаг 4.

В данной версии программа запрашивает дополнительную память, однако предварительно данная память не была освобождена, поэтому функция 48h устанавливает флаг CF – невозможно выделить память. Программа выводит в консоль сообщение об этом.

```
Not enough memory...
Amount of available memory: 64 bytes
Size of extended memory: 15728640 bytes
MCB:01  MCB Adress:  016F  PSP owner address: 0008  Area size: 16 bytes
SD/SC:
MCB:02  MCB Adress:  0171  PSP owner address: 0000  Area size: 64 bytes
SD/SC:
MCB:03  MCB Adress:  0176  PSP owner address: 0040  Area size: 256 bytes
SD/SC:
MCB:04  MCB Adress:  0187  PSP owner address: 0192  Area size: 144 bytes
SD/SC:
MCB:05  MCB Adress:  0191  PSP owner address: 0192  Area size: 648912 bytes
SD/SC: LB3_4
```

Вывод.

Были исследованы организации управления памятью и рассмотрены нестраничная память и способ управления динамическими разделами. А так же реализовано управление памятью.

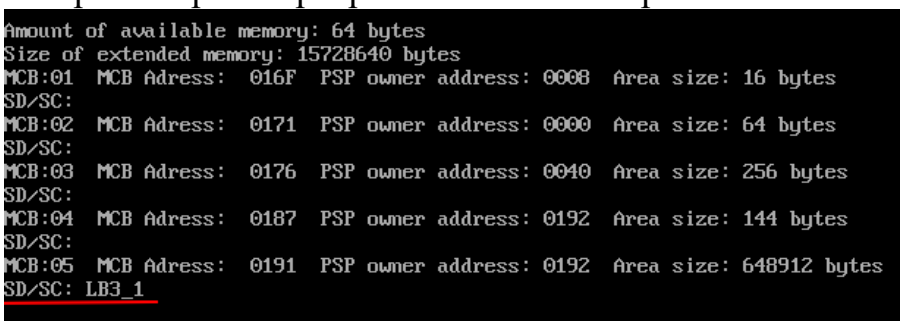
ВОПРОСЫ

1. Что означает "доступный объем памяти"?

Доступный объём памяти — это размер наибольшего свободного участка памяти, то есть максимальный размер блока, который может запросить программа. Для двух свободных блоков памяти, расположенных друг за другом, доступным объемом будет считаться наибольший блок памяти среди них.

2. Где MCB блок Вашей программы в списке?

В первой версии программы MCB блок располагается в самом конце списка.



Во второй версии данный блок является предпоследним в списке, так как после него следует свободный блок.

В третье версии в MCB блок моей программы входят 5 и 6 блоки. 6-ой блок — свободный.

3. Какой размер памяти занимает программа в каждом случае?

В первом случае программа занимает всё доступное пространство —648912 байта.

Во втором случае программа занимает 65536 байта.

В третьем случае программа занимает 65536 + 65536 = 131072 байта.

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