

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

ОТЧЕТ
по практической работе №3
по дисциплине «Операционные системы»
Тема: Исследование организации управления основной памятью

Студентка гр. 7383

Чемова К.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2019

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.

Необходимые сведения для составления программы.

Учёт занятой и свободной памяти ведется при помощи списка блоков управления памятью MCB (Memory Control Block). MCB занимает 16 байт (параграф) и располагается всегда с адреса кратного 16 (адрес сегмента ОП) и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет.

MCB имеет следующую структуру:

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 драйверу OS XMS UMB 0007h - участок является исключенной верхней памятью драйверов 0008h - участок принадлежит MS DOS

		FFFAh - участок занят управляющим блоком 386MAX UMB FFFDh - участок заблокирован 386MAX FFFEh - участок принадлежит 386MAX UMB
03h	2	Размер участка в параграфах
05h	3	Зарезервирован
08h	8	"SC" - если участок принадлежит MS DOS, то в нем системный код "SD" - если участок принадлежит MS DOS, то в нем системные данные

По сегментному адресу и размеру участка памяти, контролируемого этим MCB можно определить местоположение следующего MCB в списке.

Адрес первого MCB хранится во внутренней структуре MS DOS, называемой "List of Lists" (список списков). Доступ к указателю на эту структуру можно получить, используя функцию f52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX-2] и есть адрес самого первого MCB.

Размер расширенной памяти находится в ячейках 30h, 31h CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта. Размер расширенной памяти в Кбайтах можно определить обращаясь к ячейкам CMOS следующим образом:

```

mov AL, 30h ; запись адреса ячейки CMOS
out 70h, AL
in AL, 71h ; чтение младшего байта
mov BL, AL ; размера расширенной памяти
mov AL, 31h ; запись адреса ячейки CMOS
out 70h, AL

```

in AL, 71h ; чтение старшего байта размера расширенной памяти

Постановка задачи.

Шаг 1. Необходимо написать и отладить программный модуль типа .COM, 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 MSB выводятся как символы, не следует преобразовывать их в шестнадцатеричные числа.

Шаг 2. Далее необходимо изменить программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah прерывания 21h (пример в разделе «Использование функции 4AH»).

Шаг 3. Затем необходимо изменить программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H.

Шаг 4. Далее нужно изменить первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти.

Шаг 5. Оформить отчёт и ответить на контрольные вопросы.

Процедуры, используемые в программе.

TETR_TO_HEX – Используется для перевода половины байта в шестнадцатеричную систему счисления.

BYTE_TO_HEX – Используется для перевода байта регистра AL в шестнадцатеричную систему счисления, помещая результат в AH.

WRD_TO_HEX – Используется для перевода двух байт регистра AX в шестнадцатеричную систему счисления, помещая результат в регистр DI.

BYTE_TO_DEC – Используется для перевода байта регистра AL в десятичную систему счисления, помещая результат в SI.

GET_MCB – Вывод таблицу MCB

PRINT – Вывод на экран.

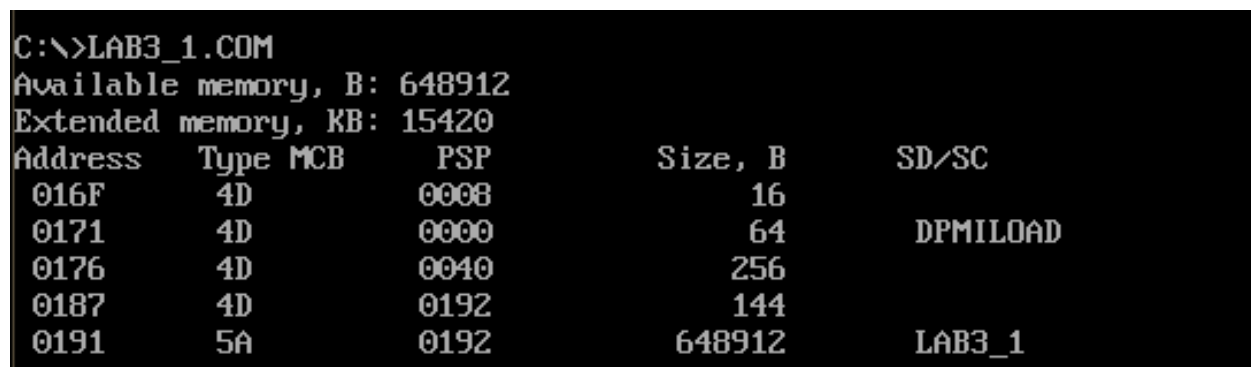
Структуры данных.

Таблица 1 – Структуры данных

Название поля данных	Тип	Назначение
_avl_mem	db	Доступная память
_ext_mem	db	Расширенная память
_table_MCB	db	Таблица MCB
result	Тат db	Строка для вывода

Результат работы.

На рис. 1 представлена работа программы, выполняющая шаг 1. Все доступные 648912 байт отдаются программе.



Address	Type	MCB	PSP	Size, B	SD/SC
016F	4D	0008		16	
0171	4D	0000		64	DPMILOAD
0176	4D	0040		256	
0187	4D	0192		144	
0191	5A	0192		648912	LAB3_1

Рисунок 1 – Результат работы lab3_1.com

Был изменён исходный код программы: теперь программа освобождает не занимаемую ею память. Создается новый блок, который обозначен как свободный участок, размером 636016 байт. Результат работы представлен на рис.2.

```

C:\>LAB3_2.COM
Available memory, B: 648912
Extended memory, KB: 15420
Address   Type MCB   PSP      Size, B   SD/SC
016F      4D      0008        16
0171      4D      0000        64   DPMILOAD
0176      4D      0040       256
0187      4D      0192       144
0191      4D      0192     12880   LAB3_2
04B7      5A      0000    636016

```

Рисунок 2 – Результат работы lab3_2.com

Код программы снова был изменён. Сначала, как и во втором случае, происходит освобождение памяти, затем программа запрашивает 64 Кбайт (65536 байт) памяти. На свободном участке создается новый блок, который следует за основным блоком программы и занимает 65536 байт. На рис. 3 представлен результат.

```

C:\>LAB3_3.COM
Available memory, B: 648912
Extended memory, KB: 15420
Address   Type MCB   PSP      Size, B   SD/SC
016F      4D      0008        16
0171      4D      0000        64   DPMILOAD
0176      4D      0040       256
0187      4D      0192       144
0191      4D      0192     12992   LAB3_3
04BE      4D      0192    65536   LAB3_3
14BF      5A      0000    570352   #_tA&e▲

```

Рисунок 3 – Результат работы lab3_3.com

Код программы снова был изменён. Происходит запрос 64 Кбайт до освобождения памяти. Однако выдаётся ошибка, так как запрос памяти происходит в тот момент, когда вся доступная память занята программой. Затем происходит освобождение памяти, аналогично второму случаю. Результат работы представлен на рис.4.

```
C:\>LAB3_4.COM
Available memory, B: 648912
Memory allocation error!!!
Extended memory, KB: 15420
Address   Type MCB   PSP      Size, B   SD/SC
016F      4D      0008        16
0171      4D      0000        64   DPMILOAD
0176      4D      0040       256
0187      4D      0192       144
0191      4D      0192     13584   LAB3_4
04E3      5A      0000    635312
```

Рисунок 4 – Результат работы lab3_4.com

Ответы на контрольные вопросы.

1) Что означает «доступный объём памяти»?

Доступный объём памяти – это максимальный объем памяти, в который можно загружать пользовательские программы.

2) Где МСВ блок Вашей программы в списке?

По адресу 0191h расположен блок МСВ в первой программе.

По адресу 0191h расположен блок МСВ, а по адресу 04B7 расположен блок освобожденной памяти в первой модификации.

По адресу 0191h расположен блок МСВ, а по адресу 04BE расположен блок запрошенной памяти, а по 14BF – блок освобожденной памяти.

По адресу 0191h расположен блок МСВ, а по адресу 04E3 – блок, обозначенный как пустой участок.

3) Какой размер памяти занимает программа в каждом случае?

1. 648912 байт, вся выделенная под программу память.

2. $648912 - 636016 - 16 = 12880$ байт.

3. $648912 - 570352 - 65536 - 2 * 16 = 12992$ байт.

4. Произошла ошибка выделения дополнительной памяти,
 $648912 - 635312 - 16 = 13584$ байт.

Вывод.

В ходе работы было проведено исследование структуры данных и работы функций управления памятью ядра операционной системы, а также рассмотрены нестраничная память и способы управления динамическими разделами.