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. 9383 _____ Соседков К.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

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.

Выполнение работы.

Шаг 1.

Для выполнения первого шага был написан и отлажен программный модуль типа .COM, который выбирает и распечатывает следующую информацию(см. Рисунок 1):

1) Количество доступной памяти.

2) Размер расширенной памяти.

3) Выводит цепочку блоков управления памятью.

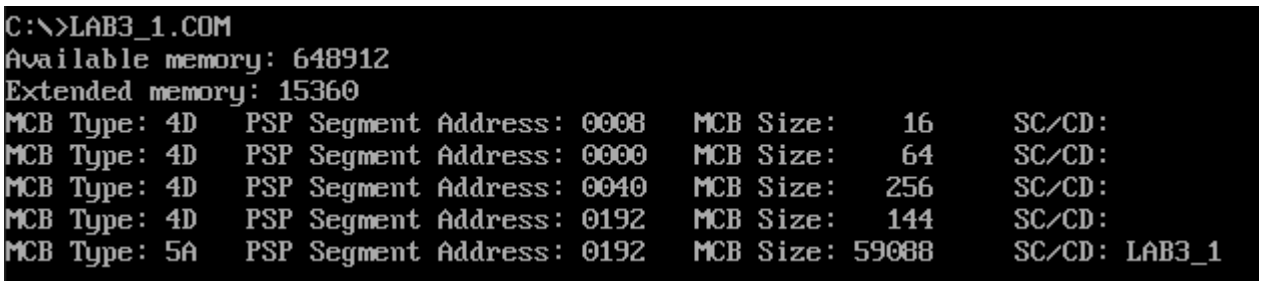

Рисунок 1

Шаг 2.

Далее программа была изменена таким образом, чтобы она освобождала память, которую она не занимает(см. Рисунок 2).

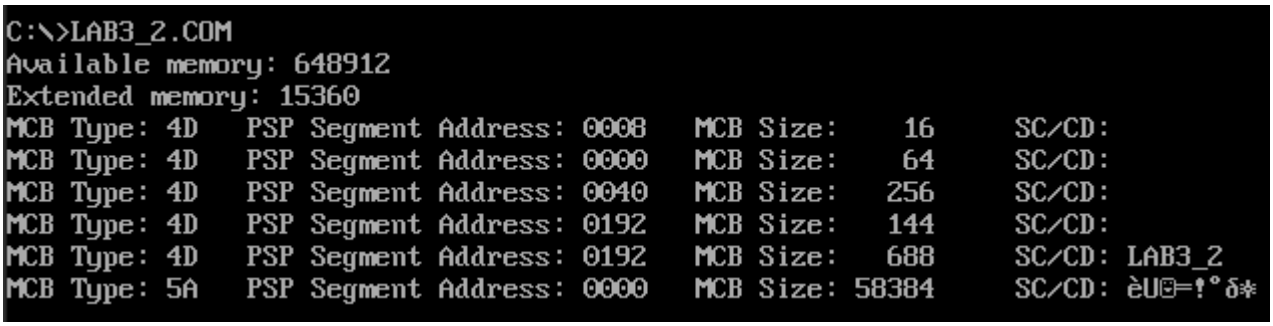

Рисунок 2

Шаг 3.

Для выполнения третьего шага программа была изменена таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H(см. Рисунок 3).

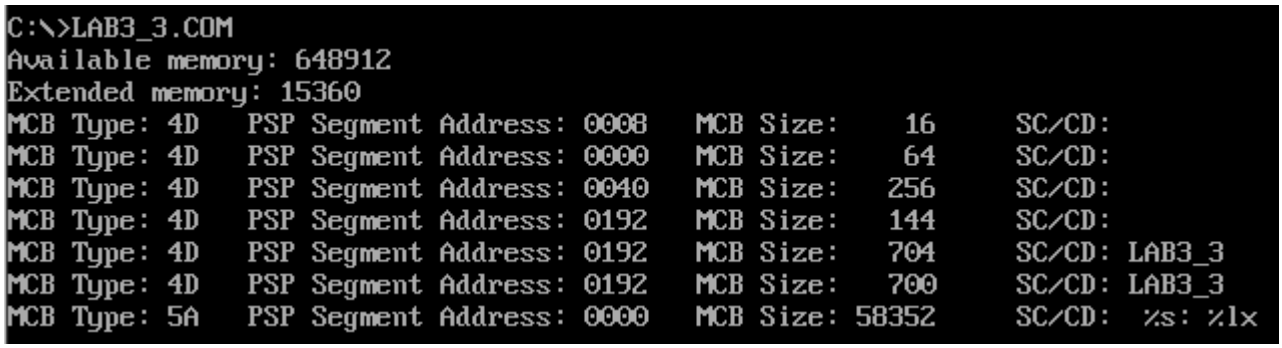

Рисунок 3

Шаг 4.

На данном шаге был выполнен запрос на 64Кб памяти функцией 48H прерывания 21H до освобождения памяти(см. Рисунок 4).

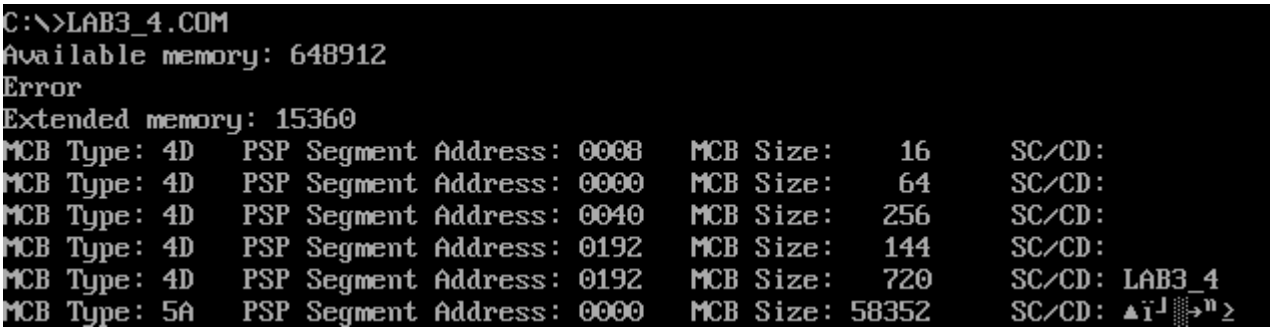

Рисунок 4

Контрольные вопросы.

1) Что означает "доступный объем памяти"?

Объем памяти который может использовать программа.

2) Где MCB блок Вашей программы в списке?

Для первой, второй и четвертой программ MCB блок находится на 4 и 5 позиции в списке. У третьей программы MCB блок находится на позициях 4,5 и 6.

3) Какой размер памяти занимает программа в каждом случае?

3.1) 59232 байта

3.2) 832 байта

3.3) 1548 байт

3.4) 864 байта

Выводы.

При выполнении лабораторной работы была освоена работа функций ядра управления памятью операционной системы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: lab3.asm

```
TESTPC  SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG 100H

START: JMP BEGIN



memory_size db 'Available memory:                ',0DH,0AH,'$'
extended_size db 'Extended memory:       ',0DH,0AH,'$'
MCB_Type db 'MCB Type:      $'
PSP_Segment_Address db 'PSP Segment Address:        $'
MCB_Size db 'MCB Size:          $'
SCCD db 'SC/CD: $'
;------------------------------
BYTE_TO_DEC PROC near
    push CX
    push DX
    mov CX,10
loop_bd:
    div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp AX,10
    jae loop_bd
    cmp AL,00h
    je end_l
    or AL,30h
    mov [SI],AL
```

```
end_l:
   pop DX
   pop CX
   ret
BYTE_TO_DEC ENDP
;------------------------------
TETR_TO_HEX PROC near
   and AL,0Fh
   cmp AL,09
   jbe NEXT
   add AL,07
   NEXT: add AL,30h
   ret
TETR_TO_HEX ENDP
;------------------------------
BYTE_TO_HEX PROC near
   ; байт в AL переводится в два символа шестн. числа в AX
   push CX
   mov AH,AL
   call TETR_TO_HEX
   xchg AL,AH
   mov CL,4
   shr AL,CL
   call TETR_TO_HEX ;в AL старшая цифра
   pop CX ;в AH младшая
   ret
BYTE_TO_HEX ENDP
;------------------------------
WRD_TO_HEX PROC near
   push BX
   mov BH,AH
   call BYTE_TO_HEX
   mov [DI],AH
   dec DI
```

```
    mov [DI],AL
    dec DI
    mov AL,BH
    call BYTE_TO_HEX
    mov [DI],AH
    dec DI
    mov [DI],AL
    pop BX
    ret
WRD_TO_HEX ENDP
;--------------------------------



BEGIN:
    ; Memory
    mov ah,4ah
    mov bx,0ffffh
    int 21h

    xor ax,ax
    mov ax, bx
    mov cx, 10h
    mul cx

    mov si, offset memory_size+23
    call BYTE_TO_DEC


    mov DX,offset memory_size
    mov AH,09h
    int 21h
```

```
; Extended
mov AL,30h
out 70h,AL
in AL,71h
mov BL,AL
mov AL,31h
out 70h,AL
in AL,71h

mov ah, al
mov al, bl

mov si, offset extended_size+21
xor dx, dx
call BYTE_TO_DEC

mov DX,offset extended_size
mov AH,09h
int 21h


;MCB
mov ah, 52h
int 21h
mov ax, es:[bx-2]
mov es, ax

read_mcb:
;MCB Type
mov al, es:[0h]
call BYTE_TO_HEX
mov si, offset MCB_Type+10
```

```
mov [si], ax
mov DX,offset MCB_Type
mov AH,09h
int 21h

;PSP Segment
mov ax, es:[1h]
mov di, offset PSP_Segment_Address+24
call WRD_TO_HEX

mov DX,offset PSP_Segment_Address
mov AH,09h
int 21h

;Size
mov ax, es:[3h]
mov cx, 10h
mul cx
mov si, offset MCB_Size+14
xor dx, dx
call BYTE_TO_DEC

mov DX,offset MCB_Size
mov AH,09h
int 21h

;Symbols
mov DX,offset SCCD
mov AH,09h
int 21h

mov ah, 02h
mov cx, 8
xor bx, bx
```

```
    print_symbols:
        mov dl, es:[8h+bx]
        int 21h
        inc bx
    loop print_symbols

    mov dl, 0dh
    int 21h
    mov dl, 0ah
    int 21h

    cmp byte ptr es:[0000h], 05Ah
    je exit
    mov ax, es
    add ax, es:[3h]
    inc ax
    mov es, ax
    jmp read_mcb

exit:
    xor al,al
    mov ah,4ch
    int 21h

_end:
TESTPC ENDS
END START
```