

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

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

Студентка гр. 8382

\_\_\_\_\_

Кузина А.М.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

## Цель работы.

Исследование структур данных и работы функций управления памятью ядра операционной системы.

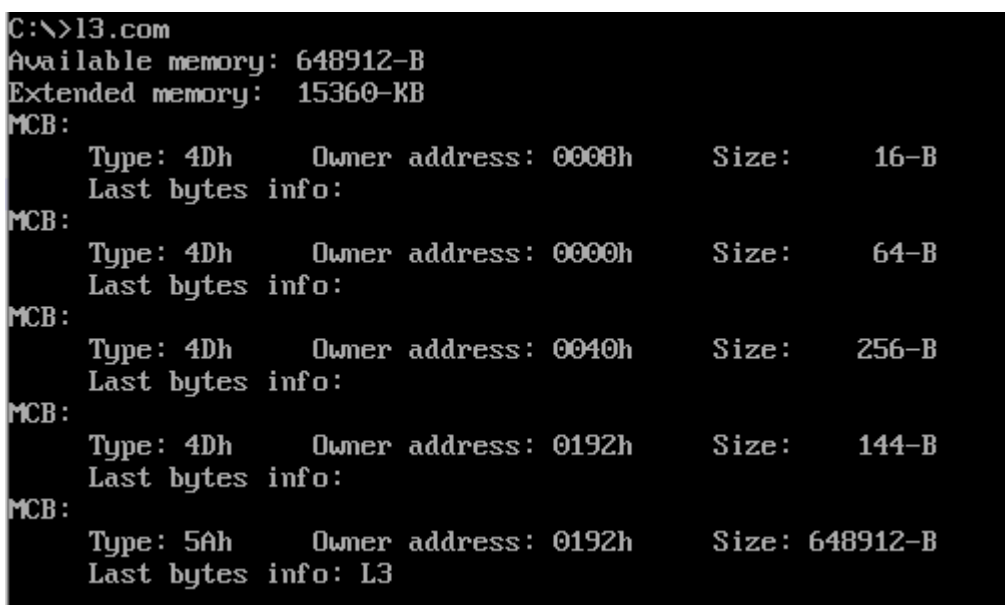
## Ход выполнения работы.

Была написана программа, исходный код которой приведен в приложении А, распечатывающая на экран следующую информацию:

- Количество доступной памяти.
- Размер расширенной памяти.
- Цепочку блоков управления памятью.

Результат работы программы представлен на рисунке 1.

Рисунок 1 — результат работы исходной программы.



```
C:\>13.com
Available memory: 648912-B
Extended memory: 15360-KB
MCB:
  Type: 4Dh      Owner address: 0008h      Size:      16-B
  Last bytes info:
MCB:
  Type: 4Dh      Owner address: 0000h      Size:      64-B
  Last bytes info:
MCB:
  Type: 4Dh      Owner address: 0040h      Size:     256-B
  Last bytes info:
MCB:
  Type: 4Dh      Owner address: 0192h      Size:     144-B
  Last bytes info:
MCB:
  Type: 5Ah      Owner address: 0192h      Size: 648912-B
  Last bytes info: L3
```

Размер доступной памяти определяется с помощью функции 4Ah прерывания 21h. При занесении в регистр bx больший размер памяти, чем может предоставить ОС, то в bx возвратится размер доступной памяти. Размер расширенной памяти находится в ячейках 30h, 31h CMOS. Адрес первого MCB можно получить используя функцию 52h прерывания 21h. Тогда слово по адресу es:[bx – 2] будет

искомым адресом. Адрес следующего блока можно вычислить из адреса и размера текущего MCB.

О каждом MCB выводится следующая информация:

- Его тип: 5Ah — если он последний в списке, и 4Dh иначе.
- Сегментный адрес владельца участка памяти.
- Размер участка в байтах.
- Информация, хранящаяся в последних 8 байтах блока.

Далее исходный код был модифицирован таким образом, чтобы программа освобождала память, которую она не занимает. Для этого была использована функция 4Ah прерывания 21h. Результат работы данной программы представлен на рисунке 2.

Рисунок 2 — результат работы l3\_1.com

```
C:\>L3_1.com
Available memory: 648912-B
Extended memory: 15360-KB
MCB:
  Type: 4Dh      Owner address: 0008h      Size:      16-B
  Last bytes info:
MCB:
  Type: 4Dh      Owner address: 0000h      Size:      64-B
  Last bytes info:
MCB:
  Type: 4Dh      Owner address: 0040h      Size:     256-B
  Last bytes info:
MCB:
  Type: 4Dh      Owner address: 0192h      Size:     144-B
  Last bytes info:
MCB:
  Type: 4Dh      Owner address: 0192h      Size:   12928-B
  Last bytes info: L3_1
MCB:
  Type: 5Ah      Owner address: 0000h      Size:  635968-B
  Last bytes info: ♦i^♦èF°ê
```

Далее программа вновь была модифицирована так, что после освобождения не занимаемой памяти, программа запрашивает 64Кб памяти дополнительно с помощью функции 48h прерывания 21h. Результат ее работы представлен на рисунке 3.

Рисунок 3 — результат работы l3\_2.com

```

C:\>l3_2.com
Available memory: 648912-B
Successful allocation
Extended memory: 15360-KB
MCB:
  Type: 4Dh   Owner address: 0008h   Size:      16-B   Last bytes info:
MCB:
  Type: 4Dh   Owner address: 0000h   Size:      64-B   Last bytes info:
MCB:
  Type: 4Dh   Owner address: 0040h   Size:     256-B   Last bytes info:
MCB:
  Type: 4Dh   Owner address: 0192h   Size:     144-B   Last bytes info:
MCB:
  Type: 4Dh   Owner address: 0192h   Size:    13472-B   Last bytes info: L3_2
MCB:
  Type: 4Dh   Owner address: 0192h   Size:    65536-B   Last bytes info: L3_2
MCB:
  Type: 5Ah   Owner address: 0000h   Size:   569872-B   Last bytes info:

```

После этого программа вновь была изменена: программа запрашивает 64Кб памяти дополнительно с помощью функции 48h прерывания 21h и затем освобождает незанятую память. В программах 3 и 4, успешность выделения памяти проверяется с помощью флага CF. На рисунке 4 представлен результат работы данной программы.

```

C:\>l3_3.com
Available memory: 648912-B
!Something goes wrong with allocation!
Extended memory: 15360-KB
MCB:
  Type: 4Dh   Owner address: 0008h   Size:      16-B
  Last bytes info:
MCB:
  Type: 4Dh   Owner address: 0000h   Size:      64-B
  Last bytes info:
MCB:
  Type: 4Dh   Owner address: 0040h   Size:     256-B
  Last bytes info:
MCB:
  Type: 4Dh   Owner address: 0192h   Size:     144-B
  Last bytes info:
MCB:
  Type: 4Dh   Owner address: 0192h   Size:    13776-B
  Last bytes info: L3_3
MCB:
  Type: 5Ah   Owner address: 0000h   Size:   635120-B
  Last bytes info: 00>¿C u?

```

### **Контрольные вопросы**

- Что означает «доступный объем памяти»?

Максимальный объем памяти, который может быть выделен программе.

- Где MCB блоки вашей программе в списке?

В первых двух программах это четвертый и пятый блоки, в третьей программе это четвертый, пятый и шестой блоки, в четвертой программе это четвертый и пятый блоки. Данные блоки имеют одного владельца с сегментным адресом PSP — 0912h.

- Какой размер памяти занимает программа в каждом случае?

Размер памяти занимаемой программой, можно узнать просуммировав память, занимаемую MCB блоками программы.

Для первой программы это 649056 байт, для второй программы память была освобождена, поэтому в итоге программа занимает 13072 байта, для третьей программы была успешно получена дополнительная память и в итоге память программы составила 79152 байта, в четвертой программе дополнительная память получена не была, и память программы составила 13920 байт.

### **Выводы**

В ходе работы были исследованы структуры данных и работы функций управления памятью ядра операционной системы.

## ПРИЛОЖЕНИЕ А

### Исходный код программы l3.asm

```
STT    SEGMENT
        ASSUME CS:STT, DS:STT, ES:NOTHING, SS:NOTHING
        ORG 100H
START: JMP BEGIN

AM db 'Available memory:      -B', 13,10, '$'
EM db 'Extended memory:      -KB', 13, 10, '$'
McbL db 'MCB:', 13, 10, '$'
Typ db 'Type: ', '$'
Hex db 'h', '$'
Sect db 'Owner address: ', '$'
Siz db 'Size:      -B', '$'
EndLine db ' ', 13, 10, '$'
Tab db ' ', '$'
Info db 'Last bytes info: ', '$'
Mem db ' !Something goes wrong with allocation!', 13, 10, '$'

BEGIN:
;Available memory

        mov ah, 4ah
        mov bx, 0ffffh
        int 21h
        mov ax, bx
        mov bx, 10h
        mul bx
        mov si, offset AM
        add si, 23
        call WRDTODEC
        mov dx, offset AM
        mov ah, 09h
        int 21h

;Extended memory
        mov ax, 0
        mov dx, 0
        mov al, 30h
        out 70h, al
        in al, 71h
        mov bl, al
        mov al, 31h
        out 70h, al
        in al, 71h
        mov bh, al
        mov ax, bx
        mov si, offset EM
        add si, 22
        call WRDTODEC
        mov dx, offset EM
        mov ah, 09h
        int 21h

;MCB-s
        mov ax, 0
        mov ah, 52h
        int 21h
        mov cx, es:[bx-2]
        mov es, cx
```

```

mcb:
    mov     dx, offset McbL
    mov     ah, 09h
    int     21h

;MCBtype
    mov     dx, offset Tab
    mov     ah, 09h
    int     21h
    mov     dx, offset typ
    mov     ah, 09h
    int     21h

    mov     al, es:[00h]
    call    WRBYTE
    mov     dx, offset Hex
    mov     ah, 09h
    int     21h
    mov     dx, offset Tab
    mov     ah, 09h
    int     21h

;MCBaddress
    mov     dx, offset sect
    mov     ah, 09h
    int     21h
    mov     bx, es:[01h]
    mov     al, bh
    call    WrByte
    mov     al, bl
    call    WrByte

    mov     dx, offset Hex
    mov     ah, 09h
    int     21h
    mov     dx, offset Tab
    mov     ah, 09h
    int     21h

;MCBsize
    mov     ax, es:[03h]
    mov     bx, 10h
    mul     bx
    mov     si, offset siz
    add     si, 11
    call    WRDTODEC
    mov     dx, offset siz
    mov     ah, 09h
    int     21h
    mov     dx, offset EndLine
    mov     ah, 09h
    int     21h
    mov     dx, offset Tab
    mov     ah, 09h
    int     21h

;info
    mov     dx, offset info

```

```

        mov ah, 09h
        int 21h
        mov bx, 0
fin:
        mov dl, es:[08h + bx]
        mov ah, 02h
        int 21h
        inc bx
        cmp bx, 8h
        jl fin

        mov dx, offset EndLine
        mov ah, 09h
        int 21h
;5Ah - last one
        mov al, es:[00h]
        cmp al, 5Ah
        je ext

        mov bx, es
        add bx, es:[03h]
        inc bx
        mov es, bx
        jmp mcb

ext:
        xor al, al
        mov ah, 4Ch
        int 21h

;////////////////////////////////////

```

```

WrByte PROC near
        push ax
        push dx
        push cx
        call BYTETOHEX
        xor cx, cx
        mov ch, ah
        mov dl, al
        mov ah, 02h
        int 21h
        mov dl, ch
        mov ah, 02h
        int 21h
        pop cx
        pop dx
        pop ax
        ret
WrByte ENDP

```

```

TetrToHex PROC near
        and al, 0Fh
        cmp al, 09h
        jbe next
        add al, 07h
next:
        add al, 30h
        ret
TetrToHex ENDP

```



```

ByteToHex PROC near
    push cx
    mov ah,al
    call TetrToHex
    xchg al,ah
    mov cl,4
    shr al,cl
    call TetrToHex
    pop cx
    ret
ByteToHex ENDP

WrdToHex PROC near
    push bx
    mov bh,ah
    call ByteToHex
    mov [di],ah
    dec di
    mov [di],al
    dec di
    mov al,bh
    call ByteToHex
    mov [di],ah
    dec di
    mov [di],al
    pop bx
    ret
WrdToHex ENDP

WrdToDec PROC NEAR
    push cx
    push dx
    mov cx,10
loop_b:
    div cx
    or     dl,30h
    mov [si],dl
    dec si
    xor dx,dx
    cmp ax,10
    jae loop_b
    cmp al,00h
    je     endl
    or     al,30h
    mov [si],al
endl:
    pop dx
    pop cx
    ret
WrdToDec ENDP

OURMEM:
STT     ENDS
        END START

```