

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. 7381

Преподаватель

Кортев Ю.В.

Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системные данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Основные теоретические положения.

Учет занятой и свободной памяти ведется при помощи списка блоков управления памятью MCB. MCB занимает 16 байт и располагается всегда с адреса кратного 16 и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет.

MCB имеет следующую структуру:

Смещение	Длина поля (байт)	Содержание 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 исключительной 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.

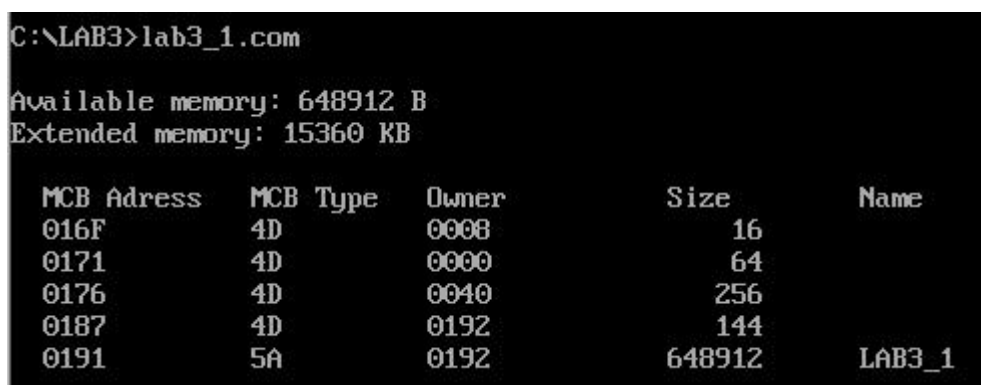
Адрес первого MCB хранится во внутренней структуре MS DOS, называемой “List of Lists”. Доступ к указателю на эту структуру можно получить, используя функцию 52h “Get Lists of Lists” int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX-2] и есть 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, 71; 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
```

Порядок выполнения работы

Были написаны и отлажены программные модули, для всех 4 вариантов программы. На рис. 1-4 представлены результаты запуска программы.



C:\LAB3>lab3_1.com

Available memory: 648912 B
Extended memory: 15360 KB

MCB Address	MCB Type	Owner	Size	Name
016F	4D	0008	16	
0171	4D	0000	64	
0176	4D	0040	256	
0187	4D	0192	144	
0191	5A	0192	648912	LAB3_1

Рисунок 1 – Шаг 1

```
Available memory: 648912 B
Extended memory: 15360 KB
```

MCB Address	MCB Type	Owner	Size	Name
016F	4D	0008	16	
0171	4D	0000	64	
0176	4D	0040	256	
0187	4D	0192	144	
0191	4D	0192	13136	LAB3_2
04C7	5A	0000	635760	♦Fè Gèðδ

Рисунок 2 – Шаг 2

```
Available memory: 648912 B
Extended memory: 15360 KB
```

MCB Address	MCB Type	Owner	Size	Name
016F	4D	0008	16	
0171	4D	0000	64	
0176	4D	0040	256	
0187	4D	0192	144	
0191	4D	0192	13248	LAB3_3
04CE	4D	0192	65536	LAB3_3
14CF	5A	0000	570096	

Рисунок 3 – Шаг 3

```
Available memory: 648912 B
Extended memory: 15360 KB
Failed!
```

MCB Address	MCB Type	Owner	Size	Name
016F	4D	0008	16	
0171	4D	0000	64	
0176	4D	0040	256	
0187	4D	0192	144	
0191	4D	0192	13600	LAB3_4
04E4	5A	0000	635296	lãx PøF>

Рисунок 4 – Шаг 4

Вывод

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

Ответы на контрольные вопросы

1. Что означает «доступный объём памяти?»

Доступный объём памяти - часть оперативной памяти, выделенная программе для работы.

2. Где МСВ блок вашей программы в списке?

В первом случае 2 МСВ блока: предпоследний(адрес 187h) – блок памяти переменных среды, и последний(адрес 191h) – программный блок.

Во втором случае блоки МСВ располагаются по эти же адресам.

В третьем случае добавляется еще один блок МСВ – дополнительный. Он находится по адресу 04EBB.

В четвёртом случае программа запрашивает память до того, как освобождает неиспользуемую – и при обработке завершения функций ядра возникает ошибка.

3. Какой размер памяти занимает программа в каждом случае?

В первом случае всю свободную память (64 8912 б).

Во втором случае только необходимый программе объём памяти (648 912-635 296 – 16 = 13 600б).

В третьем случае 648 912-569 632-65 536-32 = 13 712б.

В четвертом случае 648912-634720-16 = 14 176б.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД
TESTPC SEGMENT
ASSUME CS:TESTPC, DS:TESTPC,
ES:NOTHING, SS:NOTHING
ORG 100H

START: JMP BEGIN

;
; _____
; Данные

AvailableMemory db 0dh,0ah,'Available memory:
B',0dh,0ah,'\$'

ExtendedMemory db 'Extended memory:
KB',0dh,0ah,'\$'

TableHead db 0dh,0ah,' MCB Adress MCB Type
Owner Size Name ',0dh,0ah,'\$'
MCB db '
,0dh,0ah,'\$'

;
; _____
; Процедуры

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
push cx
mov ah,al
call TETR_TO_HEX
xchg al,ah
mov cl,4
shr al,cl
call TETR_TO_HEX
pop cx
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

BYTE_TO_DEC PROC near

```
    push cx
    push dx
    xor ah,ah
    xor dx,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_1
    or al,30h
    mov [si],al
end_1: pop dx
    pop cx
    ret
```

BYTE_TO_DEC ENDP

WRD_TO_DEC PROC near

```
    push cx
    push dx
    push ax
```

```

    mov cx,10
loop_wd:
    div cx
    or dl,30h
    mov [si],dl
    dec si
    xor dx,dx
    cmp ax,10
    jae loop_wd
    cmp ax,00h
    jbe end_1_2
    or al,30h
    mov [si],al
end_1_2:
    pop ax
    pop dx
    pop cx
    ret
WRD_TO_DEC ENDP

```

Print PROC near

```

    push ax
    mov ah,09h
    int 21h
    pop ax
    ret

```

Print ENDP

PrintAvailableMemory PROC near

```

    push ax
    push bx
    push dx
    push si

    mov ah,04Ah
    mov bx,0FFFFh
    int 21h
    mov ax,10h
    mul bx
    lea si,AvailableMemory
    add si,25
    call WRD_TO_DEC
    lea dx,AvailableMemory
    call Print

```



```

    pop si
    pop dx
    pop bx
    pop ax
    ret
PrintAvailableMemory ENDP

```

```

PrintExtendedMemorySize PROC near
    push ax
    push bx
    push dx
    push si

    mov al,30h ;чтение младшего байта
    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   ;стение старшего байта размера
расширенной памяти
    mov ah,al
    mov al,bl
    sub dx,dx
    lea si,ExtendedMemory
    add si,21
    call WRD_TO_DEC
    lea dx,ExtendedMemory
    call Print

    pop si
    pop dx
    pop bx
    pop ax
    ret
PrintExtendedMemorySize ENDP

```

```

PrintMCB PROC near
    ; Address
    lea di,MCB
    mov ax,es
    add di,5

```

```

call WRD_TO_HEX

; Type
lea di,MCB
add di,15
xor ah,ah
mov al,es:[0]
call BYTE_TO_HEX
mov [di],al
inc di
mov [di],ah

; Owner
lea di,MCB
mov ax,es:[1]
add di,29
call WRD_TO_HEX

; Size
lea di,MCB
mov ax,es:[3]
mov bx,10h
mul bx
add di,46
push si
mov si,di
call WRD_TO_DEC
pop si

; Name
lea di,MCB
add di,53
mov bx,0
case_print:
    mov dl,es:[bx+8]
    mov [di],dl
    inc di
    inc bx
    cmp bx,8
    jne case_print
    mov ax,es:[3]
    mov bl,es:[0]
    ret
PrintMCB ENDP

```

```

PrintMemoryManagementUnits PROC near
    lea dx,TableHead
    call Print
    mov ah,52h
    int 21h
    sub bx,2h
    mov es,es:[bx]
case:
    call PrintMCB
    lea dx,MCB
    call Print
    mov cx,es
    add ax,cx
    inc ax
    mov es,ax
    cmp bl,4Dh
    je case
    ret
PrintMemoryManagementUnits ENDP

```

```

; _____
; Код

```

```

BEGIN:
    call PrintAvailableMemory
    call PrintExtendedMemorySize
    call PrintMemoryManagementUnits

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

```

```

TESTPC ENDS
END START

```