**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. 6381 |  | Попов Н.В. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Постановка задачи.**

Цель работы: Исследование организации управления памятью, а также структур данных и работы функций управления памятью ядра операционной системы.

Описание функций:

1. TETR\_TO\_HEX: вспомогательная функция для работы функции BYTE\_TO\_HEX.
2. BYTE\_TO\_HEX: переводит число AL в коды символов 16-ой с/с, записывая получившееся в bl и bh.
3. WRD\_TO\_HEX: переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры.
4. PRINT\_STR: выводит строку, помещенную в DX.
5. PRINT\_SIZE: переводит из Кб в байты число АХ и выводит его.
6. AMOUNT\_OF\_AVAILABLE\_MEMORY: выводит количество доступной памяти.
7. EXTENDED\_MEMORY\_SIZE: выводит размер расширенной памяти.
8. CHAIN\_OF\_MEMORY\_CONTROL\_BLOCKS: выводит цепочку блоков управления памятью.
9. FREE\_MEM: освобождает память, которая не занята программой, и обрабатывает корректное завершение.
10. MEMORY\_ALLOCATION: запрашивает 64 Кб памяти и обрабатывает корректное завершение.
11. TEST\_ERROR: проводит проверку на корректное завершение функций ядра.

Последовательность действий, выполняемых утилитой:

1. Шаг 1:
   1. Определение и печать количества доступной памяти.
   2. Определение и печать размера расширенной памяти.
   3. Определение и печать цепочки блоков управления памятью.



Рисунок 1. Работа программы L3\_1.COM(Шаг 1).

1. Шаг 2:
   1. Освобождение памяти, которую программа не занимает.
   2. Пункты а-с шага 1.

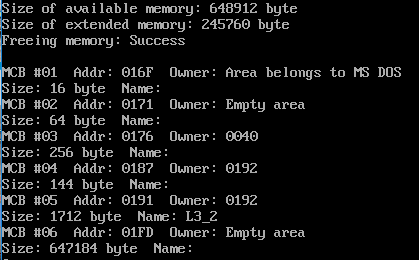


Рисунок 2. Работа программы L3\_2.COM(Шаг 2).

1. Шаг 3:
   1. Освобождение памяти, которую программа не занимает.
   2. Запрашивание 64х Кб памяти.
   3. Пункты а-с шага 1.

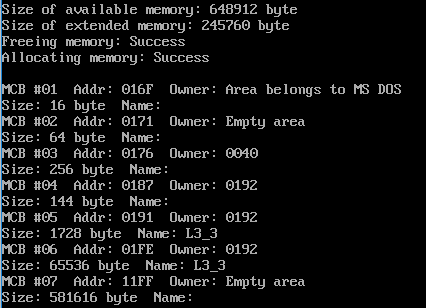


Рисунок 3. Работа программы L3\_3.COM(Шаг 3).

1. Шаг 4:
   1. Запрашивание 64х Кб памяти.
   2. Освобождение памяти, которую программа не занимает.
   3. Пункты а-с шага 1.

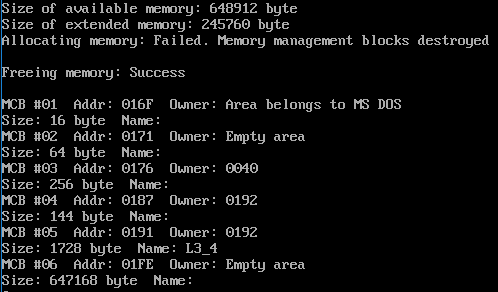


Рисунок 4. Работа программы L3\_4.COM(Шаг 4).

**Заключение.**

В процессе выполнения данной лабораторной работы была исследована организация управления памятью, структур данных и работы функций управления памятью ядра операционной системы.

**Ответы на контрольные вопросы.**

1. **Что означает «доступный объём памяти»?**

~~Доступная память – это память, занимаемая и используемая программой.~~

**Освобождаем память как во втором случае: у нас объем доступной памяти уменьшился, ибо программа занимает и использует памяти меньше?**

Объем доступной памяти остается прежним (648912 байт), несмотря на то, что программа освобождает память, которую не занимает. Таким образом, доступная память – это максимальный объем памяти, который программа может использовать.

1. **Где MCB блок Вашей программы в списке?**

~~MCB блок текущей программы находится в конце списка.~~

**Что показывает на принадлежность MCB процессу? Для 3й программы Ваш ответ не верен. Так же вопрос: чем являются 144 байта, которые находятся стабильно до MCB Вашей программы?**

В программах L3\_1.com, L3\_2.com и L3\_4.com MCB блок имеет адрес 0191h, в третьей программе два MCB блока: первый имеет адрес 0191h, а второй – 01FEh. В каждой из программ присутствует еще один блок MCB, имеющий адрес 0187h и размер 144 байт. Это блок управления памятью для области среды программы.

1. **Какой размер памяти занимает программа в каждом случае?**

~~Программа занимает всю наличную память. Однако с помощью функции 4Аh можно оставить программе ровно столько памяти сколько ей требуется. В нашем случае размер памяти – 648912 байт.~~

**Первый вопрос: является ли размер MCB - частью той памяти, что выделяется? Второй: если во втором случае у нас освобождается память, то логично предположить, что программа - это остальное. Так что ответ не верный, а расчеты должны быть по каждому случаю.**

MCB не входит в ту память, которая выделяется программе.

В первой программе память составляет 648912 байт.

Во второй – 1712 б.

В третьей – 67 264 б.

В четвертой – 1728 б.

**Приложение А**

**Исходный код L2.asm**

PROG SEGMENT

ASSUME CS:PROG, DS:PROG, ES:NOTHING, SS:NOTHING

ORG 100H

START:

jmp main

;Данные

avail\_mem db 13,10,"Size of available memory: $"

exten\_mem db 13,10,"Size of extended memory: $"

str\_MCB db "MCB #0 Addr: Owner: $"

str\_size db 13, 10, "Size: $"

str\_own\_name db " Name: $"

str\_own\_1 db "Empty area$"

str\_own\_2 db "Area belongs to OS XMS UMB driver $"

str\_own\_3 db "Area of excluded upper driver memory $"

str\_own\_4 db "Area belongs to MS DOS $"

str\_own\_5 db "Area occuped by control block 386MAX UMB $"

str\_own\_6 db "Area blocked 386MAX $"

str\_own\_7 db "Area belongs 386MAX UMB$"

str\_owner db " $"

str\_Byte db " byte$"

str\_enter db 13,10,"$"

fr\_mem db "Freeing memory: $"

al\_mem db "Allocating memory: $"

er7 db "Failed. Memory management blocks destroyed",13,10,"$"

er8 db "Failed. Not enough memory",13,10,"$"

er9 db "Failed. ES contains invalid address",13,10,"$"

er\_unk db "Failed. Unknown error",13,10,"$"

success db "Success",13,10,"$"

;---------------------Procedures-------------------

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 ;перевод в 16 с/с 16-ти разрядного числа

push BX ;AX - число, DI - адрес последего символа

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 ; перевод в 10с/с, SI - адрес поля младшей цифры

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\_l

or AL,30h

mov [SI],AL

end\_l:

pop DX

pop CX

ret

BYTE\_TO\_DEC ENDP

;--------------------------------------------------

PRINT\_STR PROC near ;Печать строки, помещенной в DX

push ax

mov ah,09h

int 21h

pop ax

ret

PRINT\_STR ENDP

;--------------------------------------------------

PRINT\_SIZE PROC near

mov bx,10h

mul bx

mov bx,0ah

xor cx,cx

del:

div bx

push dx

inc cx

xor dx,dx

cmp ax,0

jnz del

writeSymb:

pop dx

or dl,30h

mov ah,02h

int 21h

loop writeSymb

ret

PRINT\_SIZE ENDP

;--------------------------------------------------

AMOUNT\_OF\_AVAILABLE\_MEMORY PROC NEAR

mov ah,4Ah

mov bx,0FFFFh

int 21h

mov ax,bx

lea dx,avail\_mem

call PRINT\_STR

call PRINT\_SIZE

lea dx,str\_Byte

call PRINT\_STR

ret

AMOUNT\_OF\_AVAILABLE\_MEMORY ENDP

;--------------------------------------------------

EXTENDED\_MEMORY\_SIZE PROC NEAR

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

lea dx,exten\_mem

call PRINT\_STR

call PRINT\_SIZE

lea dx,str\_Byte

call PRINT\_STR

ret

EXTENDED\_MEMORY\_SIZE ENDP

;--------------------------------------------------

CHAIN\_OF\_MEMORY\_CONTROL\_BLOCKS PROC NEAR

mov ah,52h

int 21h

mov ax,es:[bx-2]

mov es,ax

xor cx,cx

inc cx

lea dx, str\_enter

call PRINT\_STR

nextMCB:

; lea dx, string\_enter

; call PRINT\_STR

lea si, str\_MCB

add si, 6

mov al,cl

push cx

call BYTE\_TO\_DEC

mov ax,es

lea di,str\_MCB

add di,18

call WRD\_TO\_HEX

xor ah,ah

mov al,es:[0]

push ax

mov ax,es:[1]

lea dx,str\_MCB

call PRINT\_STR

cmp ax,0000h

je g1

cmp ax,0006h

je g2

cmp ax,0007h

je g3

cmp ax,0008h

je g4

cmp ax,0FFFAh

je g5

cmp ax,0FFFDh

je g6

cmp ax,0FFFEh

je g7

lea di,str\_owner

add di, 3

call WRD\_TO\_HEX

lea dx,str\_owner

call PRINT\_STR

jmp go

g1:

lea dx,str\_own\_1

call PRINT\_STR

jmp go

g2:

lea dx,str\_own\_2

call PRINT\_STR

jmp go

g3:

lea dx,str\_own\_3

call PRINT\_STR

jmp go

g4:

lea dx,str\_own\_4

call PRINT\_STR

jmp go

g5:

lea dx,str\_own\_5

call PRINT\_STR

jmp go

g6:

lea dx,str\_own\_6

call PRINT\_STR

jmp go

g7:

lea dx,str\_own\_7

call PRINT\_STR

go:

mov ax,es:[3]

lea dx,str\_size

call PRINT\_STR

call PRINT\_SIZE

lea dx,str\_Byte

call PRINT\_STR

lea dx,str\_own\_name

call PRINT\_STR

mov cx,8

xor di,di

write:

mov dl,es:[di+8]

mov ah,02h

int 21h

inc di

loop write

mov ax,es:[3]

mov bx,es

add bx,ax

inc bx

mov es,bx

pop ax

pop cx

inc cx

cmp al,5ah

je exit

cmp al,4dh

jne exit

lea dx,str\_enter

call PRINT\_STR

jmp nextMCB

exit:

ret

CHAIN\_OF\_MEMORY\_CONTROL\_BLOCKS ENDP

;--------------------------------------------------

FREE\_MEM PROC NEAR

lea dx,fr\_mem

call PRINT\_STR

lea BX, newstk

mov CL,04h

add BX,10Fh

shr BX, CL

mov AH,4Ah

int 21h

call TEST\_ERROR

ret

FREE\_MEM ENDP

;--------------------------------------------------

MEMORY\_ALLOCATION PROC NEAR

lea dx,al\_mem

call PRINT\_STR

mov ah,48h

mov bx,1000h

int 21h

call TEST\_ERROR

ret

MEMORY\_ALLOCATION ENDP

;--------------------------------------------------

TEST\_ERROR PROC NEAR

jnc ok

cmp ax,07h

lea dx,er7

call PRINT\_STR

jmp ex

cmp ax,08h

lea dx,er8

call PRINT\_STR

jmp ex

cmp ax,09h

lea dx,er9

call PRINT\_STR

jmp ex

lea dx,er\_unk

call PRINT\_STR

jmp ex

ok:

lea dx,success

call PRINT\_STR

ex:

ret

TEST\_ERROR ENDP

;------------------------CODE----------------------

main:

call FREE\_MEM

call MEMORY\_ALLOCATION

call AMOUNT\_OF\_AVAILABLE\_MEMORY

call EXTENDED\_MEMORY\_SIZE

call CHAIN\_OF\_MEMORY\_CONTROL\_BLOCKS

xor AL,AL

mov AH,4Ch

int 21H

dw 64 dup(?)

newstk=$

PROG ENDS

END START