**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. 7383 |  | Корякин М.П. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2019

**Цель работы:** Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованной 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 .

В лабораторной работе исследуются структуры данных и работа функций управления памятью ядра операционной системы.

**Описание функций и структур данных.**

* AV\_MEM - Cтрока, информирующая о том, что дальше выведется размер доступной памяти.
* WORD\_BYTE – Строка для хранения размера доступной памяти.
* EX\_MEM - Строка, информирующая о том, что дальше выведется размер расширенной памяти.
* CHAR\_KB - Строка для хранения размера расширенной памяти.
* FOR\_LMCB - Строка, хранящая названия столбцов таблицы, в которую будут выводиться данные о MCB.
* LIST\_MCB - Строка для хранения данных о MCB.
* STRENDL - Строка, переводящая курсор на начало новой строки.
* NON – строка, информирующая о том, что при выделении памяти произошла ошибка.
* FIRST – стартовая функция.
* MAIN - распечатывает количество доступной памяти, распечатывает размер расширенной памяти, выводит цепочку блоков управления памятью.
* PRINT - вызывает функцию печати строки.
* BYTE\_TO\_HEX - переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah.
* TETR\_TO\_HEX - вспомогательная функция для работы функции BYTE\_TO\_HEX.
* WRD\_TO\_HEX - переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры.
* BYTE\_TO\_DEC - переводит байт из AL в десятичную с/с и записывает получившееся число по адресу si, начиная с младшей цифры.

**Результаты работ написанных программ.**

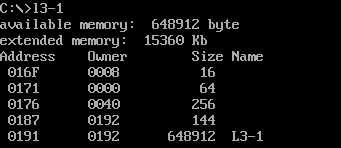
****

Рисунок 1 – результат работы первоначальной программы

Была написана программа, которая выводит следуюущую информацию: количество доступной памяти, размер расширенной памяти, цепочку блоков управления памятью.

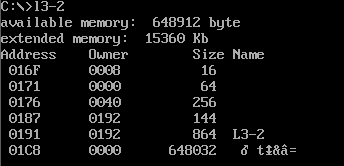


Рисунок 2 – результат работы программы с первой модификацией

Проведена первая модификация первоначальной программы, таким образом, что она освобождает память, которую не занимает.

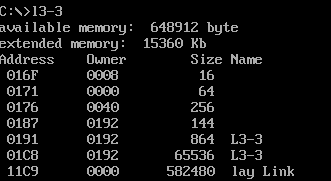


Рисунок 3 – результат работы программы со второй модификацией

Проведена вторая модификация программы, таким образом, что после освобождение ненужной памяти программа дополнительно выделяет 64Кб.

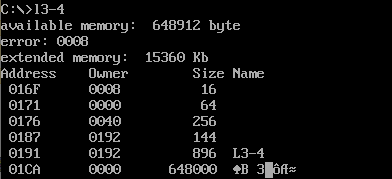
****

Рисунок 4 – результат работы программы с третьей модификацией.

Последняя модификация программы – третья. Здесь программа пытается выделить 64Кб до освобождения памяти. Но это не получилось, поэтому вывелось дополнительно сообщение об ошибке.

**Вывод.**

В лабораторной работе были исследованы структуры данных и работа функций управления памятью ядра операционной системы.

**Ответы на контрольные вопросы.**

1. Что означает «доступный объем памяти»?

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

1. Где MCB блок Вашей программы в списке?

Блок первой программы расположен в конце списка (см. рис. 1). Блок первой модификации программы - предпоследняя строка списка (см. рис. 2). В последней строке расположен блок освобожденной памяти. Блок второй модификации программы расположен в пятой строке, после него идут блоки выделенной по запросу памяти и свободной памяти соответственно (см. рис. 3). Блок третьей модификации программы - предпоследняя строка списка (см. рис. 4)

1. Какой размер памяти занимает программа в каждом случае?

В первоначальная программа занимает всю выделенную память: 648912 байт. Во втором случае 864 байт. В третьем случае 864+65536 =66400 байт, в четвертом 896 байт.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММ**

**L3-1.asm:**

TESTPC SEGMENT

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

ORG 100H

START: JMP FIRST

AV\_MEM db 'available memory:'

WORD\_BYTE db ' byte',0DH,0AH,'$'

EX\_MEM db 'extended memory:'

CHAR\_KB db ' Kb',0DH,0AH,'$'

FOR\_LMCB db 'Address Owner Size Name',0DH,0AH,'$'

LIST\_MCB db ' $'

STRENDL db 0DH,0AH,'$'

PRINT PROC

push ax

mov ah,09h

int 21h

pop ax

ret

PRINT ENDP

MAIN PROC

;print available memory

mov ah,4Ah

mov bx,0FFFFh

int 21h

mov ax,bx

mov bx,16

mul bx

mov si,offset WORD\_BYTE+7

call TO\_DEC

mov dx,offset AV\_MEM

call PRINT

;print extended memory

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

xor dx,dx

mov si,offset CHAR\_KB+6

call TO\_DEC

mov dx,offset EX\_MEM

call PRINT

;List of memory control block

mov dx,offset FOR\_LMCB

call PRINT

mov ah,52h

int 21h

mov bx,es:[bx-2]

mov es,bx

CYCLE:

mov ax,es

mov di,offset LIST\_MCB+4

call WRD\_TO\_HEX

mov ax,es:[01h]

mov di,offset LIST\_MCB+14

call WRD\_TO\_HEX

mov ax,es:[03h]

mov si,offset LIST\_MCB+26

mov dx, 0

mov bx, 10h

mul bx

call TO\_DEC

mov dx,offset LIST\_MCB

call PRINT

mov cx,8

mov bx,8

mov ah,02h

CYCLE2:

mov dl,es:[bx]

inc bx

int 21h

loop CYCLE2

mov dx,offset STRENDL

call PRINT

mov ax,es

inc ax

add ax,es:[03h]

mov bl,es:[00h]

mov es,ax

push bx

mov ax,' '

mov bx,offset LIST\_MCB

mov [bx+19],ax

mov [bx+21],ax

mov [bx+23],ax

pop bx

cmp bl,4Dh

je CYCLE

ret

MAIN 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

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\_l

or AL,30h

mov [SI],AL

end\_l: pop DX

pop CX

ret

BYTE\_TO\_DEC ENDP

TO\_DEC PROC near

push CX

push DX

mov CX,10

loop\_bd2: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd2

cmp AL,00h

je end\_l2

or AL,30h

mov [SI],AL

end\_l2: pop DX

pop CX

ret

TO\_DEC ENDP

FIRST:

call MAIN

mov AH,4Ch

int 21H

TESTPC ENDS

END START

**L3-2.asm:**

TESTPC SEGMENT

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

ORG 100H

START: JMP FIRST

AV\_MEM db 'available memory:'

WORD\_BYTE db ' byte',0DH,0AH,'$'

EX\_MEM db 'extended memory:'

CHAR\_KB db ' Kb',0DH,0AH,'$'

FOR\_LMCB db 'Address Owner Size Name',0DH,0AH,'$'

LIST\_MCB db ' $'

STRENDL db 0DH,0AH,'$'

PRINT PROC

push ax

mov ah,09h

int 21h

pop ax

ret

PRINT ENDP

MAIN PROC

push ax

push bx

push cx

push dx

push es

;print available memory

mov ah,4Ah

mov bx,0FFFFh

int 21h

mov ax,bx

mov bx,16

mul bx

mov si,offset WORD\_BYTE+7

call TO\_DEC

mov dx,offset AV\_MEM

call PRINT

mov bx,offset stc\_p

add bx,0Fh

push cx

mov cl,4

shr bx,cl

pop cx

xor al,al

mov ah, 4ah

int 21h

;print extended memory

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

xor dx,dx

mov si,offset CHAR\_KB+6

call TO\_DEC

mov dx,offset EX\_MEM

call PRINT

;List of memory control block

mov dx,offset FOR\_LMCB

call PRINT

mov ah,52h

int 21h

mov bx,es:[bx-2]

mov es,bx

CYCLE:

mov ax,es

mov di,offset LIST\_MCB+4

call WRD\_TO\_HEX

mov ax,es:[01h]

mov di,offset LIST\_MCB+14

call WRD\_TO\_HEX

mov ax,es:[03h]

mov si,offset LIST\_MCB+26

xor dx, dx

mov bx, 10h

mul bx

call TO\_DEC

mov dx,offset LIST\_MCB

call PRINT

mov cx,8

mov bx,8

mov ah,02h

CYCLE2:

mov dl,es:[bx]

inc bx

int 21h

loop CYCLE2

mov dx,offset STRENDL

call PRINT

mov ax,es

inc ax

add ax,es:[03h]

mov bl,es:[00h]

mov es,ax

push bx

mov ax,' '

mov bx,offset LIST\_MCB

mov [bx+19],ax

mov [bx+21],ax

mov [bx+23],ax

pop bx

cmp bl,4Dh

je CYCLE

pop es

pop dx

pop cx

pop bx

pop ax

ret

MAIN 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

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\_l

or AL,30h

mov [SI],AL

end\_l: pop DX

pop CX

ret

BYTE\_TO\_DEC ENDP

TO\_DEC PROC near

push CX

push DX

mov CX,10

loop\_bd2: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd2

cmp AL,00h

je end\_l2

or AL,30h

mov [SI],AL

end\_l2: pop DX

pop CX

ret

TO\_DEC ENDP

FIRST:

mov sp,offset stc\_p

call MAIN

xor al, al

mov AH,4Ch

int 21H

dw 64 dup (?)

stc\_p=$

TESTPC ENDS

END START

**L3-3.asm:**

TESTPC SEGMENT

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

ORG 100H

START: JMP FIRST

AV\_MEM db 'available memory:'

WORD\_BYTE db ' byte',0DH,0AH,'$'

EX\_MEM db 'extended memory:'

CHAR\_KB db ' Kb',0DH,0AH,'$'

FOR\_LMCB db 'Address Owner Size Name',0DH,0AH,'$'

LIST\_MCB db ' $'

STRENDL db 0DH,0AH,'$'

PRINT PROC

push ax

mov ah,09h

int 21h

pop ax

ret

PRINT ENDP

MAIN PROC

push ax

push bx

push cx

push dx

push es

;print available memory

mov ah,4Ah

mov bx,0FFFFh

int 21h

mov ax,bx

mov bx,16

mul bx

mov si,offset WORD\_BYTE+7

call TO\_DEC

mov dx,offset AV\_MEM

call PRINT

;clean excess mem

mov bx,offset stc\_p

add bx,0Fh

push cx

mov cl,4

shr bx,cl

pop cx

xor al,al

mov ah, 4ah

int 21h

mov bx, 4096

mov ah, 48h

int 21h

;print extended memory

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

xor dx,dx

mov si,offset CHAR\_KB+6

call TO\_DEC

mov dx,offset EX\_MEM

call PRINT

;List of memory control block

mov dx,offset FOR\_LMCB

call PRINT

mov ah,52h

int 21h

mov bx,es:[bx-2]

mov es,bx

CYCLE:

mov ax,es

mov di,offset LIST\_MCB+4

call WRD\_TO\_HEX

mov ax,es:[01h]

mov di,offset LIST\_MCB+14

call WRD\_TO\_HEX

mov ax,es:[03h]

mov si,offset LIST\_MCB+26

xor dx, dx

mov bx, 10h

mul bx

call TO\_DEC

mov dx,offset LIST\_MCB

call PRINT

mov cx,8

mov bx,8

mov ah,02h

CYCLE2:

mov dl,es:[bx]

inc bx

int 21h

loop CYCLE2

mov dx,offset STRENDL

call PRINT

mov ax,es

inc ax

add ax,es:[03h]

mov bl,es:[00h]

mov es,ax

push bx

mov ax,' '

mov bx,offset LIST\_MCB

mov [bx+19],ax

mov [bx+21],ax

mov [bx+23],ax

pop bx

cmp bl,4Dh

je CYCLE

pop es

pop dx

pop cx

pop bx

pop ax

ret

MAIN 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

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\_l

or AL,30h

mov [SI],AL

end\_l: pop DX

pop CX

ret

BYTE\_TO\_DEC ENDP

TO\_DEC PROC near

push CX

push DX

mov CX,10

loop\_bd2: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd2

cmp AL,00h

je end\_l2

or AL,30h

mov [SI],AL

end\_l2: pop DX

pop CX

ret

TO\_DEC ENDP

FIRST:

mov sp,offset stc\_p

call MAIN

;xor al, al

mov AH,4Ch

int 21H

dw 64 dup (?)

stc\_p=$

TESTPC ENDS

END START

**L3-4.asm:**

TESTPC SEGMENT

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

ORG 100H

START: JMP FIRST

AV\_MEM db 'available memory:'

WORD\_BYTE db ' byte',0DH,0AH,'$'

EX\_MEM db 'extended memory:'

CHAR\_KB db ' Kb',0DH,0AH,'$'

FOR\_LMCB db 'Address Owner Size Name',0DH,0AH,'$'

LIST\_MCB db ' $'

STRENDL db 0DH,0AH,'$'

NON db 'error: ',0Dh,0Ah,'$'

PRINT PROC

push ax

mov ah,09h

int 21h

pop ax

ret

PRINT ENDP

MAIN PROC

push ax

push bx

push cx

push dx

push es

;print available memory

mov ah,4Ah

mov bx,0FFFFh

int 21h

mov ax,bx

mov bx,16

mul bx

mov si,offset WORD\_BYTE+7

call TO\_DEC

mov dx,offset AV\_MEM

call PRINT

;clean excess mem

mov BX,4096

mov AH,48h

int 21h

;Обработка завершения функции ядра:

jnc FRE

mov DI,OFFSET NON

add DI,10

call WRD\_TO\_HEX

mov DX,OFFSET NON

call PRINT

int 21h

FRE:

mov bx,offset stc\_p

add bx,0Fh

push cx

mov cl,4

shr bx,cl

pop cx

xor al,al

mov ah, 4ah

int 21h

;print extended memory

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

xor dx,dx

mov si,offset CHAR\_KB+6

call TO\_DEC

mov dx,offset EX\_MEM

call PRINT

;List of memory control block

mov dx,offset FOR\_LMCB

call PRINT

mov ah,52h

int 21h

mov bx,es:[bx-2]

mov es,bx

CYCLE:

mov ax,es

mov di,offset LIST\_MCB+4

call WRD\_TO\_HEX

mov ax,es:[01h]

mov di,offset LIST\_MCB+14

call WRD\_TO\_HEX

mov ax,es:[03h]

mov si,offset LIST\_MCB+26

xor dx, dx

mov bx, 10h

mul bx

call TO\_DEC

mov dx,offset LIST\_MCB

call PRINT

mov cx,8

mov bx,8

mov ah,02h

CYCLE2:

mov dl,es:[bx]

inc bx

int 21h

loop CYCLE2

mov dx,offset STRENDL

call PRINT

mov ax,es

inc ax

add ax,es:[03h]

mov bl,es:[00h]

mov es,ax

push bx

mov ax,' '

mov bx,offset LIST\_MCB

mov [bx+19],ax

mov [bx+21],ax

mov [bx+23],ax

pop bx

cmp bl,4Dh

je CYCLE

pop es

pop dx

pop cx

pop bx

pop ax

ret

MAIN 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

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\_l

or AL,30h

mov [SI],AL

end\_l: pop DX

pop CX

ret

BYTE\_TO\_DEC ENDP

TO\_DEC PROC near

push CX

push DX

mov CX,10

loop\_bd2: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd2

cmp AL,00h

je end\_l2

or AL,30h

mov [SI],AL

end\_l2: pop DX

pop CX

ret

TO\_DEC ENDP

FIRST:

mov sp,offset stc\_p

call MAIN

;xor al, al

mov AH,4Ch

int 21H

dw 64 dup (?)

stc\_p=$

TESTPC ENDS

END START