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. 8382 _____ Чирков С.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

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.

Выполнение работы.

В ходе работы был написан и отлажен программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

1. Количество доступной памяти.
2. Размер расширенной памяти.
3. Выводит цепочку блоков управления памятью.

Количество доступной памяти было получено с использованием функции 4Ah с заведомо большей памятью в регистре BX, чем может предоставить ОС.

Размер расширенной памяти был получен при обращении к ячейкам CMOS по адресам 30h, 31h.

Для получения адреса первого MCB программа обращается к внутренней структуре MS DOS, называемой “List of Lists”. Доступ к указателю на эту структуру можно получить используя функцию 52h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX-2] и есть адрес самого первого MCB.

Результат выполнения программы показан на рисунке 1.

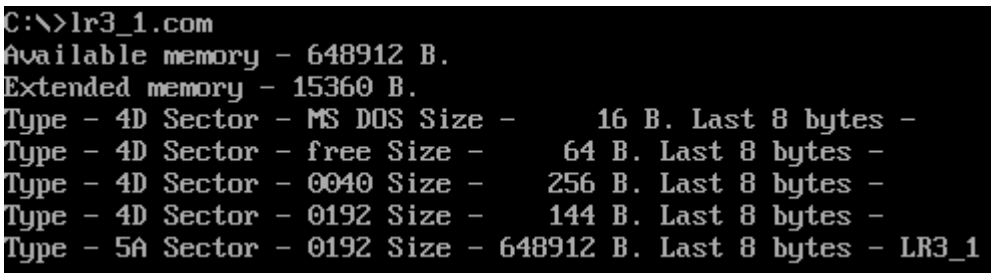

Рисунок 1. Результат выполнения программы lr3_1.com

Далее в программу было добавлено освобождение памяти, которую она не занимает с помощью функции 4Ah. Результат выполнения программы показан на рисунке 2.

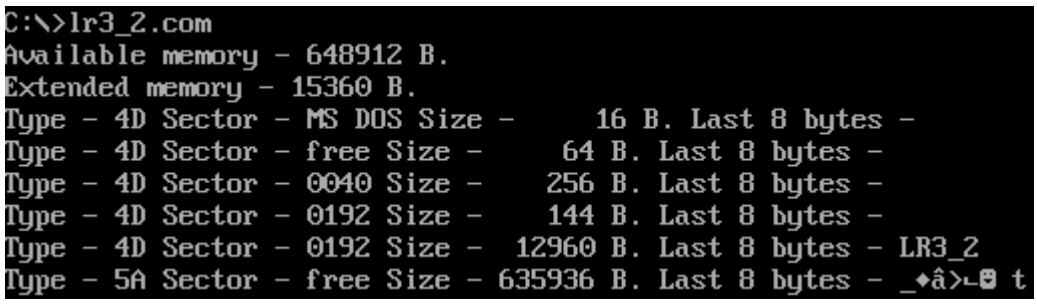

Рисунок 2. Результат выполнения программы lr3_2.com

Затем в программу было добавлено выделение 64Кб памяти, с помощью функции 48h. Результат выполнения программы показан на рисунке 3.

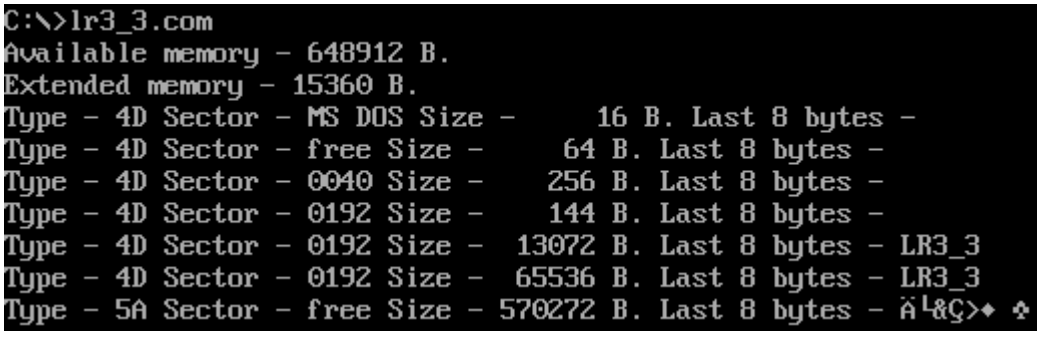

Рисунок 3. Результат выполнения программы lr3_3.com

Также в начальный вариант программы было добавлено выделение 64Кб памяти, с помощью функции 48h (до освобождения памяти). Результат выполнения программы показан на рисунке 4 – произошла ошибка.

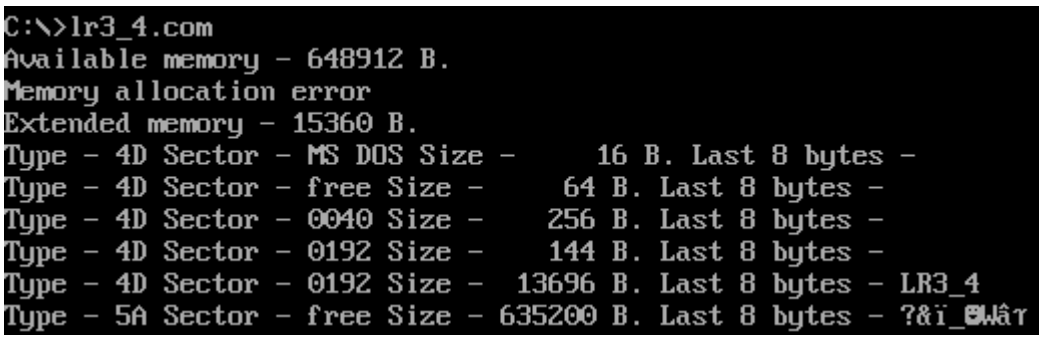

Рисунок 4. Результат выполнения программы lr3_4.com

Контрольные вопросы.

1. Что означает «доступный объем памяти»?

Это объем памяти, к которому программа может обратиться для ее выполнения.

2. Где MCB блок Вашей программы в списке?

Четвертый и пятый блоки в первой, второй и последней программах. В третьей программе четвертый, пятый и шестой (блоки с 0192h адресом PSP владельца участка памяти).

3. Какой размер памяти программа занимает в каждом случае?

Первая программа – 649056 байт.

Вторая программа – 13104 байт.

Третья программа – 78752 байта.

Последняя программа – 13840 байт.

Выводы.

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

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ LR3_1.ASM

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

typeln db 'Type - $'
sectorln db ' Sector - $'
mem db 'Available memory -        B.', 0DH,0AH, '$'
exmem db'Extended memory -       B.', 0DH,0AH, '$'
free db 'free', '$'
xms db 'OS XMS UMB', '$'
driver db 'excluded upper driver memory', '$'
msdos db 'MS DOS', '$'
occup db '386MAX UMB occupied', '$'
blocked db '386MAX UMB blocked', '$'
maxumb db '386MAX UMB', '$'
sector db '$'
mcbsize db ' Size -       $'
lbytes db ' B. Last 8 bytes - $'
endln db 0DH,0AH, '$'

PRINT PROC near
push CX
push DX

call BYTE_TO_HEX
mov CH, AH

mov DL, AL
mov AH, 02h
int 21h

mov DL, CH
mov AH, 02h
int 21h

pop DX
pop CX
ret
PRINT 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_DEC 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,0
jnz   loop_b
endl: pop   dx
pop   cx
ret
WRD_TO_DEC ENDP

BEGIN:
mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx
mov cx, 16
mul cx
mov si, offset mem + 24
call WRD_TO_DEC
mov dx, offset mem
mov ah, 09h
int 21h

xor ax, ax
xor dx, dx
```

```
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 exmem + 22
call WRD_TO_DEC
mov  dx, offset exmem
mov ah, 09h
int 21h

xor ax, ax
mov ah, 52h
int 21h
mov es, es:[bx-2]
mainloop:
mov  dx, offset typeln
mov ah, 09h
int 21h
mov al, es:[0]
call PRINT

mov  dx, offset sectorln
mov ah, 09h
int 21h
mov ax, es:[1]
mov DX, offset free
cmp ax, 0000h
je rdy
mov DX, offset xms
cmp ax, 0006h
je rdy
mov DX, offset driver
cmp ax, 0007h
je rdy
mov DX, offset msdos
cmp ax, 0008h
je rdy
mov DX, offset occup
cmp ax, 0fffah
je rdy
mov DX, offset blocked
cmp ax, 0fffdh
je rdy
```

```
mov DX, offset maxumb
cmp ax, 0fffeh
je rdy
mov DX, offset sector
xchg ah, al
mov cl, ah
call print
mov al, cl
call print
rdy:
mov AH,09h
int 21h


mov ax, es:[3]
mov cx, 16
mul cx
mov si, offset mcbsize + 13
call WRD_TO_DEC
mov dx, offset mcbsize
mov ah, 09h
int 21h
mov dx, offset lbytes
mov ah, 09h
int 21h

mov cx,8
mov si,8
mov ah, 2
whilenoteight:
mov dl, es:[si]
int 21h
inc si
loop whilenoteight

mov dx, offset endln
mov ah, 09h
int 21h

mov al, es:[0]
cmp al, 5Ah
je return

mov bx, es
add bx, es:[3]
inc bx
mov es, bx
jmp mainloop
```

```
return:
xor AL,AL
mov AH,4Ch
int 21H
TESTPC ENDS
END START;
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ LR3_2.ASM

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

typeln db 'Type - $'
sectorln db ' Sector - $'
mem db 'Available memory -        B.', 0DH,0AH, '$'
exmem db'Extended memory -       B.', 0DH,0AH, '$'
free db 'free', '$'
xms db 'OS XMS UMB', '$'
driver db 'excluded upper driver memory', '$'
msdos db 'MS DOS', '$'
occup db '386MAX UMB occupied', '$'
blocked db '386MAX UMB blocked', '$'
maxumb db '386MAX UMB', '$'
sector db '$'
mcbsize db ' Size -       $'
lbytes db ' B. Last 8 bytes - $'
endln db 0DH,0AH, '$'

PRINT PROC near
push CX
push DX

call BYTE_TO_HEX
mov CH, AH

mov DL, AL
mov AH, 02h
int 21h

mov DL, CH
mov AH, 02h
int 21h

pop DX
pop CX
ret
PRINT 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_DEC 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,0
jnz   loop_b
endl: pop   dx
pop   cx
ret
WRD_TO_DEC ENDP

BEGIN:
mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx
mov cx, 16
mul cx
mov si, offset mem + 24
call WRD_TO_DEC
mov dx, offset mem
mov ah, 09h
int 21h

mov ah, 4ah
mov bx, offset lr3end
```

```
int 21h

xor ax, ax
xor dx, dx
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 exmem + 22
call WRD_TO_DEC
mov   dx, offset exmem
mov ah, 09h
int 21h

xor ax, ax
mov ah, 52h
int 21h
mov es, es:[bx-2]
mainloop:
mov   dx, offset typeln
mov ah, 09h
int 21h
mov al, es:[0]
call PRINT

mov   dx, offset sectorln
mov ah, 09h
int 21h
mov ax, es:[1]
mov DX, offset free
cmp ax, 0000h
je rdy
mov DX, offset xms
cmp ax, 0006h
je rdy
mov DX, offset driver
cmp ax, 0007h
je rdy
mov DX, offset msdos
cmp ax, 0008h
je rdy
mov DX, offset occup
cmp ax, 0fffah
```

```
je rdy
mov DX, offset blocked
cmp ax, 0fffdh
je rdy
mov DX, offset maxumb
cmp ax, 0fffeh
je rdy
mov DX, offset sector
xchg ah, al
mov cl, ah
call print
mov al, cl
call print
rdy:
mov AH,09h
int 21h


mov ax, es:[3]
mov cx, 16
mul cx
mov si, offset mcbsize + 13
call WRD_TO_DEC
mov dx, offset mcbsize
mov ah, 09h
int 21h
mov dx, offset lbytes
mov ah, 09h
int 21h

mov cx,8
mov si,8
mov ah, 2
whilenoteight:
mov dl, es:[si]
int 21h
inc si
loop whilenoteight

mov dx, offset endln
mov ah, 09h
int 21h

mov al, es:[0]
cmp al, 5Ah
je return

mov bx, es
```

```
add bx, es:[3]
inc bx
mov es, bx
jmp mainloop

return:
xor AL,AL
mov AH,4Ch
int 21H
lr3end:
TESTPC ENDS
END START;
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ LR3_3.ASM

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

typeln db 'Type - $'
sectorln db ' Sector - $'
mem db 'Available memory -        B.', 0DH,0AH, '$'
exmem db'Extended memory -       B.', 0DH,0AH, '$'
free db 'free', '$'
xms db 'OS XMS UMB', '$'
driver db 'excluded upper driver memory', '$'
msdos db 'MS DOS', '$'
occup db '386MAX UMB occupied', '$'
blocked db '386MAX UMB blocked', '$'
maxumb db '386MAX UMB', '$'
sector db '$'
mcbsize db ' Size -       $'
lbytes db ' B. Last 8 bytes - $'
endln db 0DH,0AH, '$'

PRINT PROC near
push CX
push DX

call BYTE_TO_HEX
mov CH, AH

mov DL, AL
mov AH, 02h
int 21h

mov DL, CH
mov AH, 02h
int 21h

pop DX
pop CX
ret
PRINT 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_DEC 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,0
jnz   loop_b
endl: pop   dx
pop   cx
ret
WRD_TO_DEC ENDP

BEGIN:
mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx
mov cx, 16
mul cx
mov si, offset mem + 24
call WRD_TO_DEC
mov dx, offset mem
mov ah, 09h
int 21h

mov ah, 4ah
mov bx, offset lr3end
```

```
int 21h

mov bx, 1000h
mov ah, 48h
int 21h

xor ax, ax
xor dx, dx
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 exmem + 22
call WRD_TO_DEC
mov  dx, offset exmem
mov ah, 09h
int 21h

xor ax, ax
mov ah, 52h
int 21h
mov es, es:[bx-2]
mainloop:
mov  dx, offset typeln
mov ah, 09h
int 21h
mov al, es:[0]
call PRINT

mov  dx, offset sectorln
mov ah, 09h
int 21h
mov ax, es:[1]
mov DX, offset free
cmp ax, 0000h
je rdy
mov DX, offset xms
cmp ax, 0006h
je rdy
mov DX, offset driver
cmp ax, 0007h
je rdy
mov DX, offset msdos
```

```
cmp ax, 0008h
je rdy
mov DX, offset occup
cmp ax, 0fffah
je rdy
mov DX, offset blocked
cmp ax, 0fffdh
je rdy
mov DX, offset maxumb
cmp ax, 0fffeh
je rdy
mov DX, offset sector
xchg ah, al
mov cl, ah
call print
mov al, cl
call print
rdy:
mov AH,09h
int 21h


mov ax, es:[3]
mov cx, 16
mul cx
mov si, offset mcbsize + 13
call WRD_TO_DEC
mov dx, offset mcbsize
mov ah, 09h
int 21h
mov dx, offset lbytes
mov ah, 09h
int 21h

mov cx,8
mov si,8
mov ah, 2
whilenoteight:
mov dl, es:[si]
int 21h
inc si
loop whilenoteight

mov dx, offset endln
mov ah, 09h
int 21h

mov al, es:[0]
```

```
cmp al, 5Ah
je return

mov bx, es
add bx, es:[3]
inc bx
mov es, bx
jmp mainloop

return:
xor AL,AL
mov AH,4Ch
int 21H
lr3end:
TESTPC ENDS
END START;
```

ПРИЛОЖЕНИЕ Г

ИСХОДНЫЙ КОД ПРОГРАММЫ LR3_4.ASM

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

typeln db 'Type - $'
sectorln db ' Sector - $'
mem db 'Available memory -        B.', 0DH,0AH, '$'
exmem db'Extended memory -       B.', 0DH,0AH, '$'
free db 'free', '$'
xms db 'OS XMS UMB', '$'
driver db 'excluded upper driver memory', '$'
msdos db 'MS DOS', '$'
occup db '386MAX UMB occupied', '$'
blocked db '386MAX UMB blocked', '$'
maxumb db '386MAX UMB', '$'
sector db '$'
mcbsize db ' Size -       $'
lbytes db ' B. Last 8 bytes - $'
endln db 0DH,0AH, '$'
allerr db 'Memory allocation error ',0DH,0AH, '$'

PRINT PROC near
push CX
push DX

call BYTE_TO_HEX
mov CH, AH

mov DL, AL
mov AH, 02h
int 21h

mov DL, CH
mov AH, 02h
int 21h

pop DX
pop CX
ret
PRINT 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_DEC 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,0
jnz   loop_b
endl: pop   dx
pop   cx
ret
WRD_TO_DEC ENDP

BEGIN:
mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx
mov cx, 16
mul cx
mov si, offset mem + 24
call WRD_TO_DEC
mov dx, offset mem
mov ah, 09h
int 21h

mov bx, 1000h
```

```
mov ah, 48h
int 21h
jc exception
jmp nxt
exception:
mov dx, offset allerr
mov ah, 09h
int 21h
nxt:
mov ah, 4ah
mov bx, offset lr3end
int 21h

xor ax, ax
xor dx, dx
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 exmem + 22
call WRD_TO_DEC
mov   dx, offset exmem
mov ah, 09h
int 21h

xor ax, ax
mov ah, 52h
int 21h
mov es, es:[bx-2]
mainloop:
mov   dx, offset typeln
mov ah, 09h
int 21h
mov al, es:[0]
call PRINT

mov   dx, offset sectorln
mov ah, 09h
int 21h
mov ax, es:[1]
mov DX, offset free
cmp ax, 0000h
je rdy
```

```
mov DX, offset xms
cmp ax, 0006h
je rdy
mov DX, offset driver
cmp ax, 0007h
je rdy
mov DX, offset msdos
cmp ax, 0008h
je rdy
mov DX, offset occup
cmp ax, 0fffah
je rdy
mov DX, offset blocked
cmp ax, 0fffdh
je rdy
mov DX, offset maxumb
cmp ax, 0fffeh
je rdy
mov DX, offset sector
xchg ah, al
mov cl, ah
call print
mov al, cl
call print
rdy:
mov AH,09h
int 21h


mov ax, es:[3]
mov cx, 16
mul cx
mov si, offset mcbsize + 13
call WRD_TO_DEC
mov dx, offset mcbsize
mov ah, 09h
int 21h
mov dx, offset lbytes
mov ah, 09h
int 21h

mov cx,8
mov si,8
mov ah, 2
whilenoteight:
mov dl, es:[si]
int 21h
inc si
```

```
loop whilenoteight

mov dx, offset endln
mov ah, 09h
int 21h

mov al, es:[0]
cmp al, 5Ah
je return

mov bx, es
add bx, es:[3]
inc bx
mov es, bx
jmp mainloop

return:
xor AL,AL
mov AH,4Ch
int 21H
lr3end:
TESTPC ENDS
END START;
```