

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. 0382

Крючков А.М.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

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 не страничная 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, предусматривают и преобразуют этот список.

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

Постановка задачи.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

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

Адреса при выводе представляются шестнадцатеричными числами. Объем памяти функциями управления памятью выводится в параграфах. Необходимо преобразовать его в байты и выводить в виде десятичных чисел. Последние восемь байт МСВ выводятся как символы, не следует преобразовывать их в шестнадцатеричные числа.

Запустите программу и внимательно оцените результаты. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 2. Измените программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah прерывания 21h (пример в разделе «Использование функции 4AH»). Повторите эксперимент, запустив модифицированную программу.

Сравните выходные данные с результатами, полученными на предыдущем шаге. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

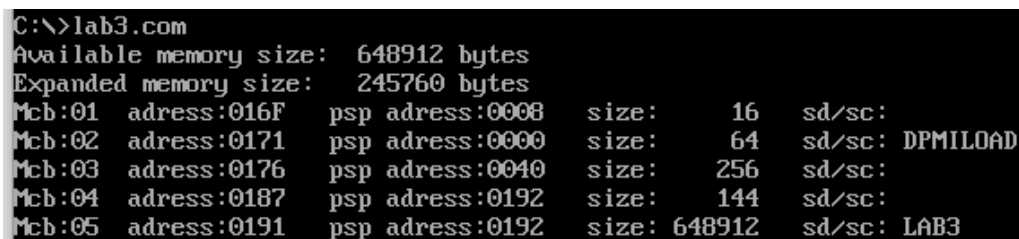
Шаг 3. Измените программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H. Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущих шагах. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 4. Измените первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти. Обязательно обрабатывайте завершение функций ядра, проверяя флаг CF. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 5. Оцените результаты, полученные на предыдущих шагах. Ответьте на контрольные вопросы и оформите отчет

Выполнение работы.

Шаг 1. Для распечатывания количества доступной памяти и размера расширенной памяти была написана процедура *memory*. Размер доступной памяти узнаётся при помощи прерывания 21h ah=4ah bx = 0ffff и распечатывается в десятичном формате при помощи процедуры *par_to_dec* и процедуры *print*. Размер расширенной памяти находится в ячейках 30h и 31h CMOS. Которая выводится аналогично количеству доступной памяти. Для вывода информации о каждом блоке mcb была написана одноимённая процедура. Из результата работы программы видим, что программа занимает 648912 байта.



```
C:\>lab3.com
Available memory size: 648912 bytes
Expanded memory size: 245760 bytes
Mcb:01 address:016F psp address:0008 size: 16 sd/sc:
Mcb:02 address:0171 psp address:0000 size: 64 sd/sc: DPMILOAD
Mcb:03 address:0176 psp address:0040 size: 256 sd/sc:
Mcb:04 address:0187 psp address:0192 size: 144 sd/sc:
Mcb:05 address:0191 psp address:0192 size: 648912 sd/sc: LAB3
```

Рисунок 1 – результат работы программы на шаге 1.

Шаг 2. Для освобождения неиспользуемой памяти была написана процедура *free*, которая использует прерывание 21h ah = 4ah. Замети, что количество занимаемой программой памяти сильно уменьшилось – 816 байт.

```
Available memory size: 648912 bytes
Expanded memory size: 245760 bytes
Mcb:01 address:016F psp address:0008 size: 16 sd/sc:
Mcb:02 address:0171 psp address:0000 size: 64 sd/sc: DPMILOAD
Mcb:03 address:0176 psp address:0040 size: 256 sd/sc:
Mcb:04 address:0187 psp address:0192 size: 144 sd/sc:
Mcb:05 address:0191 psp address:0192 size: 816 sd/sc: LAB3_2
Mcb:06 address:01C5 psp address:0000 size: 648080 sd/sc:
```

Рисунок 2 – результат работы программы на шаге 2.

Шаг 3. Для запроса дополнительной памяти была написана процедура *req_mem*, которая использует прерывание 21h функции 48h. Сначала была запрошена память, а затем освобождена не используемая память.

```
Available memory size: 648912 bytes
Expanded memory size: 245760 bytes
Mcb:01 address:016F psp address:0008 size: 16 sd/sc:
Mcb:02 address:0171 psp address:0000 size: 64 sd/sc: DPMILOAD
Mcb:03 address:0176 psp address:0040 size: 256 sd/sc:
Mcb:04 address:0187 psp address:0192 size: 144 sd/sc:
Mcb:05 address:0191 psp address:0192 size: 864 sd/sc: LAB3_3
Mcb:06 address:01C8 psp address:0192 size: 65536 sd/sc: LAB3_3
Mcb:07 address:11C9 psp address:0000 size: 582480 sd/sc: pTP
```

Рисунок 3 – результат работы программы после шага 3.

Шаг 4. Сначала вызывается *free*, а затем *req_mem*. Выводится сообщение об ошибке, так как при запросе об выделении дополнительной памяти у программы не осталось свободной памяти.

```
Available memory size: 648912 bytes
Expanded memory size: 245760 bytes
Memory allocation failed
Mcb:01 address:016F psp address:0008 size: 16 sd/sc:
Mcb:02 address:0171 psp address:0000 size: 64 sd/sc: DPMILOAD
Mcb:03 address:0176 psp address:0040 size: 256 sd/sc:
Mcb:04 address:0187 psp address:0192 size: 144 sd/sc:
Mcb:05 address:0191 psp address:0192 size: 864 sd/sc: LAB3_4
Mcb:06 address:01C8 psp address:0000 size: 648032 sd/sc:
```

Рисунок 4 – результат работы программы после шага 4.

Контрольные вопросы.

Сегментный адрес недоступной памяти:

1. Что означает «доступный объём памяти»?

Это количество оперативной памяти, которое может быть использовано программой.

2. Где MCB блок Вашей программы в списке?

Такие блоки при выводе помечаются в пункте “SD/SC”, как lab3_(номер шага).

3. Какой размер памяти занимает программа в каждом случае?

На первом шаге — 648912 байт, на втором — 816, на третьем — 66400, в четвёртом 864.

Выводы.

В ходе работы были изучены основные принципы структур данных и работы функций управления памятью ядра операционной системы.

ПРИЛОЖЕНИЕ А.

Исходный код модулей

lab3.asm:

```
code segment
assume cs:code, ds:code, es:nothing, ss:nothing
org 100h
start: jmp begin

free_mem db "Available memory size:          bytes",0dh,0ah,'$'
exp_mem  db "Expanded memory size:          bytes",0dh,0ah,'$'
mcbt     db "Mcb:0      adress:          psp adress:          size:
sd/sc: $"
newline  db 0dh,0ah,'$'

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

```

```

par_to_dec proc near
push ax
push bx
push dx
push si
add si, 7
mov bx,10h
mul bx
mov bx,10
write_loop:
div bx
or dl,30h
mov [si], dl
dec si
xor dx,dx
cmp ax,0h
jnz write_loop
pop si
pop dx
pop bx
pop ax
ret
par_to_dec endp

```

```

print proc near
push ax
mov ah, 09h
int 21h
pop ax
ret
print endp

```

```

memory proc near
push ax
push bx
push dx
push si

```

```

mov ah, 4ah
mov bx, 0ffffh
int 21h

```

```

mov ax, bx
mov dx, offset free_mem
mov si, dx
add si, 22
call par_to_dec
call print

```

```

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 dx, offset exp_mem
mov si, dx
add si, 22
call par_to_dec
call print

```

```

pop si
pop dx
pop bx
pop ax
ret
memory endp

```

```

mcb proc near
push ax
push bx
push cx
push dx
push di
push si

```

```

mov ah, 52h
int 21h
mov ax, es:[bx-2]
mov es, ax
xor cx, cx

```

```

mcb_block:
inc cx
mov al, cl
mov dx, offset mcbl
mov si, dx
add si, 5
call byte_to_dec

```

```

mov ax, es
mov di, si
add di, 14
call wrd_to_hex

```

```

mov ax, es:[1]
add di, 21
call wrd_to_hex

```



```

mov ax, es:[3]
mov si, di
add si, 11
call par_to_dec
call print

xor di, di
write_char:
mov dl, es:[di+8]
mov ah, 02h
int 21h
inc di
cmp di, 8
jl write_char
mov dx, offset newline
call print

```

```

mov al, es:[0]
cmp al, 5ah
je exit
mov bx, es
add bx, es:[3]
inc bx
mov es, bx
jmp mcb_block
exit:
pop si
pop di
pop dx
pop cx
pop bx
pop ax
ret
mcb endp

```

```

begin:
call memory
call mcb
xor al, al
mov ah, 4ch
int 21h
enddd:
code ends
end start

```

lab3_2.asm:

```

code segment
assume cs:code, ds:code, es:nothing, ss:nothing
org 100h
start: jmp begin

```

```

free_mem db "Available memory size:          bytes", 0dh, 0ah, '$'
exp_mem  db "Expanded memory size:          bytes", 0dh, 0ah, '$'
mcbt     db "Mcb:0      adress:                psp adress:                size:
sd/sc: $"
newline  db 0dh, 0ah, '$'

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

par_to_dec proc near

```

```

push ax
push bx
push dx
push si
add si, 7
mov bx, 10h
mul bx
mov bx, 10
write_loop:
div bx
or dl, 30h
mov [si], dl
dec si
xor dx, dx
cmp ax, 0h
jnz write_loop
pop si
pop dx
pop bx
pop ax
ret
par_to_dec endp

print proc near
push ax
mov ah, 09h
int 21h
pop ax
ret
print endp

memory proc near
push ax
push bx
push dx
push si

mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx
mov dx, offset free_mem
mov si, dx
add si, 22
call par_to_dec
call print

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 dx, offset exp_mem
mov si, dx
add si, 22
call par_to_dec

```

```

call print

pop si
pop dx
pop bx
pop ax
ret
memory endp

mcb proc near
push ax
push bx
push cx
push dx
push di
push si

mov ah, 52h
int 21h
mov ax, es:[bx-2]
mov es, ax
xor cx, cx

mcb_block:
inc cx
mov al, cl
mov dx, offset mcbt
mov si, dx
add si, 5
call byte_to_dec

mov ax, es
mov di, si
add di, 14
call wrd_to_hex

mov ax, es:[1]
add di, 21
call wrd_to_hex

mov ax, es:[3]
mov si, di
add si, 11
call par_to_dec
call print

xor di, di
write_char:
mov dl, es:[di+8]
mov ah, 02h
int 21h
inc di
cmp di, 8
jl write_char
mov dx, offset newline
call print

mov al, es:[0]
cmp al, 4dh
jne exit

```

```

mov bx, es
add bx, es:[3]
inc bx
mov es, bx
jmp mcb_block
exit:
pop si
pop di
pop dx
pop cx
pop bx
pop ax
ret
mcb endp

```

```

free proc near
push ax
push bx
push dx
mov ax, offset enddd
mov bx, 10h
xor dx, dx
div bx
add ax, 4
mov bx, ax
mov ah, 4ah
int 21h
pop dx
pop bx
pop ax
ret
free endp

```

```

begin:
call memory

```

```

call free

```

```

call mcb
xor al, al
mov ah, 4ch
int 21h
enddd:
code ends
end start

```

lab3_3.asm:

```

code segment
assume cs:code, ds:code, es:nothing, ss:nothing
org 100h
start: jmp begin

```

```

free_mem db "Available memory size:          bytes", 0dh, 0ah, '$'
exp_mem  db "Expanded memory size:          bytes", 0dh, 0ah, '$'
mcbt     db "Mcb:0      adress:                psp adress:                size:
sd/sc: $"
err DB "Memory allocation failed", 0DH, 0AH, '$'

```

```
newline      db 0dh,0ah,'$'
```

```
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

par_to_dec proc near
push ax
push bx
push dx
push si
add si, 7
mov bx, 10h
mul bx
mov bx, 10
write_loop:
div bx
or dl, 30h
mov [si], dl
dec si
xor dx, dx
cmp ax, 0h
jnz write_loop
pop si
pop dx
pop bx
pop ax
ret
par_to_dec endp

print proc near
push ax
mov ah, 09h
int 21h
pop ax
ret
print endp

memory proc near
push ax
push bx
push dx
push si

mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx
mov dx, offset free_mem
mov si, dx
add si, 22
call par_to_dec
call print

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 dx, offset exp_mem

```

```

mov si, dx
add si, 22
call par_to_dec
call print

pop si
pop dx
pop bx
pop ax
ret
memory endp

mcb proc near
push ax
push bx
push cx
push dx
push di
push si

mov ah, 52h
int 21h
mov ax, es:[bx-2]
mov es, ax
xor cx, cx

mcb_block:
inc cx
mov al, cl
mov dx, offset mcbl
mov si, dx
add si, 5
call byte_to_dec

mov ax, es
mov di, si
add di, 14
call wrd_to_hex

mov ax, es:[1]
add di, 21
call wrd_to_hex

mov ax, es:[3]
mov si, di
add si, 11
call par_to_dec
call print

xor di, di
write_char:
mov dl, es:[di+8]
mov ah, 02h
int 21h
inc di
cmp di, 8
jl write_char
mov dx, offset newline
call print

```



```

mov al, es:[0]
cmp al, 4dh
jne exit
mov bx, es
add bx, es:[3]
inc bx
mov es, bx
jmp mcb_block
exit:
pop si
pop di
pop dx
pop cx
pop bx
pop ax
ret
mcb endp

```

```

free proc near
push ax
push bx
push dx
mov ax, offset enddd
mov bx, 10h
xor dx, dx
div bx
add ax, 4
mov bx, ax
mov ah, 4ah
int 21h
pop dx
pop bx
pop ax
ret
free endp

```

```

req_mem proc near
push ax
push bx
push dx

```

```

mov bx, 1000h
mov ah, 48h
int 21h
jnc ex
mov dx, offset err
call print

```

```

ex:
pop dx
pop bx
pop ax
ret
req_mem endp

```

```

begin:
call memory

```

```

call free

```

```
call req_mem
```

```
call mcb  
xor al,al  
mov ah, 4ch  
int 21h  
enddd:  
code ends  
end start
```

```
lab3_4.asm:
```

```
code segment
```

```
assume cs:code, ds:code, es:nothing, ss:nothing
```

```
org 100h
```

```
start: jmp begin
```

```
free_mem db "Available memory size:      bytes",0dh,0ah,'$'  
exp_mem  db "Expanded memory size:      bytes",0dh,0ah,'$'  
mcbt     db "Mcb:0  adress:    psp adress:    size:      sd/sc: $"  
err      db "Memory allocation failed",0dh,0ah,'$'  
newline  db 0dh,0ah,'$'
```

```
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
wrд_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
```

```
par_to_dec proc near
push ax
push bx
push dx
push si
add si, 7
mov bx,10h
mul bx
mov bx,10
write_loop:
div bx
or dl,30h
```

```
mov [si], dl
dec si
xor dx,dx
cmp ax,0h
jnz write_loop
pop si
pop dx
pop bx
pop ax
ret
par_to_dec endp
```

```
print proc near
push ax
mov ah, 09h
int 21h
pop ax
ret
print endp
```

```
memory proc near
push ax
push bx
push dx
push si
```

```
mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx
mov dx, offset free_mem
mov si, dx
add si, 22
call par_to_dec
call print
```

```
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 dx, offset exp_mem
```

```
mov si, dx
add si, 22
call par_to_dec
call print
pop si
pop dx
pop bx
pop ax
ret
memory endp
```

```
mcb proc near
push ax
push bx
push cx
push dx
push di
push si
```

```
mov ah, 52h
int 21h
mov ax, es:[bx-2]
mov es, ax
xor cx, cx
```

```
mcb_block:
inc cx
mov al, cl
mov dx, offset mcbt
mov si, dx
add si, 5
call byte_to_dec
```

```
mov ax, es
mov di, si
add di, 14
call wrd_to_hex
```

```
mov ax, es:[1]
add di, 21
call wrd_to_hex
```

```
mov ax, es:[3]
mov si, di
add si, 11
call par_to_dec
```

call print

xor di,di
write_char:
mov dl, es:[di+8]
mov ah, 02h
int 21h
inc di
cmp di, 8
jl write_char
mov dx, offset newline
call print

mov al, es:[0]
cmp al, 4dh
jne exit
mov bx, es
add bx, es:[3]
inc bx
mov es, bx
jmp mcb_block
exit:
pop si
pop di
pop dx
pop cx
pop bx
pop ax
ret
mcb endp

free proc near
push ax
push bx
push dx
mov ax, offset enddd
mov bx, 10h
xor dx,dx
div bx
add ax, 4
mov bx, ax
mov ah, 4ah
int 21h
pop dx
pop bx

```

pop ax
ret
free endp

req_mem proc near
push ax
push bx
push dx

mov bx, 1000h
mov ah, 48h
int 21h
jnc ex
mov dx, offset err
call print
ex:
pop dx
pop bx
pop ax
ret
req_mem endp

begin:
call memory
call req_mem
call free
call mcb
xor al,al
mov ah, 4ch
int 21h
enddd:
code ends
end start

```