

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. 8383

Федоров И.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

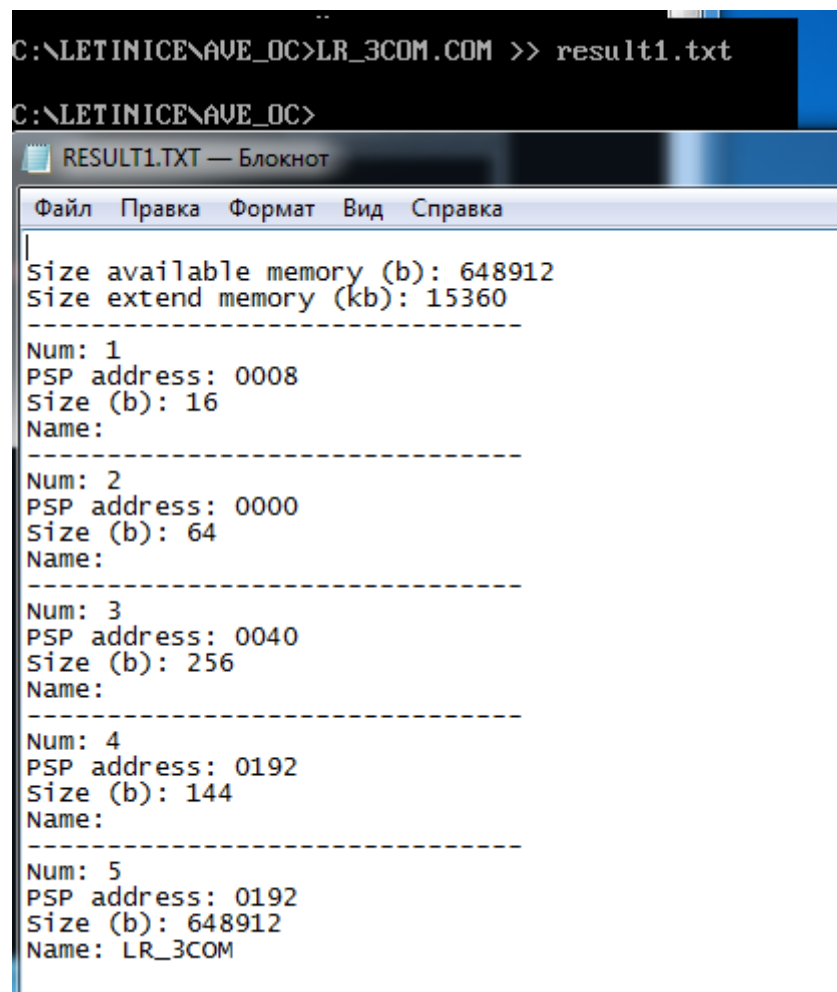
Исследование организации управления основной памятью, структуры данных и работы функций управления памятью ядра операционной системы.

Ход работы.

Был написан и отлажен программный модуль типа **.COM**, который выбирает и распечатывает следующую информацию:

1. Количество доступной памяти.
2. Размер расширенной памяти.
3. Цепочку блоков управления памятью (MCB).

Результаты работы программы представлен на рис. 1. Исходный код **.COM** модуля приведен в приложении А. Дальше результаты работы будут представлены в виде скриншота .txt файла, в который перенаправлен вывод.



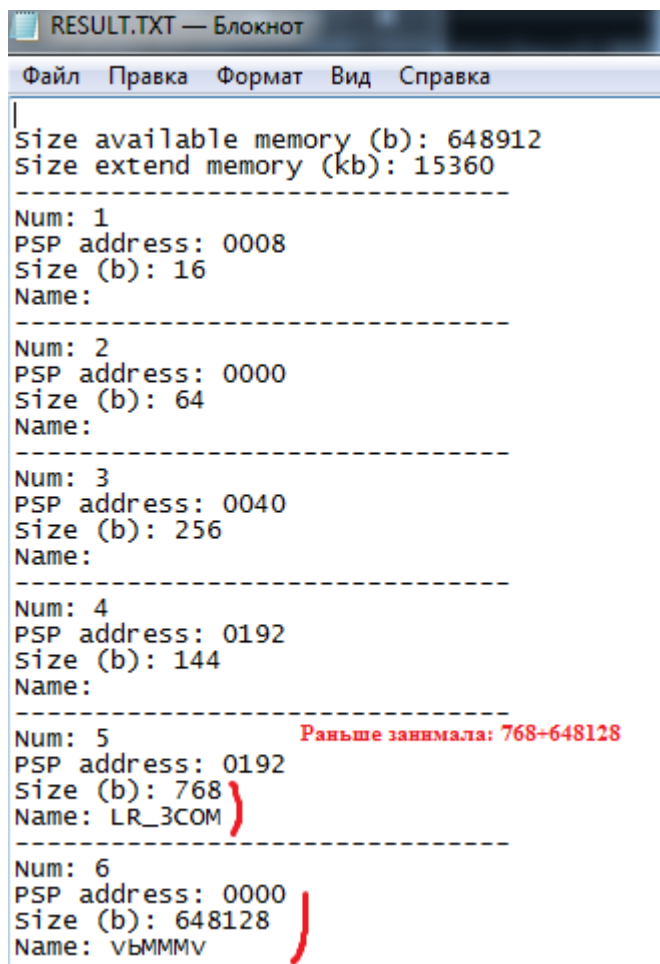
```
C:\LETINICE\AVE_OC>LR_3COM.COM >> result1.txt
C:\LETINICE\AVE_OC>

RESULT1.TXT — Блокнот
Файл  Правка  Формат  Вид  Справка

Size available memory (b): 648912
Size extend memory (kb): 15360
-----
Num: 1
PSP address: 0008
Size (b): 16
Name:
-----
Num: 2
PSP address: 0000
Size (b): 64
Name:
-----
Num: 3
PSP address: 0040
Size (b): 256
Name:
-----
Num: 4
PSP address: 0192
Size (b): 144
Name:
-----
Num: 5
PSP address: 0192
Size (b): 648912
Name: LR_3COM
```

Рисунок 1 – Результат работы .COM программы

В программу были внесены изменения таким образом, чтобы она освобождала не занятую ей память с помощью функции 4Ah прерывания 21h. Видно, что теперь программа LR_3COM занимает лишь необходимое количество памяти, а освобожденная теперь представлена в 6-м блоке.



```
RESULT.TXT — Блокнот
Файл  Правка  Формат  Вид  Справка

Size available memory (b): 648912
Size extend memory (kb): 15360
-----
Num: 1
PSP address: 0008
Size (b): 16
Name:
-----
Num: 2
PSP address: 0000
Size (b): 64
Name:
-----
Num: 3
PSP address: 0040
Size (b): 256
Name:
-----
Num: 4
PSP address: 0192
Size (b): 144
Name:
-----
Num: 5
PSP address: 0192
Size (b): 768
Name: LR_3COM
-----
Num: 6
PSP address: 0000
Size (b): 648128
Name: VBMMMV
```

Рисунок 2 – Результат работы программы с очисткой памяти

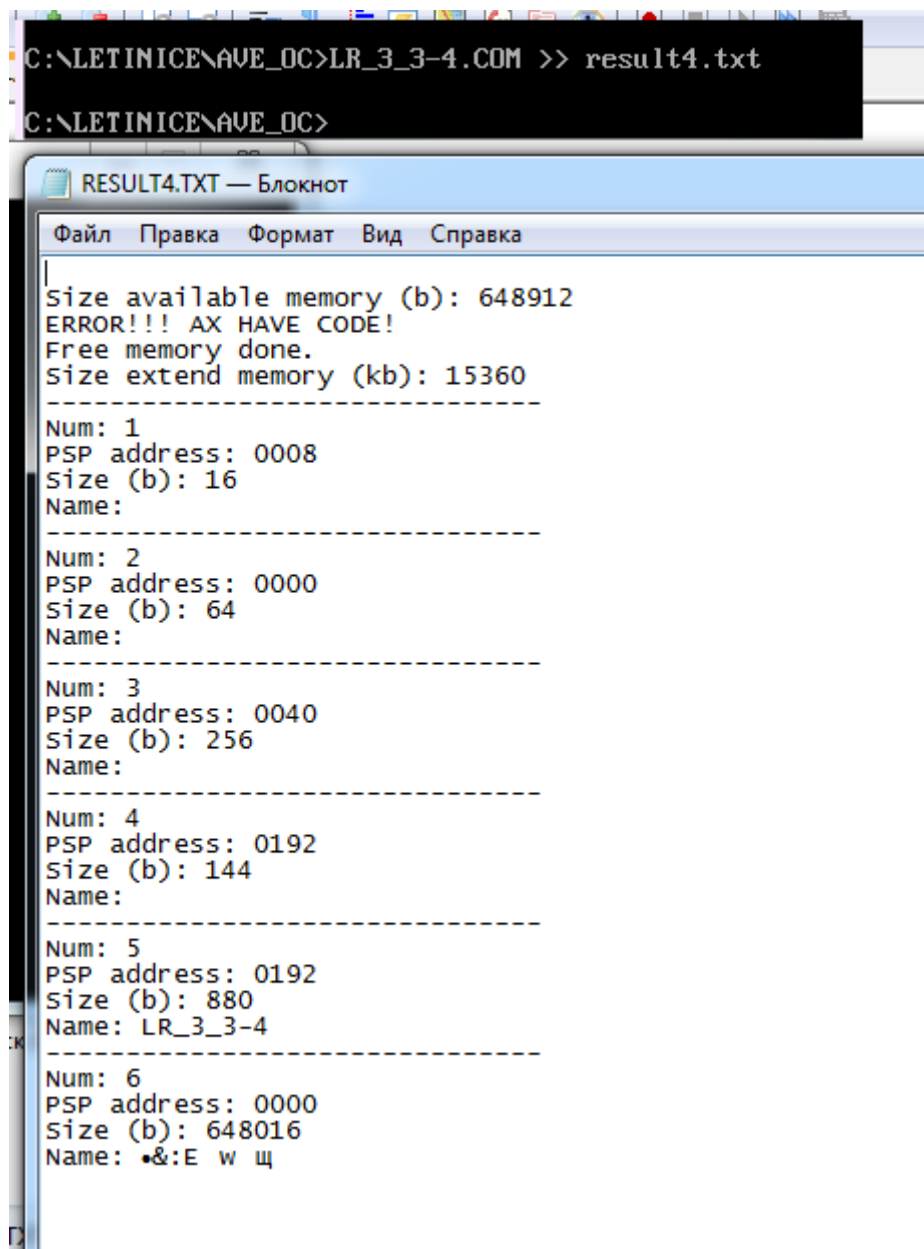
После этого в программу добавлен запрос 64Кб памяти после освобождения с помощью функции 48h прерывания 21h. 6-й блок отвечает за выделенные 64Кб.

```
C:\LETINICE\AVE_OC>LR_3_3-4.COM >> new.txt

NEW.TXT — Блокнот
Файл  Правка  Формат  Вид  Справка
Size available memory (b): 648912
Free memory done.
Memory asking is normal
Size extend memory (kb): 15360
-----
Num: 1
PSP address: 0008
Size (b): 16
Name:
-----
Num: 2
PSP address: 0000
Size (b): 64
Name:
-----
Num: 3
PSP address: 0040
Size (b): 256
Name:
-----
Num: 4
PSP address: 0192
Size (b): 144
Name:
-----
Num: 5
PSP address: 0192
Size (b): 880
Name: LR_3_3-4
-----
Num: 6
PSP address: 0192
Size (b): 65536
Name: LR_3_3-4
-----
Num: 7
PSP address: 0000
Size (b): 582464
Name: reserve
```

Рисунок 3 – Результат работы с запросом памяти.

Были внесены изменения в первоначальный вариант программы таким образом, чтобы программа запрашивала 64Кб памяти до освобождения памяти. Программа в этом случае выводит предупреждающее сообщение. Результат приведен на рис. 4.



```
C:\LETINICE\AVE_OC>LR_3_3-4.COM >> result4.txt
C:\LETINICE\AVE_OC>

RESULT4.TXT — Блокнот
Файл  Правка  Формат  Вид  Справка

Size available memory (b): 648912
ERROR!!! AX HAVE CODE!
Free memory done.
Size extend memory (kb): 15360
-----
Num: 1
PSP address: 0008
Size (b): 16
Name:
-----
Num: 2
PSP address: 0000
Size (b): 64
Name:
-----
Num: 3
PSP address: 0040
Size (b): 256
Name:
-----
Num: 4
PSP address: 0192
Size (b): 144
Name:
-----
Num: 5
PSP address: 0192
Size (b): 880
Name: LR_3_3-4
-----
Num: 6
PSP address: 0000
Size (b): 648016
Name: *&:E w щ
```

Рисунок 4 - Запрос памяти до освобождения

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

Сегментный адрес (равный 9FFFh) указывает на конец основной оперативной памяти DOS. Первые 640 Кбайт адресного пространства с сегментными адресами от 0000h до 9FFFh отведены под основную оперативную память (в которую может загружаться программа). За ней находится "верхняя память", зарезервированная DOS.

2) Где расположен этот адрес по отношению области памяти, отведенной программе?

Адрес расположен сразу за концом памяти, выделенной программе.

3) Можно ли в эту область памяти писать?

Эта область предназначена для размещения ПЗУ и других целей и зарезервирована, однако, т.к. в DOS нет механизма защиты памяти, то в нее можно писать и читать из нее.

Среда передаваемая программе

1) Что такое среда?

Среда представляет собой текстовый массив, состоящий из строк вида:

"имя=параметр", 0

Здесь имя и параметр - текстовые величины, байт 0 завершает каждую строку. Имеется несколько стандартных переменных среды, например PATH (определяет пути к каталогам, в которых система ищет исполняемый файл), PROMPT (задает вид подсказки при диалоге с ОС).

2) Когда создается среда? Перед запуском приложения или в другое время?

Интерпретатор команд COMMAND.COM имеет свою среду, которую называют корневой средой (она создается при загрузке DOS). При запуске программы эта среда по умолчанию копируется для программы, однако может быть изменена в соответствии с требованиями для этой программы.

3) Откуда берётся информация, записываемая в среду?

Для этого используется пакетный файл AUTOEXEC.BAT, который устанавливает ключевые переменные среды (таких как PATH, PROMPT).

Выводы.

В ходе работы были исследованы интерфейс управляющей программы и загрузочных модулей, префикс сегмента программы PSP, а так же среды, передаваемые программе.

ПРИЛОЖЕНИЕ А

КОД ДЛЯ СОМ ФАЙЛА

```

TESTPC      SEGMENT
             ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
             ORG 100H      ;обязательно!
START:      JMP MAIN

LEN_CLOSE_MEM EQU 30
LEN_ENV_SEG EQU 28
;TAIL db 83 DUP(?)
STR_CLOSE_MEM db 13,10, "Address of close memory:          $"
STR_ENV_SEG db 13,10, "Address of enviroment:              $"
STR_TAIL db 13,10, "Tail comand line: $"
STR_EMPTY_TAIL db " (nothing) $"
STR_ENVIROMENT_AREA db 13,10, "Enviroment: $"
STR_ENTER db 13,10, " $"
STR_PATH db 13,10, "Path: $"

;ПРОЦЕДУРЫ
;-----

WRITE_STR PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE_STR 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
; 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
; в AX - число, DI - адрес последнего символа
    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
; 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
;-----
;-----
; Funct lab_2

PRINT_ADDRESS_CLOSE_MEM PROC near
        push ax
        push dx
        mov ax, ds:[02h]                ;в PSP
        mov di, offset STR_CLOSE_MEM
        add di, LEN_CLOSE_MEM
        call WRD_TO_HEX
        mov dx, offset STR_CLOSE_MEM
        call WRITE_STR
        pop dx
        pop ax
        ret
PRINT_ADDRESS_CLOSE_MEM ENDP

PRINT_ADDRESS_ENVIROMENT PROC near
        push ax
        push dx
        mov ax, ds:[2Ch]                ; (44)
        mov di, offset STR_ENV_SEG
        add di, LEN_ENV_SEG
        call WRD_TO_HEX
        mov dx, offset STR_ENV_SEG
        call WRITE_STR
        pop dx

```



```

    pop ax
    ret
PRINT_ADDRESS_ENVIROMENT ENDP

```

```

PRINT_TAIL PROC near
    push ax
    push dx
    mov dx, offset STR_TAIL
    call WRITE_STR
    mov cx, 0
    mov cl, ds:[80h]          ;число СИМВОЛОВ В ХВОСТЕ
    cmp cl, 0
    je tail_empty
    mov di, 0
    xor dx,dx
print_tail_cycle:
    mov dl, ds:[81h+di]      ;ds+81h+di сделать потом
    mov ah,02H
    int 21h
    inc di
    loop print_tail_cycle
    jmp end_print
tail_empty:
    mov dx, offset STR_EMPTY_TAIL
    call WRITE_STR
end_print:
    pop dx
    pop ax
    ret
PRINT_TAIL ENDP

```

```

PRINT_PATH_ENVIROMENT PROC near
    push dx
    push ax
    push ds
    mov dx, offset STR_ENVIROMENT_AREA
    call WRITE_STR
    mov di, 0
    mov es, ds:[2Ch]
cycle_env:
    cmp byte ptr es:[di], 00h          ;mov dl, ds:[di] и сранивать dl с 0
    je enter_                          ;==
    mov dl, es:[di]
    mov ah, 02h
    int 21h
    inc di
    jmp cycle_env
enter_:
    inc di
    cmp word ptr es:[di], 0001h
    je path_
    mov dx, offset STR_ENTER
    call WRITE_STR
    jmp cycle_env
path_:
    inc di
    inc di
    mov DX, offset STR_PATH
    call WRITE_STR

```

```

cycle_p:
    cmp byte ptr es:[di], 00h
    je end_print_p
    mov dl, es:[di]
    mov ah, 02h
    int 21h
    inc di
    jmp cycle_p
end_print_p:
    pop dx
    pop ax
    pop ds
    ret
PRINT_PATH_ENVIROMENT ENDP

```

```

MAIN:
    call PRINT_ADDRESS_CLOSE_MEM
    call PRINT_ADDRESS_ENVIROMENT
    call PRINT_TAIL
    call PRINT_PATH_ENVIROMENT
    xor al, al
    mov AH, 4Ch
    int 21H
TESTPC ENDS
END START

```