

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

Шишкин И.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследовать структуры данных и работу функций управления памятью ядра операционной системы.

Ход работы.

Был написан .COM модуль, который выбирает и распечатывает следующую информацию: количество доступной памяти; размер расширенной памяти; цепочка блоков управления памятью. Результат выполнения программы представлен на рис. 1.

```
C:\>os3_1.com

Amount of available memory: E6D0
Extended memory size: 3C00
-----
Number 1
Area belongs to MS DOS
Area size: 0010
-----
Number 2
Free area
Area size: 0040
-----
Number 3
0040
Area size: 0100
-----
Number 4
0192
Area size: 0090
-----
Number 5
0192
Area size: E6D0      OS3_1
```

Рисунок 1 – Результат выполнения программы

Затем программа была изменена таким образом, чтобы она освобождала память, которую она не занимает. Результат выполнения измененной программы представлен на рис. 2.

```
C:\>os3_2.com  
Amount of available memory: E6D0  
Memory freed  
Extended memory size: 3C00
```

```
-----  
Number 1  
Area belongs to MS DOS  
Area size: 0010  
-----  
Number 2  
Free area  
Area size: 0040          DPMILOAD  
-----  
Number 3  
0040  
Area size: 0100  
-----  
Number 4  
0192  
Area size: 0090  
-----  
Number 5  
0192  
Area size: 0670          OS3_2  
-----  
Number 6  
Free area  
Area size: E050          to init  
-----
```

Рисунок 2 – Результат выполнения программы

Далее программа была еще раз изменена таким образом, чтобы после освобождения памяти, программа запрашивала 64 Кб памяти. Результат выполнения измененной еще раз программы представлен на рис. 3.

```

C:\>os3_3.com

Amount of available memory: E6D0
Memory freed
Memory has been allocated
Extended memory size: 3C00

-----
Number 1
Area belongs to MS DOS
Area size: 0010
-----
Number 2
Free area
Area size: 0040          DPMILOAD
-----
Number 3
0040
Area size: 0100
-----
Number 4
0192
Area size: 0090
-----
Number 5
0192
Area size: 06D0          OS3_3
-----
Number 6
0192
Area size: 0000          OS3_3
-----
Number 7
Free area
Area size: DFE0          ПДПД
-----

```

Рисунок 3 – Результат выполнения программы

В конце, в программе был изменен вызов функции: в начале вызывается запрос 64 Кб памяти, а затем освобождение памяти. Результат выполнения конечной программы представлен на рис. 4.

```
C:\>os3_4.com  
  
Amount of available memory: E6D0  
Memory has not been allocated  
Memory freed  
Extended memory size: 3C00
```

```
-----  
Number 1  
Area belongs to MS DOS  
Area size: 0010  
-----  
Number 2  
Free area  
Area size: 0040          DPMILOAD  
-----  
Number 3  
0040  
Area size: 0100  
-----  
Number 4  
0192  
Area size: 0090  
-----  
Number 5  
0192  
Area size: 06D0          OS3_4  
-----  
Number 6  
Free area  
Area size: DFF0          opy DOSX
```

Рисунок 4 – Результат выполнения программы

Контрольные вопросы.

1) Что означает "доступный объем памяти"?

Количество памяти, доступной для выполнения программы.

2) Где МСВ блок Вашей программы в списке?

В 1-ом случае это МСВ блок под номером 5.

Во 2-ом случае также под номером 5.

В 3-ем случае это МСВ блоки под номером 5 и 6.

В 4-ом случае под номером 5.

3) Какой размер памяти занимает программа в каждом случае?

В 1-ом случае программа занимает всю доступную память (E6D0h байт или в 10-ой системе 59088 байт).

Во 2-ом случае программа освободила память, которую она не занимает (670h байт или в 10-ой системе 1648 байт).

В 3-ем случае программа сначала освобождает память, а потом выделяет 64 Кб = 65536 байт. Так как после освобождения памяти программа стала занимать 1744 байт, то в итоге программа занимает 67280 Кб.

В 4-ом случае программа сначала выделяет 64 Кб = 65536 байт, а затем освобождает память. В итоге получается, что программа занимает 6D0h байт или в 10-ой системе 1744 байт.

Выводы.

В ходе выполнения лабораторной работы были исследованы структуры данных и работа функций управления памятью ядра ОС.

ПРИЛОЖЕНИЕ

КОД ПРОГРАММЫ OS3_4.asm

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

    AM_STR db 13, 10, "Amount of available memory:          $"
    EM_STR db 13, 10, "Extended memory size:                $"
    NUM_STR db 13, 10, "Number                                $"
    NEW_LINE db 13, 10, "-----$"
    FREE_AREA_STR db 13, 10, "Free area$"
    OS_XMSUMP_STR db 13, 10, "Area belongs to the driver OS XMS
UMB$"
    UPPER_MEMORY_STR db 13, 10, "Area is excluded upper driver
memory$"
    MS_DOS_STR db 13, 10, "Area belongs to MS DOS$"
    BUSY_386MAX_STR db 13, 10, "Area is occupied by the control
unit 386 MAX UMB$"
    BLOCK_386MAX_STR db 13, 10, "Area is blocked by 386 MAX$"
    BELONG_386MAX_STR db 13, 10, "Area belongs to the 386 MAX
UMB$"
    PSP_MEMORY_OWNER_STR db 13, 10, "          $"
    AREA_SIZE_STR db 13, 10, "Area size:                    $"
    MEMORY_F db 13, 10, "Memory freed$"
    MEMORY_ISNT_F db 13, 10, "Memory is not freed$"
    MEMORY_REQUEST_TRUE db 13, 10, "Memory has been allocated$"
    MEMORY_REQUEST_FALSE db 13, 10, "Memory has not been
allocated$"

;-----
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, в 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 10 с.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_1
    or AL,30h
    mov [SI],AL
end_1: pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP
;-----
PRINT PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT ENDP
;-----

```

AVAILABLE_MEMORY PROC near

push ax

push bx

push dx

mov di, offset AM_STR

add di, 33

mov ah, 4Ah

mov bx, 0FFFFh

int 21h

mov ax, bx

mov bx, 10h

mul bx

call WRD_TO_HEX

mov dx, offset AM_STR

call PRINT

pop dx

pop bx

pop ax

ret

AVAILABLE_MEMORY ENDP

;-----

EXTENDED_MEMORY PROC near

push ax

push bx

push dx

mov di, offset EM_STR

add di, 27

mov al, 30h

out 70h, al

in al, 71h

mov bl, al

```

mov al,31h
out 70h, al
in al, 71h
mov bh, ah
mov ah, al
mov al, bh

call WRD_TO_HEX
mov dx, offset EM_STR
call PRINT

```

```

pop dx
pop bx
pop ax
ret

```

EXTENDED_MEMORY ENDP

;-----

MCB PROC near

```

push ax
push bx
push cx
push di
push si

```

```

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

```

```

next_mcb:
    inc cx
    mov dx, offset NEW_LINE
    call PRINT

```

```
mov si, offset NUM_STR
add si, 9
mov ax, cx
push cx
call BYTE_TO_DEC
mov dx, offset NUM_STR
call PRINT
```

```
xor ax, ax
mov al, es:[0h]
push ax
mov ax, es:[1h]
```

```
cmp ax, 0h
je if_free_area
cmp ax, 6h
je if_driver
cmp ax, 7h
je if_upper_memory
cmp ax, 8h
je if_msdos
cmp ax, 0FFFAh
je if_386max_umb
cmp ax, 0FFFDh
je if_block_386max
cmp ax, 0FFFEh
je if_belongs_386max
```

```
xor dx, dx
mov di, offset PSP_MEMORY_OWNER_STR
add di, 5
call WRD_TO_HEX
mov dx, offset PSP_MEMORY_OWNER_STR
jmp end_of_01h
```

```

        pop ax

if_free_area:
        mov dx, offset FREE_AREA_STR
        jmp end_of_01h

if_driver:
        mov dx, offset OS_XMSUMP_STR
        jmp end_of_01h

if_upper_memory:
        mov dx, offset UPPER_MEMORY_STR
        jmp end_of_01h

if_msdos:
        mov dx, offset MS_DOS_STR
        jmp end_of_01h

if_386max_umb:
        mov dx, offset BUSY_386MAX_STR
        jmp end_of_01h

if_block_386max:
        mov dx, offset BLOCK_386MAX_STR
        jmp end_of_01h

if_belongs_386max:
        mov dx, offset BELONG_386MAX_STR
        jmp end_of_01h

end_of_01h:
        call PRINT
        mov di, offset AREA_SIZE_STR
        add di, 16

```

```

        mov ax, es:[3h]
        mov bx, 10h
        mul bx
        call WRD_TO_HEX
        mov dx, offset AREA_SIZE_STR
        call PRINT
        mov cx, 8
        xor si, si

end_of_mcb:
        mov dl, es:[si+8h]
        mov ah, 02h
        int 21h
        inc si
        loop end_of_mcb

        mov ax, es:[3h]
        mov bx, es
        add bx, ax
        inc bx
        mov es, bx
        pop ax
        pop cx
        cmp al, 5Ah
        je end_of_proc
        jmp next_mcb

end_of_proc:
        pop si
        pop di
        pop cx
        pop bx
        pop ax
ret

```

```

MCB ENDP

;-----
FREEING_UP_MEMORY PROC near
    push ax
    push bx
    push dx

    mov bx, offset STACK_END
    add bx, 10Fh
    shr bx, 4
    mov ah, 4Ah
    int 21h
    jnc TRUE_FREEING
    mov dx, offset MEMORY_ISNT_F
    jmp END_OF_FREEING

TRUE_FREEING:
    mov dx, offset MEMORY_F

END_OF_FREEING:
    call PRINT
    pop dx
    pop bx
    pop ax
    ret
FREEING_UP_MEMORY ENDP

;-----
MEMORY_REQUEST PROC near
    push ax
    push bx
    push dx

    mov bx, 1000h
    mov ah, 48h

```

```

    int 21h
    jnc TRUE_REQUEST
    mov dx, offset MEMORY_REQUEST_FALSE
    jmp END_OF_REQUEST

TRUE_REQUEST:
    mov dx, offset MEMORY_REQUEST_TRUE

END_OF_REQUEST:
    call PRINT
    pop dx
    pop bx
    pop ax
    ret
MEMORY_REQUEST ENDP
;-----
BEGIN:

    call AVAILABLE_MEMORY
    call MEMORY_REQUEST
    call FREEING_UP_MEMORY
    call EXTENDED_MEMORY
    call MCB

    xor AL, AL
    mov AH, 4Ch
    int 21h

STACK_TO_FREE:
    DW 128 dup(0)
STACK_END:
TESTPC ENDS
END START

```