

**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**

Студентка гр. 9382

Сорочина М.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

### **Цель работы.**

Исследование структур данных и работы функций управления памятью ядра операционной системы.

### **Сведения, использованные для составления программы.**

Учет занятой и свободной памяти ведется при помощи списка блоков управления памятью MCB (Memory Control Block). MCB занимает 16 байт (параграф) и располагается всегда с адреса кратного 16 (адрес сегмента ОП) и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет. Структура MCB представлена в табл. 1.

Смещение	Длина поля (байт)	Содержимое поля
00h	1	тип MCB: 5Ah, если последний в списке, 4Dh, если не последний
01h	2	Сегментный адрес PSP владельца участка памяти, либо 0000h - свободный участок, 0006h - участок принадлежит драйверу OS XMS UMB 0007h - участок является исключенной верхней памятью драйверов 0008h - участок принадлежит MS DOS FFFAh - участок занят управляющим блоком 386MAX UMB FFFDh - участок заблокирован 386MAX FFFEh - участок принадлежит 386MAX UMB
03h	2	Размер участка в параграфах
05h	3	Зарезервирован
08h	8	"SC" - если участок принадлежит MS DOS, то в нем системный код "SD" - если участок принадлежит MS DOS, то в нем системные данные

Табл. 1. Структура MCB.

По сегментному адресу и размеру участка памяти, контролируемого этим MCB можно определить местоположение следующего MCB в списке.

Адрес первого MCB хранится во внутренней структуре MS DOS,

называемой "List of Lists" (список списков). Доступ к указателю на эту структуру можно получить, используя функцию 52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX-2] и есть адрес самого первого MCB.

Размер расширенной памяти находится в ячейках 30h, 31h CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта.

### Ход работы.

Был написан и отлажен программный модуль типа .COM, который выводит на экран следующую информацию:

- 1) Количество доступной памяти
- 2) Размер расширенной памяти
- 3) Цепочку блоков управления памятью

На рис. 1 представлен вывод программы.

```
C:\>LAB3_1.COM
Available memory: 648912 b
Extended memory: 15360 kb
MCB:  1 Owner: MS DOS           Size:    16
last 8 bytes:
MCB:  2 Owner: free             Size:    64
last 8 bytes:
MCB:  3 Owner: 0040             Size:   256
last 8 bytes:
MCB:  4 Owner: 0192             Size:   144
last 8 bytes:
MCB:  5 Owner: 0192             Size: 648912
last 8 bytes: LAB3_1
```

Рис. 1. Результат выполнения lab3\_1.com

По рисунку видно, что программа занимает всю свободную память.

Далее был получен lab3\_2.com, в котором была освобождена память, которую программа не занимает. На рис. 2 представлен вывод программы.

```

C:\>LAB3_2.COM
Available memory: 648912 b
Extended memory: 15360 kb
MCB: 1 Owner: MS DOS Size: 16
last 8 bytes:
MCB: 2 Owner: free Size: 64
last 8 bytes:
MCB: 3 Owner: 0040 Size: 256
last 8 bytes:
MCB: 4 Owner: 0192 Size: 144
last 8 bytes:
MCB: 5 Owner: 0192 Size: 944
last 8 bytes: LAB3_2
MCB: 6 Owner: free Size: 647952
last 8 bytes: u0000 60 A0

```

Рис. 2. Результат выполнения lab3\_2.com

По рисунку видно, что программа занимает не всю свободную память. Освобожденная память находится в шестом блоке.

Далее был получен lab3\_3.com, в котором после освобождения памяти программа запрашивает 64 Кб памяти. На рис. 3 представлен вывод программы.

```

C:\>LAB3_3.COM
Available memory: 648912 b
No error
Extended memory: 15360 kb
MCB: 1 Owner: MS DOS Size: 16
last 8 bytes:
MCB: 2 Owner: free Size: 64
last 8 bytes:
MCB: 3 Owner: 0040 Size: 256
last 8 bytes:
MCB: 4 Owner: 0192 Size: 144
last 8 bytes:
MCB: 5 Owner: 0192 Size: 1008
last 8 bytes: LAB3_3
MCB: 6 Owner: 0192 Size: 65536
last 8 bytes: LAB3_3
MCB: 7 Owner: free Size: 582336
last 8 bytes: 1Error

```

Рис. 3. Результат выполнения lab3\_3.com

По рисунку видно, что дополнительно выделенная память относится к шестому блоку. Сегментный адрес PSP владельца участка памяти для пятого и шестого блока совпадают.

Далее был получен lab3\_4.com, в котором программа сначала запрашивает дополнительно 64 Кб, а потом освобождает память. На рис. 4 представлен вывод программы.

```
C:\>LAB3_4.COM
Available memory: 648912 b
Error
Extended memory: 15360 kb
MCB: 1 Owner: MS DOS Size: 16
last 8 bytes:
MCB: 2 Owner: free Size: 64
last 8 bytes:
MCB: 3 Owner: 0040 Size: 256
last 8 bytes:
MCB: 4 Owner: 0192 Size: 144
last 8 bytes:
MCB: 5 Owner: 0192 Size: 1008
last 8 bytes: LAB3_4
MCB: 6 Owner: free Size: 647888
last 8 bytes: %|| &8%u
```

Рис. 3. Результат выполнения lab3\_4.com

По рисунку видно, что при попытке выделить дополнительную память произошла ошибка, поэтому в отличие от предыдущего варианты программы дополнительная память не была выделена.

### Ответы на контрольные вопросы.

1. Что означает “доступный объем памяти”?

Максимальный объем памяти, который доступен для запуска и выполнения программ.

2. Где MCB блок Вашей программы в списке?

Для первой, второй и четвертой программ MCB блок находится на 4 и 5 местах в списке.

Для третьей к четвертому и пятому добавляется шестой.

3. Какой размер памяти занимает программа в каждом случае?

1) 144+648'912

2) 144+944

3)144+1'008+65'536

4)144+1'008

### **Выводы.**

В ходе выполнения данной работы были исследованы структуры данных и работа функций управления памятью ядра ОС. Были рассмотрены нестраничная память и способ управления динамическими разделами.

## ПРИЛОЖЕНИЕ А.

### Исходный код программы lab3\_1.asm.

```
TESTPC SEGMENT
        ASSUME     CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG      100H
START:   JMP      BEGIN
; Данные
AvailableMem db 'Available memory:  b',0DH,0AH,'$'
ExtMem      db 'Extended memory:  kb',0DH,0AH,'$'
MCB         db 'MCB:  $'
Owner       db 'Owner: $'
AreaSize    db 'Size:  $'
LastBytes   db 0DH,0AH,'last 8 bytes: $'
FREE        db ' free      $'
XMS         db ' OS XMS UMB    $'
TM          db ' driver memory $'
DOS         db ' MS DOS      $'
Busy        db ' busy by 386MAX UMB $'
Block       db ' blocked by 386MAX $'
OWN_386     db ' 386MAX UMB    $'
Empty       db '              $'
EndL        db 0Dh, 0Ah, '$'
; Процедуры
;-----
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 шест. 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
;-----
PRINT proc    near
        mov    ah, 09h
        int 21h
        ret
PRINT endp
;-----
WORD_TO_DEC PROC near
        push cx
        push dx
        mov cx, 10
loop1:
        div cx
        or     dl, 30h

```



```

        mov [si], dl
        dec si
        xor dx, dx
        cmp ax, 0
        jnz loop1

        pop dx
        pop cx
        ret
WORD_TO_DEC ENDP

;-----
; Код
BEGIN:
;available mem
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h

        mov ax, bx
        mov cx, 16
        mul cx
        mov si, offset AvailableMem + 23
        call word_to_dec
        mov dx, offset AvailableMem
        call print

;-----
;extended mem
        xor ax, ax
        xor dx, dx

        mov AL, 30h ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h ; чтение младшего байта
        mov BL, AL ; расширенной памяти
        mov AL, 31h ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h ; чтение старшего байта
                                ; размера расширенной памяти
        mov bh, al
        mov ax, bx
        mov si, offset extmem+22
        call word_to_dec
        mov dx, offset extmem
        call print

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

```

```

        inc     cx
next_mcb:
;mcb number.....
        mov     si, offset mcb+7
        mov     al, cl
        push    cx
        call    byte_to_dec
        mov     dx, offset mcb
        call    print
;owner.....
        mov     dx, offset owner
        call    print
        xor     ah, ah
        mov     al, es:[0]
        push    ax
        mov     ax, es:[1]

        cmp     ax, 0
        mov     dx, offset free
        je      printOwn
        cmp     ax, 6
        mov     dx, offset xms
        je      printOwn
        cmp     ax, 7
        mov     dx, offset tm
        je      printOwn
        cmp     ax, 8
        mov     dx, offset dos
        je      printOwn
        cmp     ax, 0fffah
        mov     dx, offset busy
        je      printOwn
        cmp     ax, 0fffdh
        mov     dx, offset block
        je      printOwn
        cmp     ax, 0fffeh
        mov     dx, offset own_386
        je      printOwn

        mov     di, offset empty+4
        call    wrd_to_hex
        mov     dx, offset empty
printOwn:
        call    print
;size.....
        mov     ax, es:[3]
        mov     cx, 16
        mul     cx
        mov     si, offset arease+11
        call    word_to_dec
        mov     dx, offset arease
        call    print
;data.....

```

```

        xor     dx, dx
        mov     dx, offset lastbytes
        call print
        mov     cx, 8
        xor     di, di
symbol:
        mov     dl, es:[di+8]
        mov     ah, 02h
        int     21h
        inc     di
        loop    symbol
        mov     dx, offset endl
        call    print
;.....
        mov     ax, es:[3]
        mov     bx, es
        add     bx, ax
        inc     bx
        mov     es, bx
        pop     ax
        pop     cx
        inc     cx
        cmp     al, 5Ah ; проверка на не последний ли это сегмент
        je      exit
        cmp     al, 4Dh
        jne     exit
        jmp     next_mcb

exit:    ; Выход в DOS
        xor     AL, AL
        mov     AH, 4Ch
        int     21H
TESTPC ENDS
END START

```

## ПРИЛОЖЕНИЕ Б.

### Исходный код программы lab3\_2.asm.

```
TESTPC SEGMENT
        ASSUME     CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG      100H
START:   JMP      BEGIN
; Данные
AvailableMem db 'Available memory:  b',0DH,0AH,'$'
ExtMem      db 'Extended memory:  kb',0DH,0AH,'$'
MCB         db 'MCB:  $'
Owner       db 'Owner: $'
AreaSize    db 'Size:  $'
LastBytes   db 0DH,0AH,'last 8 bytes: $'
FREE        db ' free      $'
XMS         db ' OS XMS UMB    $'
TM          db ' driver memory $'
DOS         db ' MS DOS      $'
Busy        db ' busy by 386MAX UMB $'
Block       db ' blocked by 386MAX $'
OWN_386     db ' 386MAX UMB    $'
Empty       db '              $'
EndL        db 0Dh, 0Ah, '$'
; Процедуры
;-----
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 шест. 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
;-----
PRINT proc    near
        mov    ah, 09h
        int 21h
        ret
PRINT endp
;-----
WORD_TO_DEC PROC near
        push cx
        push dx
        mov cx, 10
loop1:
        div cx
        or     dl, 30h

```

```

        mov [si], dl
        dec si
        xor dx, dx
        cmp ax, 0
        jnz loop1

        pop dx
        pop cx
        ret
WORD_TO_DEC ENDP
;-----
FREE_MEM  PROC
        push ax
        push bx
        push cx
        push dx

        lea ax, endofprogram
        mov bx, 10h
        xor dx, dx
        div bx
        inc ax
        mov bx, ax
        mov al, 0
        mov ah, 4Ah
        int 21h

        pop dx
        pop cx
        pop bx
        pop ax
        ret
FREE_MEM  ENDP
;-----
; Код
BEGIN:
;available mem
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h

        mov ax, bx
        mov cx, 16
        mul cx
        mov si, offset AvailableMem + 23
        call word_to_dec
        mov dx, offset AvailableMem
        call print
;-----
        call free_mem
;-----
;extended mem

```

```

xor ax, ax
xor dx, dx

mov AL, 30h ; запись адреса ячейки CMOS
out 70h, AL
in AL, 71h ; чтение младшего байта
mov BL, AL ; расширенной памяти
mov AL, 31h ; запись адреса ячейки CMOS
out 70h, AL
in AL, 71h ; чтение старшего байта
                    ; размера расширенной памяти

mov bh, al
mov ax, bx
mov si, offset extmem+22
call word_to_dec
mov dx, offset extmem
call print

;-----
;MCB

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

next_mcb:
;mcb number.....
mov si, offset mcb+7
mov al, cl
push cx
call byte_to_dec
mov dx, offset mcb
call print

;owner.....
mov dx, offset owner
call print
xor ah, ah
mov al, es:[0]
push ax
mov ax, es:[1]

cmp ax, 0
mov dx, offset free
je printOwn
cmp ax, 6
mov dx, offset xms
je printOwn
cmp ax, 7
mov dx, offset tm
je printOwn
cmp ax, 8
mov dx, offset dos

```

```

        je      printOwn
        cmp     ax, 0fffah
        mov     dx, offset busy
        je      printOwn
        cmp     ax, 0fffdh
        mov     dx, offset block
        je      printOwn
        cmp     ax, 0fffeh
        mov     dx, offset own_386
        je      printOwn

        mov     di, offset empty+4
        call wrd_to_hex
        mov     dx, offset empty
printOwn:
        call print
;size.....
        mov     ax, es:[3]
        mov     cx, 16
        mul     cx
        mov     si, offset areabase+11
        call word_to_dec
        mov     dx, offset areabase
        call print
;data.....
        xor     dx, dx
        mov     dx, offset lastbytes
        call print
        mov     cx, 8
        xor     di, di
symbol:
        mov     dl, es:[di+8]
        mov     ah, 02h
        int     21h
        inc     di
        loop    symbol
        mov     dx, offset endl
        call print
;.....
        mov     ax, es:[3]
        mov     bx, es
        add     bx, ax
        inc     bx
        mov     es, bx
        pop     ax
        pop     cx
        inc     cx
        cmp     al, 5Ah ; проверка на не последний ли это сегмент
        je      exit
        cmp     al, 4Dh
        jne     exit
        jmp     next_mcb

```



```
exit:    ; Выход в DOS
        xor AL,AL
        mov AH,4Ch
        int 21H
```

```
endOfProgram:
```

```
TESTPC ENDS
        END START
```

## ПРИЛОЖЕНИЕ В.

### Исходный код программы lab3\_3.asm.

```
TESTPC SEGMENT
        ASSUME     CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG      100H
START:   JMP      BEGIN
; Данные
AvailableMem db 'Available memory:  b',0DH,0AH,'$'
ExtMem      db 'Extended memory:  kb',0DH,0AH,'$'
MCB         db 'MCB:  $'
Owner       db 'Owner: $'
AreaSize    db 'Size:  $'
LastBytes   db 0DH,0AH,'last 8 bytes: $'
FREE        db ' free      $'
XMS         db ' OS XMS UMB  $'
TM          db ' driver memory  $'
DOS         db ' MS DOS      $'
Busy        db ' busy by 386MAX UMB $'
Block       db ' blocked by 386MAX $'
OWN_386     db ' 386MAX UMB  $'
Empty       db '          $'
ErrStr      db 'Error',0Dh,0Ah,'$'
noErrStr    db 'No error',0Dh,0Ah,'$'
EndL        db 0Dh,0Ah,'$'

; Процедуры
;-----
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 шест. 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
;-----
PRINT proc    near
    mov     ah, 09h
    int 21h
    ret
PRINT endp
;-----
WORD_TO_DEC PROC near
    push cx
    push dx
    mov cx, 10

```

```

loop1:
    div cx
    or     dl, 30h
    mov [si], dl
    dec si
    xor dx, dx
    cmp ax, 0
    jnz loop1

    pop dx
    pop cx
    ret
WORD_TO_DEC ENDP
;-----
FREE_MEM  PROC
    push ax
    push bx
    push cx
    push dx

    lea ax, endofprogram
    mov bx, 10h
    xor dx, dx
    div bx
    inc ax
    mov bx, ax
    mov al, 0
    mov ah, 4Ah
    int 21h

    pop dx
    pop cx
    pop bx
    pop ax
    ret
FREE_MEM  ENDP
;-----
; Код
BEGIN:
;available mem
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, bx
    mov cx, 16
    mul cx
    mov  si, offset AvailableMem + 23
    call word_to_dec
    mov dx, offset AvailableMem
    call print
;-----
    call free_mem

```

```

;-----
        push ax
        push bx
        push dx
        mov  bx, 1000h
        mov  ah, 48h
        int  21h

        jc err ;проверка флага CF
        jmp noerr

err:
        mov dx, offset errstr
        call print
        JMP end1

noerr:
        mov dx, offset noerrstr
        call print

end1:

        pop dx
        pop bx
        pop ax

;-----
;extended mem
        xor ax, ax
        xor dx, dx

        mov AL, 30h ; запись адреса ячейки CMOS
        out 70h, AL
        in  AL, 71h ; чтение младшего байта
        mov BL, AL ; расширенной памяти
        mov AL, 31h ; запись адреса ячейки CMOS
        out 70h, AL
        in  AL, 71h ; чтение старшего байта
                                ; размера расширенной памяти
        mov bh, al
        mov ax, bx
        mov si, offset extmem+22
        call word_to_dec
        mov dx, offset extmem
        call print

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

next_mcb:
;mcb number.....

```

```

        mov     si, offset mcb+7
        mov     al, cl
        push    cx
        call    byte_to_dec
        mov     dx, offset mcb
        call    print
;owner.....
        mov     dx, offset owner
        call    print
        xor     ah, ah
        mov     al, es:[0]
        push    ax
        mov     ax, es:[1]

        cmp     ax, 0
        mov     dx, offset free
        je      printOwn
        cmp     ax, 6
        mov     dx, offset xms
        je      printOwn
        cmp     ax, 7
        mov     dx, offset tm
        je      printOwn
        cmp     ax, 8
        mov     dx, offset dos
        je      printOwn
        cmp     ax, 0fffah
        mov     dx, offset busy
        je      printOwn
        cmp     ax, 0fffdh
        mov     dx, offset block
        je      printOwn
        cmp     ax, 0fffeh
        mov     dx, offset own_386
        je      printOwn

        mov     di, offset empty+4
        call    wrd_to_hex
        mov     dx, offset empty
printOwn:
        call    print
;size.....
        mov     ax, es:[3]
        mov     cx, 16
        mul     cx
        mov     si, offset areabase+11
        call    word_to_dec
        mov     dx, offset areabase
        call    print
;data.....
        xor     dx, dx
        mov     dx, offset lastbytes
        call    print

```

```

        mov cx, 8
        xor     di, di
symbol:
        mov     dl, es:[di+8]
        mov     ah, 02h
        int 21h
        inc di
        loop symbol
        mov dx, offset endl
        call print
;.....
        mov ax,es:[3]
        mov bx,es
        add bx,ax
        inc bx
        mov es,bx
        pop ax
        pop cx
        inc cx
        cmp al,5Ah ; проверка на не последний ли это сегмент
        je      exit
        cmp al,4Dh
        jne exit
        jmp next_mcb

exit:    ; Выход в DOS
        xor AL,AL
        mov AH,4Ch
        int 21H

endOfProgram:

TESTPC ENDS
        END START

```

## ПРИЛОЖЕНИЕ В.

### Исходный код программы lab3\_3.asm.

```
TESTPC SEGMENT
        ASSUME     CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG      100H
START:   JMP      BEGIN
; Данные
AvailableMem db 'Available memory:  b',0DH,0AH,'$'
ExtMem      db 'Extended memory:  kb',0DH,0AH,'$'
MCB         db 'MCB:  $'
Owner       db 'Owner: $'
AreaSize    db 'Size:  $'
lastBytes   db 0DH,0AH,'last 8 bytes: $'
FREE        db ' free      $'
XMS         db ' OS XMS UMB    $'
TM          db ' driver memory $'
DOS         db ' MS DOS      $'
Busy        db ' busy by 386MAX UMB $'
Block       db ' blocked by 386MAX $'
OWN_386     db ' 386MAX UMB    $'
Empty       db '          $'
ErrStr      db 'Error',0Dh,0Ah,'$'
noErrStr    db 'No error',0Dh,0Ah,'$'
EndL        db 0Dh,0Ah,'$'

; Процедуры
;-----
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 шест. 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
;-----
PRINT proc    near
    mov     ah, 09h
    int 21h
    ret
PRINT endp
;-----
WORD_TO_DEC PROC near
    push cx
    push dx
    mov cx, 10

```

```

loop1:
    div cx
    or     dl, 30h
    mov [si], dl
    dec si
    xor dx, dx
    cmp ax, 0
    jnz loop1

    pop dx
    pop cx
    ret
WORD_TO_DEC ENDP
;-----
FREE_MEM  PROC
    push ax
    push bx
    push cx
    push dx

    lea ax, endofprogram
    mov bx, 10h
    xor dx, dx
    div bx
    inc ax
    mov bx, ax
    mov al, 0
    mov ah, 4Ah
    int 21h

    pop dx
    pop cx
    pop bx
    pop ax
    ret
FREE_MEM  ENDP
;-----
; Код
BEGIN:
;available mem
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, bx
    mov cx, 16
    mul cx
    mov  si, offset AvailableMem + 23
    call word_to_dec
    mov dx, offset AvailableMem
    call print

;-----

```

```

        push ax
        push bx
        push dx
        mov     bx, 1000h
        mov     ah, 48h
        int     21h

        jc err ;проверка флага CF
        jmp noerr

err:
        mov dx, offset errstr
        call print
        JMP end1

noerr:
        mov dx, offset noerrstr
        call print

end1:

        pop dx
        pop bx
        pop ax

;-----
        call free_mem
;-----
;extended mem
        xor ax, ax
        xor dx, dx

        mov AL, 30h ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h ; чтение младшего байта
        mov BL, AL ; расширенной памяти
        mov AL, 31h ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h ; чтение старшего байта
                                ; размера расширенной памяти

        mov bh, al
        mov ax, bx
        mov si, offset extmem+22
        call word_to_dec
        mov dx, offset extmem
        call print

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

next_mcb:

```

```

;mcb number.....
    mov     si, offset mcb+7
    mov     al, cl
    push    cx
    call    byte_to_dec
    mov     dx, offset mcb
    call    print

;owner.....
    mov     dx, offset owner
    call    print
    xor     ah, ah
    mov     al, es:[0]
    push    ax
    mov     ax, es:[1]

    cmp     ax, 0
    mov     dx, offset free
    je      printOwn
    cmp     ax, 6
    mov     dx, offset xms
    je      printOwn
    cmp     ax, 7
    mov     dx, offset tm
    je      printOwn
    cmp     ax, 8
    mov     dx, offset dos
    je      printOwn
    cmp     ax, 0fffah
    mov     dx, offset busy
    je      printOwn
    cmp     ax, 0fffdh
    mov     dx, offset block
    je      printOwn
    cmp     ax, 0fffeh
    mov     dx, offset own_386
    je      printOwn

    mov     di, offset empty+4
    call    wrd_to_hex
    mov     dx, offset empty

printOwn:
    call    print

;size.....
    mov     ax, es:[3]
    mov     cx, 16
    mul     cx
    mov     si, offset areabase+11
    call    word_to_dec
    mov     dx, offset areabase
    call    print

;data.....
    xor     dx, dx
    mov     dx, offset lastbytes

```

```

        call print
        mov cx, 8
        xor     di, di
symbol:
        mov     dl, es:[di+8]
        mov     ah, 02h
        int 21h
        inc di
        loop symbol
        mov dx, offset endl
        call print
;.....
        mov ax,es:[3]
        mov bx,es
        add bx,ax
        inc bx
        mov es,bx
        pop ax
        pop cx
        inc cx
        cmp al,5Ah ; проверка на не последний ли это сегмент
        je      exit
        cmp al,4Dh
        jne exit
        jmp next_mcb

exit:    ; Выход в DOS
        xor AL,AL
        mov AH,4Ch
        int 21H

endOfProgram:

TESTPC ENDS
        END START

```