

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. 8382

Рочева А.К.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

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

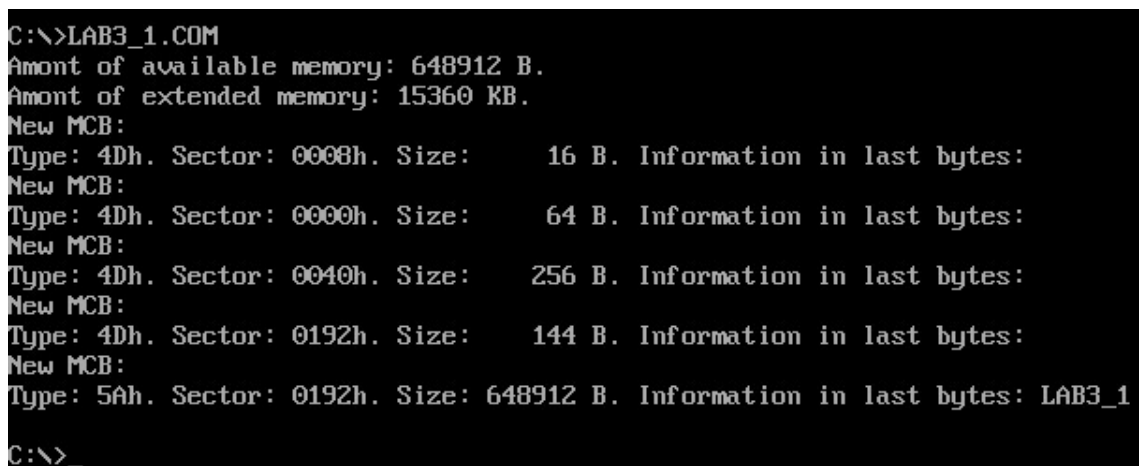
Ход выполнения.

Количество доступной памяти было найдено с помощью функции 4Ah прерывания 21h (т.к. был занесен заведомо больший размер памяти, чем может предоставить ОС, то в регистр BX возвратился размер доступной памяти в параграфах).

Размер расширенной памяти был определен с помощью обращения к ячейкам 30h, 31h CMOS.

Адрес первого блока управления памятью (MCB) был получен с помощью функции 52h прерывания 21h. В результате выполнения этой функции ES:BX указывает на список списков, а слово по адресу ES:[BX-2] и есть адрес самого первого блока. Тип MCB находится по смещению 00h, владелец участка памяти по смещению 01h, размер участка в параграфах по смещению 03h и по смещению 08h находятся последние 8 байт MCB, в которых могут находиться системные данные. Адрес следующего блока памяти вычисляется с помощью адреса текущего блока и его размера.

Результат работы программы показан на рис. 1.



```
C:\>LAB3_1.COM
Amount of available memory: 648912 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size:      16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size:      64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size:     256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     144 B. Information in last bytes:
New MCB:
Type: 5Ah. Sector: 0192h. Size: 648912 B. Information in last bytes: LAB3_1
C:\>_
```

рис.1 — результат работы lab3_1.com

Первый блок принадлежит MS DOS (т. к. в поле сектор содержит 0008h), второй блок свободный (содержит 0000h). Владельцы третьего, четвертого и пятого блоков имеют сегментные адреса PSP 0040h, 0192h, 0192h соответственно.

Затем программа была изменена таким образом, чтобы она освобождала память, которую она не занимает (так же с помощью функции 4Ah прерывания 21h). Результат работы программы представлен на рис. 2.



```
C:\>LAB3_2.COM
Amount of available memory: 648912 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size: 16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size: 64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size: 256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size: 144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size: 11824 B. Information in last bytes: LAB3_2
New MCB:
Type: 5Ah. Sector: 0000h. Size: 637072 B. Information in last bytes: 2â-8eF.8
C:\>
```

рис. 2 — результат работы lab3_2.com

Как видно из скриншота, был создан новый пустой блок (шестой).

Далее программа была изменена так, чтобы после освобождения памяти она запрашивала 64Кб памяти функцией 48h прерывания 21h. Результат работы программы представлен на рис. 3. Как видно, был создан еще один блок размером 64Кб.

```

C:\>LAB3_3.COM
Amount of available memory: 648912 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size:      16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size:      64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size:     256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:   11936 B. Information in last bytes: LAB3_3
New MCB:
Type: 4Dh. Sector: 0192h. Size:  65536 B. Information in last bytes: LAB3_3
New MCB:
Type: 5Ah. Sector: 0000h. Size: 571408 B. Information in last bytes:
C:\>_

```

рис. 3 — результат работы lab3_3.com

В конце был изменен начальный вариант программы запросом 64Кб памяти функцией 48h прерывания 21h до освобождения памяти. Результат работы программы представлен на рис. 4.

```

C:\>LAB3_4.COM
Amount of available memory: 648912 B.
Memory cannot be allocated.Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size:      16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size:      64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size:     256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:   12576 B. Information in last bytes: LAB3_4
New MCB:
Type: 5Ah. Sector: 0000h. Size: 636320 B. Information in last bytes: 7h0P7.4P
C:\>

```

рис. 4 — результат работы lab3_4.com

Программе не удалось запросить 64Кб до освобождения (Во второй строке было выведено сообщение об этом).

Ответы на вопросы:

1. Что означает «доступный объем памяти»?

- это объем памяти, который доступен для выполнения программы.

2. Где МСВ блок Вашей программы в списке?

- в первой и второй программе — это четвертый и пятые блоки (в обеих программах), в третьей — четвертый, пятый и шестой блок, в четвертой программе — четвертый и пятый блоки.

Эти блоки имеют одного владельца с сегментным адресом PSP 0192h и в последних 8 байтах пятых блоков (в случае с третьей программой — еще и в шестом блоке) содержится имя файла программы, которой принадлежит блок.

3. Какой размер памяти занимает программа в каждом случае?

Первая программа занимает 649 056 байт.

Вторая программа занимает 11 968 байт.

Третья программа занимает 11 936 байт + дополнительно 65 536 байт, которые мы запросили.

Четвертая программа занимает 12 720 байт.

Выводы

В ходе выполнения работы была исследована работа функций управления памятью ядра операционной системы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД LAB3_1.ASM

```

LAB3  SEGMENT
        ASSUME CS:LAB3, DS:LAB3, ES:NOTHING, SS:NOTHING
        ORG 100H ; резервирование места для PSP
START:  JMP BEGIN

availableMemory db 'Amount of available memory:      B.', 0DH, 0AH, '$'
extendedMemory db 'Amount of extended memory:      KB.', 0DH, 0AH, '$'
endLine db 0DH, 0AH, '$'
mcbline db 'New MCB:', 0DH, 0AH, 'Type: $'
mcbsector db 'h. Sector: $'
mcbsize db 'h. Size:      B$'
lastBytes db '. Information in last bytes: $'

BEGIN:
        ; кол-во доступной памяти
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h
        mov ax, bx
        mov bx, 16
        mul bx
        lea si, availableMemory + 32
        call WRD_TO_DEC
        lea dx, availableMemory
        call WRITE

        ; размер расширенной памяти
        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
        lea si, extendedMemory + 30
        call WRD_TO_DEC
        lea dx, extendedMemory
        call WRITE

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

mcb:
        lea dx, mcbline
        call WRITE

        ; тип
        mov al, es:[00h]
        call WRITE_BYTE

        ; сектор
        lea dx, mcbsector
        call WRITE
        mov ax, es:[01h]
        mov ch, ah
        mov ah, al
        mov al, ch
        call WRITE_BYTE
        mov ch, ah
        mov ah, al

```

```

        mov al, ch
        call WRITE_BYTE

        ; размер
        mov ax, es:[03h]
        mov bx, 10h
        mul bx
        mov si, offset mcbsize
        add si, 14
        call WRD_TO_DEC
        mov dx, offset mcbsize
        call WRITE

        ; информация в последних восьми байтах
        lea dx, lastBytes
        call WRITE
        mov bx, 0

last:
        mov dl, es:[bx+08h]
        mov ah, 02h
        int 21h
        inc bx
        cmp bx, 8
        jl last

        lea dx, endLine
        call WRITE
        ; если последний тип
        mov al, es:[00h]
        cmp al, 5Ah
        je endmcb

        xor cx, cx
        mov cx, es:[03h]
        mov bx, es
        add bx, cx
        inc bx
        mov es, bx
        jmp mcb

endmcb:
        xor al, al
        mov ah, 4Ch
        int 21h

WRITE_BYTE PROC near
    push ax
    push dx
    push cx
    call BYTE_TO_HEX
    xor cx, cx
    mov ch, ah
    mov dl, al
    mov ah, 02h
    int 21h
    mov dl, ch
    mov ah, 02h
    int 21h
    pop cx
    pop dx
    pop ax
    ret
WRITE_BYTE ENDP

WRITE PROC near
    mov ah, 09
    int 21h
WRITE 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 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_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,10
    jae loop_b
    cmp al,00h
    je endl
    or al,30h
    mov [si],al
endl: pop dx
    pop cx
    ret
WRD_TO_DEC 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

LAB3 ENDS
        END START

```


ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД LAB3_2.ASM

```
LAB3          SEGMENT
               ASSUME CS:LAB3, DS:LAB3, ES:NOTHING, SS:NOTHING
               ORG 100H ; резервирование места для PSP
START:        JMP BEGIN

availableMemory db 'Amount of available memory:      B.', 0DH,0AH, '$'
extendedMemory db 'Amount of extended memory:      KB.', 0DH,0AH, '$'
endLine db 0DH,0AH, '$'
mcbline db 'New MCB:', 0DH,0AH, 'Type: $'
mcbsector db 'h. Sector: $'
mcbsize db 'h. Size:      B$'
lastBytes db '. Information in last bytes: $'

BEGIN:

               ; кол-во доступной памяти
               mov ah, 4ah
               mov bx, 0ffffh
               int 21h
               mov ax, bx
               mov bx, 16
               mul bx
               lea si, availableMemory + 32
               call WRD_TO_DEC
               lea dx, availableMemory
               call WRITE

               ; освобождение
               mov ah, 4ah
               mov bx, offset LAB_END
               int 21h

               ; размер расширенной памяти
               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
               lea si, extendedMemory + 30
```

```

        call WRD_TO_DEC
        lea  dx, extendedMemory
        call WRITE

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

mcb:

        lea  dx, mcbline
        call WRITE

; тип
mov al, es:[00h]
call WRITE_BYTE

; сектор
lea dx, mcbsector
call WRITE
mov ax, es:[01h]
mov ch, ah
mov ah, al
mov al, ch
call WRITE_BYTE
mov ch, ah
mov ah, al
mov al, ch
call WRITE_BYTE

; размер
mov ax, es:[03h]
mov bx, 10h
mul bx
mov si, offset mcbsize
add si, 14
call WRD_TO_DEC
mov dx, offset mcbsize
call WRITE

; информация в последних восьми байтах
lea dx, lastBytes
call WRITE
xor bx, bx

last:

mov dl, es:[bx+08h]
mov ah, 02h

```

```

        int 21h
        inc bx
        cmp bx, 8
        jl last

        lea dx, endLine
        call WRITE
        ; если последний тип
        mov al, es:[00h]
        cmp al, 5Ah
        je endmcb

        xor cx, cx
        mov cx, es:[03h]
        mov bx, es
        add bx, cx
        inc bx
        mov es, bx
        jmp mcb

endmcb:

        xor al, al
        mov ah, 4Ch
        int 21h

WRITE_BYTE PROC near
        push ax
        push dx
        push cx
        call BYTE_TO_HEX
        xor cx, cx
        mov ch, ah
        mov dl, al
        mov ah, 02h
        int 21h
        mov dl, ch
        mov ah, 02h
        int 21h
        pop cx
        pop dx
        pop ax
        ret
WRITE_BYTE ENDP

WRITE PROC near

```

```

                                mov ah, 09
                                int 21h

WRITE 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 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_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,10
                                jae loop_b
                                cmp al,00h
                                je endl
                                or al,30h
                                mov [si],al
endl:
                                pop dx
                                pop cx
                                ret
WRD_TO_DEC 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

LAB_END:
LAB3          ENDS
                END START

```

ПРИЛОЖЕНИЕ С

ИСХОДНЫЙ КОД LAB3_3.ASM

```
LAB3          SEGMENT
               ASSUME CS:LAB3, DS:LAB3, ES:NOTHING, SS:NOTHING
               ORG 100H ; резервирование места для PSP
START:        JMP BEGIN

availableMemory db 'Amount of available memory:      B.', 0DH,0AH, '$'
extendedMemory db 'Amount of extended memory:      KB.', 0DH,0AH, '$'
endLine db 0DH,0AH, '$'
mcbline db 'New MCB:', 0DH,0AH, 'Type: $'
mcbsector db 'h. Sector: $'
mcbsize db 'h. Size:      B$'
lastBytes db '. Information in last bytes: $'

BEGIN:

               ; кол-во доступной памяти
               mov ah, 4ah
               mov bx, 0ffffh
               int 21h
               mov ax, bx
               mov bx, 16
               mul bx
               lea si, availableMemory + 32
               call WRD_TO_DEC
               lea dx, availableMemory
               call WRITE

               ; освобождение
               mov bx, offset LAB_END
               mov ah, 4ah
               int 21h

               ; + 64 кб
               mov bx, 1000h
               mov ah, 48h
               int 21h

               ; размер расширенной памяти
               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
lea si, extendedMemory + 30
call WRD_TO_DEC
lea dx, extendedMemory
call WRITE

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

mcb:

lea dx, mcbline
call WRITE

; тип
mov al, es:[00h]
call WRITE_BYTE

; сектор
lea dx, mcbsector
call WRITE
mov ax, es:[01h]
mov ch, ah
mov ah, al
mov al, ch
call WRITE_BYTE
mov ch, ah
mov ah, al
mov al, ch
call WRITE_BYTE

; размер
mov ax, es:[03h]
mov bx, 10h
mul bx
mov si, offset mcbsize
add si, 14
call WRD_TO_DEC
mov dx, offset mcbsize
call WRITE

; информация в последних восьми байтах
lea dx, lastBytes

```

```

        call WRITE
        xor bx, bx

last:

        mov dl, es:[bx+08h]
        mov ah, 02h
        int 21h
        inc bx
        cmp bx, 8
        jl last

        lea dx, endLine
        call WRITE
        ; если последний тип
        mov al, es:[00h]
        cmp al, 5Ah
        je endmcb

        xor cx, cx
        mov cx, es:[03h]
        mov bx, es
        add bx, cx
        inc bx
        mov es, bx
        jmp mcb

endmcb:

        xor al, al
        mov ah, 4Ch
        int 21h

WRITE_BYTE PROC near
        push ax
        push dx
        push cx
        call BYTE_TO_HEX
        xor cx, cx
        mov ch, ah
        mov dl, al
        mov ah, 02h
        int 21h
        mov dl, ch
        mov ah, 02h
        int 21h
        pop cx
        pop dx

```



```

                pop ax
                ret
WRITE_BYTE     ENDP

WRITE PROC near
                mov ah, 09
                int 21h
WRITE 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 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_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,10
                jae     loop_b
                cmp     al,00h
                je      endl
                or      al,30h
                mov     [si],al
endl:           pop     dx

```

```

                pop    cx
                ret

WRD_TO_DEC 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

LAB_END:
LAB3          ENDS
                END START

```

ПРИЛОЖЕНИЕ D

ИСХОДНЫЙ КОД LAB3_4.ASM

```
LAB3          SEGMENT
               ASSUME CS:LAB3, DS:LAB3, ES:NOTHING, SS:NOTHING
               ORG 100H ; резервирование места для PSP
START:        JMP BEGIN

availableMemory db 'Amount of available memory:      B.', 0DH,0AH, '$'
extendedMemory db 'Amount of extended memory:      KB.', 0DH,0AH, '$'
endLine db 0DH,0AH, '$'
mcbline db 'New MCB:', 0DH,0AH, 'Type: $'
mcbsector db 'h. Sector: $'
mcbsize db 'h. Size:      B$'
lastBytes db '. Information in last bytes: $'
memError db 'Memory cannot be allocated.$'

BEGIN:

               ; кол-во доступной памяти
               mov ah, 4ah
               mov bx, 0ffffh
               int 21h
               mov ax, bx
               mov bx, 16
               mul bx
               lea si, availableMemory + 32
               call WRD_TO_DEC
               lea dx, availableMemory
               call WRITE

               ; + 64 кб
               mov ah, 48h
               mov bx, 1000h
               int 21h
               jc  allocError
               jmp continue

allocError:
               lea dx, memError
               call WRITE

continue:
               ; освобождение
               mov ah, 4ah
               mov bx, offset LAB_END
               int 21h

               ; размер расширенной памяти
```

```

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
lea si, extendedMemory + 30
call WRD_TO_DEC
lea dx, extendedMemory
call WRITE

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

mcb:

lea dx, mcbline
call WRITE

; тип
mov al, es:[00h]
call WRITE_BYTE

; сектор
lea dx, mcbsector
call WRITE
mov ax, es:[01h]
mov ch, ah
mov ah, al
mov al, ch
call WRITE_BYTE
mov ch, ah
mov ah, al
mov al, ch
call WRITE_BYTE

; размер
mov ax, es:[03h]
mov bx, 10h
mul bx
mov si, offset mcbsize

```

```

        add si, 14
        call WRD_TO_DEC
        mov dx, offset mcbsize
        call WRITE

        ; информация в последних восьми байтах
        lea dx, lastBytes
        call WRITE
        xor bx, bx

last:

        mov dl, es:[bx+08h]
        mov ah, 02h
        int 21h
        inc bx
        cmp bx, 8
        jl last

        lea dx, endLine
        call WRITE
        ; если последний тип
        mov al, es:[00h]
        cmp al, 5Ah
        je endmcb

        xor cx, cx
        mov cx, es:[03h]
        mov bx, es
        add bx, cx
        inc bx
        mov es, bx
        jmp mcb

endmcb:

        xor al, al
        mov ah, 4Ch
        int 21h

WRITE_BYTE PROC near
        push ax
        push dx
        push cx
        call BYTE_TO_HEX
        xor cx, cx
        mov ch, ah
        mov dl, al
        mov ah, 02h

```

```

        int 21h
        mov dl, ch
        mov ah, 02h
        int 21h
        pop cx
        pop dx
        pop ax
        ret
WRITE_BYTE    ENDP

WRITE PROC near
        mov ah, 09
        int 21h
WRITE 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 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_DEC PROC NEAR
        push cx
        push dx
        mov cx, 10
loop_b: div cx
        or dx, dx
        mov [si], dl
        dec si
        xor dx, dx
        cmp ax, 10

```

```

                                jae    loop_b
                                cmp    al,00h
                                je      endl
                                or      al,30h
                                mov     [si],al
endl:                          pop     dx
                                pop     cx
                                ret
WRD_TO_DEC 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

LAB_END:
LAB3          ENDS
              END START

```