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. 7381 _____ Габов Е. С.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург

2019

Цель работы

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

Необходимые сведения для составления программы

Учет занятой и свободной памяти ведётся 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.

MCB имеет следующую структуру:

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

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. Размер расширенной памяти в Кбайтах можно определить обращаясь к ячейкам CMOS следующим образом:

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

Порядок выполнения 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. Объём памяти функциями управления памятью выводится в параграфах. Необходимо преобразовать его в байты и выводить в виде десятичных чисел. Последние восемь байт MCB выводятся как символы, не следует преобразовывать их в шестнадцатеричные числа.
- *Далее необходимо изменить программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah прерывания 21h (пример в разделе «Использование функции 4AH»). Хвост командной строки в символьном виде.*
- *Затем необходимо изменить программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H.*
- *Далее нужно изменить первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти. Оформить отчёт и ответить на контрольные вопросы.*

Ход работы:

1. *Запуск .com файла без выделения и освобождения памяти (все доступные 648912 байт отдаются программе):*

```
W:\>lab3_1.com

Size of available memory: 648912 byte
Size of extended memory: 245760 byte

MCB #01
Addr:  016F
Owner: Area belongs to MS DOS
Size: 16 byte
Name:


MCB #02
Addr:  0171
Owner: Empty area
Size: 64 byte
Name:
MCB #03
Addr:  0176
Owner: 0040
Size: 256 byte
Name:


MCB #04
Addr:  0187
Owner: 0192
Size: 144 byte
Name:


MCB #05
Addr:  0191
Owner: 0192
Size: 648912 byte
Name: LAB3_1
Press any key...
```

2. *Запуск .com файла с освобождением незанятой памяти:*

```
Size of available memory: 648912 byte
Freeing memory...
Suscess!
Size of extended memory: 245760 byte

MCB #01
Addr:  016F
Owner: Area belongs to MS DOS
Size: 16 byte
Name:


MCB #02
Addr:  0171
Owner: Empty area
Size: 64 byte
Name:


MCB #03
Addr:  0176
Owner: 0040
Size: 256 byte
Name:
```

```
MCB #04
Addr:  0187
Owner: 0192
Size: 144 byte
Name:


MCB #05
Addr:  0191
Owner: 0192
Size: 1904 byte
Name: LAB3_2


MCB #06
Addr:  0209
Owner: Empty area
Size: 646992 byte
Name:
Press any key...
```

3. *Запуск .com файла с освобождением незанятой памяти и выделением дополнительных 64 Кбайт памяти:*

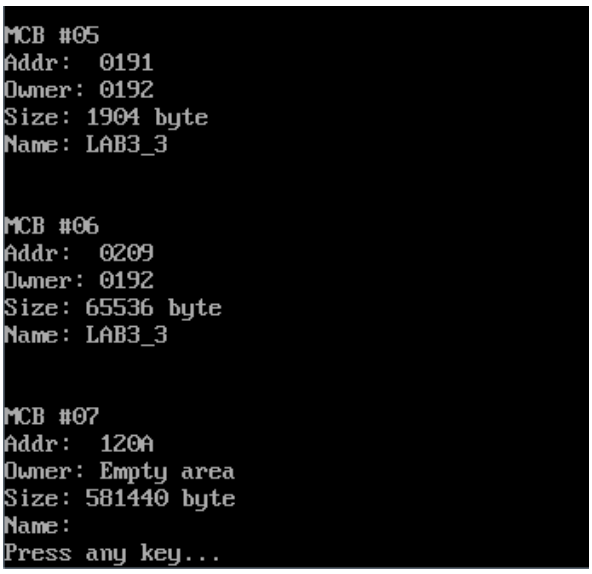
```
Size of available memory: 648912 byte
Freeing memory...
Suscess!
Getting memory...
Suscess!
Size of extended memory: 245760 byte

MCB #01
Addr:  016F
Owner: Area belongs to MS DOS
Size: 16 byte
Name:


MCB #02
Addr:  0171
Owner: Empty area
Size: 64 byte
Name:

MCB #03
Addr:  0176
Owner: 0040
Size: 256 byte
Name:


MCB #04
Addr:  0187
Owner: 0192
Size: 144 byte
Name:
```



4. *Запуск .com файла с выделением дополнительных 64 Кбайт памяти, затем с освобождением незанятой памяти:*

Ответы на контрольные вопросы:

1. Что означает «доступный объем памяти»?

Доступный объем памяти – количество памяти, не занятое другими программами, которое загрузчик выделяет программе при загрузке в основную память, после чего программа сама распоряжается этой памятью с помощью тех же функций, что использовал загрузчик. Поэтому программа может освободить не используемую память, или запросить расширение текущего объема памяти. Если программа запрашивает слишком большой объем памяти (больше, чем размер ее доступной памяти) система откажет ей в этом.

2. Где MCB блок Вашей программы в списке?

Расположен в последнем блоке программы.

3. Какой размер памяти занимает программа в каждом случае?

Для программы выделяется всегда 648912 байт = 634 Кбайт.

Вывод

В результате выполнения данной лабораторной работы были исследованы структуры данных и работа функций управления памятью ядра операционной системы.

Приложение А

1) Lab3_1.asm

```
PCinfosegment
                assume cs:PCinfo, ds:PCinfo, es:nothing, ss:nothing
        org     100h

start:
        jmp             begin

        ;data
        av_mem      db 'Amount of available memory:          b$'
        ex_mem      db 'Size of extended memory:          Kb$'
        mcb   db 'List of memory control blocks:$'
        typeMCB db 'MCB type: 00h$'
        adrPSP      db 'PSP adress: 0000h$'
        size_s      db 'Size:         b$'
   endl         db  13, 10, '$'
   tab          db      9,'$'

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

;Байт в al переводится в два символа 16-ричного числа в ax
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 ;В al старшая цифра, в ah младшая
   pop          cx
   ret
  byte_to_hex endp

;Перевод в 16 сс 16-ти разрядного числа
;ax - число, di - адрес последнего символа
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

;Перевод в 10 cc, si - адрес поля младшей цифры
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

wrd_to_dec proc near
    push        cx
    push        dx
    mov         cx, 10
wloop_bd:
    div  cx
    or   dl, 30h
    mov         [si], dl
    dec         si
         xor    dx, dx
    cmp         ax, 10
    jae  wloop_bd
    cmp         al, 00h
    je          wend_l
    or          al, 30h
    mov         [si], al
wend_l:
    pop         dx
    pop         cx
    ret
   wrd_to_dec endp

;вывод строки
print proc near
    push        ax
    push        dx
```

```
    mov         ah, 09h
    int  21h
    pop         dx
    pop         ax
    ret
   print endp

;вывод символа
print_symb proc near
        push  ax
        push  dx
        mov             ah, 02h
        int             21h
        pop             dx
        pop             ax
        ret
   print_symb endp



begin:

;количество доступной памяти
        mov   ah, 4Ah
        mov   bx, 0ffffh
        int     21h

        xor             dx, dx
        mov   ax, bx
        mov   cx, 10h
        mul     cx

        mov   si, offset av_mem+37
        call    wrd_to_dec

        mov   dx, offset av_mem
        call    print
        mov             dx, offset endl
        call    print

;размер расширенной памяти
        mov             al, 30h
        out             70h, al
        in              al, 71h
        mov             bl, al ;младший байт
        mov             al, 31h
        out             70h, al
        in              al, 71h ;старший байт
        mov             ah, al
        mov             al, bl

        mov             si, offset ex_mem+34
        xor     dx, dx
        call    wrd_to_dec

        mov             dx, offset ex_mem
```

```
        call    print
        mov             dx, offset endl
        call    print

;цепочка блоков управления памятью
   mov              dx, offset mcb
   call         print
        mov             dx, offset endl
        call    print

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

   ;тип MCB
tag1:
        mov     al, es:[0000h]
   call         byte_to_hex
   mov              di, offset typeMCB+10
   mov          [di], ax

   mov              dx, offset typeMCB
   call         print
   mov              dx, offset tab
   call         print

   ;сегментный адрес PSP владельца участка памяти
   mov          ax, es:[0001h]
   mov          di, offset adrPSP+15
   call         wrd_to_hex

   mov              dx, offset adrPSP
   call         print
   mov              dx, offset tab
   call         print

   ;размер участка в параграфах
   mov          ax, es:[0003h]
   mov          cx, 10h
   mul          cx

        mov             si, offset size_s+13
   call         wrd_to_dec
   mov              dx, offset size_s
   call         print
   mov              dx, offset tab
   call         print

   ;последние 8 байт
   push         ds
   push         es
   pop          ds

   mov          dx, 08h
   mov          di, dx
```

```
   mov        cx, 8
tag2:
        cmp           cx,0
        je            tag3
   mov                dl, byte PTR [di]
   call print_symb
   dec        cx
   inc        di
   jmp        tag2
tag3:
        pop   ds
        mov           dx, offset endl
   call       print

   ;проверка, последний блок или нет
   cmp        byte ptr es:[0000h], 5ah
   je         quit

   ;адрес следующего блока
   mov        ax, es
   add        ax, es:[0003h]
   inc  ax
   mov        es, ax
   jmp        tag1

quit:

   xor ax, ax
   mov        ah, 4ch
   int  21h
PCinfoENDS
                     END    START
```

2) Lab3_2.asm

```
        PCinfo        segment
                      assume cs:PCinfo, ds:PCinfo, es:nothing, ss:nothing
              org     100h

        start:
              jmp           begin

              ;data
              av_mem      db 'Amount of available memory:           b$'
              ex_mem      db 'Size of extended memory:          Kb$'
              mcb   db 'List of memory control blocks:$'
              typeMCB db 'MCB type: 00h$'
              adrPSP        db 'PSP adress: 0000h$'
              size_s        db 'Size:         b$'
           endl       db  13, 10, '$'
           tab                db     9,'$'

        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

;Байт в al переводится в два символа 16-ричного числа в ax
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 ;В al старшая цифра, в ah младшая
    pop         cx
    ret
   byte_to_hex endp

;Перевод в 16 сс 16-ти разрядного числа
;ax - число, di - адрес последнего символа
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

;Перевод в 10 сс, si - адрес поля младшей цифры
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

wrd_to_dec proc near
    push    cx
    push    dx
    mov     cx, 10
wloop_bd:
    div     cx
    or      dl, 30h
    mov     [si], dl
    dec     si
        xor dx, dx
    cmp     ax, 10
    jae     wloop_bd
    cmp     al, 00h
    je      wend_l
    or      al, 30h
    mov     [si], al
wend_l:
    pop     dx
    pop     cx
    ret
  wrd_to_dec endp

;вывод строки
print proc near
    push    ax
    push    dx
    mov     ah, 09h
    int 21h
    pop     dx
    pop     ax
    ret
  print endp

;вывод символа
print_symb proc near
        push  ax
        push  dx
        mov         ah, 02h
        int         21h
```

```
        pop         dx
        pop         ax
        ret
   print_symb endp



begin:

;количество доступной памяти
        mov  ah, 4Ah
        mov  bx, 0ffffh
        int     21h

        xor         dx, dx
        mov  ax, bx
        mov  cx, 10h
        mul  cx

        mov  si, offset av_mem+37
        call    wrd_to_dec

        mov  dx, offset av_mem
        call    print
        mov         dx, offset endl
        call    print

;освобождение памяти
    mov     ax,offset SegEnd
    mov     bx, 10h
    xor     dx, dx
    div     bx
    inc     ax
    mov     bx, ax
    mov     al, 0
    mov     ah, 4Ah
    int 21h

;размер расширенной памяти
        mov         al, 30h
        out         70h, al
        in          al, 71h
        mov         bl, al ;младший байт
        mov         al, 31h
        out         70h, al
        in          al, 71h ;старший байт
        mov         ah, al
        mov         al, bl

        mov         si, offset ex_mem+34
        xor    dx, dx
```

```
        call    wrd_to_dec

        mov             dx, offset ex_mem
        call    print
        mov             dx, offset endl
        call    print

;цепочка блоков управления памятью
    mov             dx, offset mcb
    call        print
        mov             dx, offset endl
        call    print

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

    ;тип MCB
tag1:
        mov  al, es:[0000h]
    call        byte_to_hex
    mov             di, offset typeMCB+10
    mov       [di], ax

    mov             dx, offset typeMCB
    call        print
    mov             dx, offset tab
    call        print

    ;сегментный адрес PSP владельца участка памяти
    mov       ax, es:[0001h]
    mov       di, offset adrPSP+15
    call        wrd_to_hex

    mov             dx, offset adrPSP
    call        print
    mov             dx, offset tab
    call        print

    ;размер участка в параграфах
    mov       ax, es:[0003h]
    mov       cx, 10h
    mul       cx

        mov             si, offset size_s+13
    call        wrd_to_dec
    mov             dx, offset size_s
    call        print
    mov             dx, offset tab
    call        print
```

```
    ;последние 8 байт
    push    ds
    push    es
    pop     ds

    mov     dx, 08h
    mov     di, dx
    mov     cx, 8
tag2:
        cmp         cx,0
        je          tag3
    mov             dl, byte PTR [di]
    call    print_symb
    dec     cx
    inc     di
    jmp             tag2
tag3:
        pop ds
        mov         dx, offset endl
    call    print

    ;проверка, последний блок или нет
    cmp     byte ptr es:[0000h], 5ah
    je      quit

    ;адрес следующего блока
    mov     ax, es
    add     ax, es:[0003h]
    inc     ax
    mov     es, ax
    jmp     tag1

quit:

    xor     ax, ax
    mov     ah, 4ch
    int 21h
SegEnd:
PCinfo      ENDS
            END   START
```

3)
```
Lab3_3
PCinfo      segment
            assume cs:PCinfo, ds:PCinfo, es:nothing, ss:nothing
        org 100h

start:
        jmp         begin

        ;data
```

```
        av_mem     db 'Amount of available memory:          b$'
        ex_mem     db 'Size of extended memory:          Kb$'
        mcb   db 'List of memory control blocks:$'
        typeMCB db 'MCB type: 00h$'
        adrPSP       db 'PSP adress: 0000h$'
        size_s       db 'Size:         b$'
    endl       db  13, 10, '$'
    tab              db     9,'$'
    error      db 'ERROR! Memory can not be 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

;Байт в al переводится в два символа 16-ричного числа в ax
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 ;В al старшая цифра, в ah младшая
    pop        cx
    ret
  byte_to_hex endp

;Перевод в 16 сс 16-ти разрядного числа
;ax - число, di - адрес последнего символа
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
```

```
;Перевод в 10 сс, si - адрес поля младшей цифры
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

wrd_to_dec proc near
    push      cx
    push      dx
    mov       cx, 10
wloop_bd:
    div       cx
    or        dl, 30h
    mov       [si], dl
    dec       si
        xor   dx, dx
    cmp       ax, 10
    jae       wloop_bd
    cmp       al, 00h
    je        wend_l
    or        al, 30h
    mov       [si], al
wend_l:
    pop       dx
    pop       cx
    ret
   wrd_to_dec endp

;вывод строки
print proc near
    push      ax
    push      dx
```

```
    mov     ah, 09h
    int 21h
    pop     dx
    pop     ax
    ret
   print endp

;вывод символа
print_symb proc near
        push  ax
        push  dx
        mov         ah, 02h
        int         21h
        pop         dx
        pop         ax
        ret
   print_symb endp



begin:

;количество доступной памяти
        mov   ah, 4Ah
        mov   bx, 0ffffh
        int     21h

        xor         dx, dx
        mov   ax, bx
        mov   cx, 10h
        mul   cx

        mov   si, offset av_mem+37
        call    wrd_to_dec

        mov   dx, offset av_mem
        call    print
        mov         dx, offset endl
        call    print

;освобождение памяти
    mov     ax,offset SegEnd
    mov     bx, 10h
    xor     dx, dx
    div     bx
    inc     ax
    mov     bx, ax
    mov     al, 0
    mov     ah, 4Ah
    int 21h
```

```
;запрос памяти
        xor             ax, ax
        mov             ah, 48h
        mov             bx, 1000h
        int             21h
        jnc             mem_ok
        mov             dx, offset error
        call    print
        mov             dx,     offset endl
        call    print
mem_ok:

;размер расширенной памяти
        mov             al, 30h
        out             70h, al
        in              al, 71h
        mov             bl, al ;младший байт
        mov             al, 31h
        out             70h, al
        in              al, 71h ;старший байт
        mov             ah, al
        mov             al, bl

        mov             si, offset ex_mem+34
        xor     dx, dx
        call    wrd_to_dec

        mov             dx, offset ex_mem
        call    print
        mov             dx, offset endl
        call    print

;цепочка блоков управления памятью
    mov             dx, offset mcb
    call        print
        mov             dx, offset endl
        call    print

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

    ;тип MCB
tag1:
        mov   al, es:[0000h]
    call        byte_to_hex
    mov             di, offset typeMCB+10
    mov       [di], ax

    mov             dx, offset typeMCB
```

```
    call        print
    mov             dx, offset tab
    call        print

    ;сегментный адрес PSP владельца участка памяти
    mov         ax, es:[0001h]
    mov         di, offset adrPSP+15
    call        wrd_to_hex

    mov             dx, offset adrPSP
    call        print
    mov             dx, offset tab
    call        print

    ;размер участка в параграфах
    mov         ax, es:[0003h]
    mov         cx, 10h
    mul         cx

        mov         si, offset size_s+13
    call        wrd_to_dec
    mov             dx, offset size_s
    call        print
    mov             dx, offset tab
    call        print

    ;последние 8 байт
    push        ds
    push        es
    pop         ds

    mov         dx, 08h
    mov         di, dx
    mov         cx, 8
tag2:
        cmp         cx,0
        je              tag3
    mov             dl, byte PTR [di]
    call        print_symb
    dec         cx
    inc         di
    jmp             tag2
tag3:
        pop     ds
        mov         dx, offset endl
    call        print

    ;проверка, последний блок или нет
    cmp         byte ptr es:[0000h], 5ah
    je          quit
```

```
    ;адрес следующего блока
    mov     ax, es
    add     ax, es:[0003h]
    inc     ax
    mov     es, ax
    jmp     tag1

quit:

    xor     ax, ax
    mov     ah, 4ch
    int 21h
SegEnd:
PCinfo      ENDS
            END   START
```

4) Lab3_4.asm

```
PCinfo      segment
            assume cs:PCinfo, ds:PCinfo, es:nothing, ss:nothing
      org   100h

start:
      jmp         begin

      ;data
      av_mem    db 'Amount of available memory:          b$'
      ex_mem    db 'Size of extended memory:          Kb$'
      mcb   db 'List of memory control blocks:$'
      typeMCB db 'MCB type: 00h$'
      adrPSP      db 'PSP adress: 0000h$'
      size_s      db 'Size:         b$'
   endl     db  13, 10, '$'
   tab              db    9,'$'
   error    db 'ERROR! Memory can not be 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

;Байт в al переводится в два символа 16-ричного числа в ax
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 ;В al старшая цифра, в ah младшая
    pop     cx
    ret
  byte_to_hex endp

;Перевод в 16 сс 16-ти разрядного числа
;ax - число, di - адрес последнего символа
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

;Перевод в 10 сс, si - адрес поля младшей цифры
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
```

```
wrd_to_dec proc near
   push     cx
   push     dx
   mov      cx, 10
wloop_bd:
   div      cx
   or       dl, 30h
   mov      [si], dl
   dec      si
       xor  dx, dx
   cmp      ax, 10
   jae      wloop_bd
   cmp      al, 00h
   je       wend_l
   or       al, 30h
   mov      [si], al
wend_l:
   pop      dx
   pop      cx
   ret
  wrd_to_dec endp

;вывод строки
print proc near
   push     ax
   push     dx
   mov      ah, 09h
   int 21h
   pop      dx
   pop      ax
   ret
  print endp

;вывод символа
print_symb proc near
       push  ax
       push  dx
       mov       ah, 02h
       int       21h
       pop       dx
       pop       ax
       ret
  print_symb endp



begin:

;количество доступной памяти
       mov   ah, 4Ah
       mov   bx, 0ffffh
```

```
        int     21h

        xor             dx, dx
        mov     ax, bx
        mov     cx, 10h
        mul     cx

        mov     si, offset av_mem+37
        call    wrd_to_dec

        mov     dx, offset av_mem
        call    print
        mov             dx, offset endl
        call    print


;запрос памяти
        xor             ax, ax
        mov             ah, 48h
        mov             bx, 1000h
        int             21h
        jnc             mem_ok
        mov             dx, offset error
        call    print
        mov             dx,     offset endl
        call    print
mem_ok:

;освобождение памяти
   mov      ax,offset SegEnd
   mov      bx, 10h
   xor      dx, dx
   div      bx
   inc      ax
   mov      bx, ax
   mov      al, 0
   mov      ah, 4Ah
   int 21h

;размер расширенной памяти
        mov             al, 30h
        out             70h, al
        in              al, 71h
        mov             bl, al ;младший байт
        mov             al, 31h
        out             70h, al
        in              al, 71h ;старший байт
        mov             ah, al
        mov             al, bl

        mov             si, offset ex_mem+34
```

```
        xor     dx, dx
        call    wrd_to_dec

        mov             dx, offset ex_mem
        call    print
        mov             dx, offset endl
        call    print

;цепочка блоков управления памятью
    mov             dx, offset mcb
    call        print
        mov             dx, offset endl
        call    print

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

    ;тип MCB
tag1:
        mov   al, es:[0000h]
    call        byte_to_hex
    mov             di, offset typeMCB+10
    mov       [di], ax

    mov             dx, offset typeMCB
    call        print
    mov             dx, offset tab
    call        print

    ;сегментный адрес PSP владельца участка памяти
    mov       ax, es:[0001h]
    mov       di, offset adrPSP+15
    call        wrd_to_hex

    mov             dx, offset adrPSP
    call        print
    mov             dx, offset tab
    call        print

    ;размер участка в параграфах
    mov       ax, es:[0003h]
    mov       cx, 10h
    mul       cx

        mov             si, offset size_s+13
    call        wrd_to_dec
    mov             dx, offset size_s
    call        print
    mov             dx, offset tab
```

```
    call        print

    ;последние 8 байт
    push        ds
    push        es
    pop         ds

    mov         dx, 08h
    mov         di, dx
    mov         cx, 8
tag2:
        cmp             cx,0
        je              tag3
    mov                 dl, byte PTR [di]
    call        print_symb
    dec         cx
    inc         di
    jmp                 tag2
tag3:
        pop     ds
        mov             dx, offset endl
    call        print

    ;проверка, последний блок или нет
    cmp         byte ptr es:[0000h], 5ah
    je          quit

    ;адрес следующего блока
    mov         ax, es
    add         ax, es:[0003h]
    inc         ax
    mov         es, ax
    jmp         tag1

quit:

    xor         ax, ax
    mov         ah, 4ch
    int 21h
SegEnd:
PCinfo      ENDS
            END    START
```