# МИНОБРНАУКИ РОССИИ САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА) Кафедра МОЭВМ

#### ЛАБОРАТОРНАЯ РАБОТА № 3

по дисциплине «Операционные системы»

ТЕМА: Исследование организации управления основной памятью.

Студентка гр. 7381	Алясова А.Н.
Преподаватель	Ефремов М.А.

Санкт-Петербург 2019

#### Цель работы.

Для исследования организации управления памятью. Необходимо ориентироваться на тип основной памяти, реализованный в компьютере и способ организации, принятый в ОС. В лабораторной работе рассматривается нестраничная память и способ управления динамическими разделами. Для реализации управления памятью в этом случае строится список занятых и свободных участков памяти. Функции ядра, обеспечивающие управление основной памятью, просматривают и преобразуют этот список.

В лабораторной работе исследуются структуры данных и работа функций управления памятью ядра операционной системы.

#### Ход работы.

- 1) Написали и отладили программный модуль типа .СОМ, который выбирает и распечатывает следующую информацию:
  - Количество доступной памяти
  - Размер расширенной памяти
  - Выводит цепочку битов управления памятью

```
:\>tasm lab3_1.asm
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International
Assembling file:
                     lab3_1.asm
Error messages:
                    None
Warning messages:
Passes:
Remaining memory:
C:\>tlink /t lab3_1.obj
Turbo Link Version 5.1 Copyright (c) 1992 Borland International
C:\>lab3_1.com
Amount of available memory:
                                   648912 Ь
Size of extended memory:
List of memory control blocks:
                 PSP adress: 0008h
PSP adress: 0000h
1CB type: 4Dh
                                           Size:
1CB type: 4Dh
                                           Size:
                                                         64 b
                                                                      DPMILOAD
1CB type: 4Dh
                 PSP adress: 0040h
                                           Size:
                                                        256 Ъ
1CB type: 4Dh
                 PSP adress: 0192h
                                           Size:
                                                        144 Ь
                                                    648912 Ъ
1CB type: 5Ah
                                                                      LAB3_1
                 PSP adress: 0192h
                                           Size:
```

2) Изменили программу таким образом, что она освобождает память, которую она не занимает. Для этого использовали функцию 4Ah прерывания 21H.

```
C:\>tasm lab3_2.asm
Turbo Assembler  Version 3.1  Copyright (c) 1988, 1992 Borland International
Assembling file:
                   lab3_2.asm
Error messages:
                   None
Warning messages:
                   None
Passes:
Remaining memory:
                   472k
C:/>tlink /t lab3_2.obj
Turbo Link Version 5.1 Copyright (c) 1992 Borland International
C:\>lab3_2.com
Amount of available memory:
                                648912 Ъ
                               15360 Kb
Size of extended memory:
List of memory control blocks:
MCB type: 4Dh
                PSP adress: 0008h
                                                     16 b
                                         Size:
MCB type: 4Dh
                PSP adress: 0000h
                                                     64 b
                                                                  DPMILOAD
                                         Size:
MCB type: 4Dh
                PSP adress: 0040h
                                         Size:
                                                    256 Ъ
                                         Size:
1CB type: 4Dh
                PSP adress: 0192h
                                                    144
                                                        b
MCB type: 4Dh
                PSP adress: 0192h
                                         Size:
                                                    816 Ъ
                                                                  LAB3 2
MCB type: 5Ah
                PSP adress: 0000h
                                         Size:
                                                 648080 Ъ
```

3) Изменили программу еще раз таким способом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48Н прерывания 21Н.

```
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International
Assembling file:
                    lab3_3.asm
Error messages:
                    None
Warning messages:
                    None
Passes:
Remaining memory:
                   471k
C:\>tlink /t lab3_3.obj
Turbo Link Version 5.1 Copyright (c) 1992 Borland International
C: \>lab3_3.com
Amount of available memory:
                                  648912 Ъ
Size of extended memory:
                                15360 Kb
List of memory control blocks:
MCB type: 4Dh
                PSP adress: 0008h
                                           Size:
                                                       16 b
CB type: 4Dh
                 PSP adress: 0000h
                                           Size:
                                                       64 b
                                                                     DPMILOAD
MCB type: 4Dh
                 PSP adress: 0040h
                                           Size:
                                                      256 b
MCB type: 4Dh
                 PSP adress: 0192h
                                           Size:
                                                       144 b
                                                                    LAB3_3
LAB3_3
MCB type: 4Dh
                 PSP adress: 0192h
                                           Size:
                                                      864 b
MCB type: 4Dh
                 PSP
                     adress: 0192h
                                           Size:
                                                    65536 ь
MCB type: 5Ah
                 PSP adress: 0000h
                                                   582480 Ъ
                                           Size:
```

4) Изменили первоначальный вариант программы, запросив 64Кб памяти функцией 48Н прерывания 21Н до освобождения памяти.

```
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International
Assembling file:
                        lab3_4.asm
Error messages:
Warning messages:
                        None
                        None
Passes:
Remaining memory:
                        471k
C:\>tlink /t lab3_4.obj
Turbo Link Version 5.1 Copyright (c) 1992 Borland International
C:\>lab3_4.com
Amount of available memory: 6489
ERROR! Memory can not be allocated!
Size of extended memory: 15360
                                         648912 Ъ
                                      15360 КЪ
List of memory control blocks:
MCB type: 4Dh
MCB type: 4Dh
MCB type: 4Dh
                    PSP adress: 0008h
PSP adress: 0000h
                                                   Size:
                                                                   16 b
                                                                  64 b
                                                                                  DPMILOAD
                                                   Size:
                    PSP adress: 0040h
                                                   Size:
                                                                 256 Ъ
MCB type: 4Dh
                    PSP adress: 0192h
                                                   Size:
                                                                 144 b
MCB type: 4Dh
                                                                                  LAB3_4
                    PSP adress: 0192h
                                                   Size:
                                                                 864 Ъ
MCB type: 5Ah
                    PSP adress: 0000h
                                                   Size:
                                                             648032 Ъ
```

Сведения о функциях и структурах данных управляющей программы:

#### Переменные:

Названиие	Тип	Содержит текст	
av_mem		Amount of available memory: b\$	
ex_mem		Size of extended memory: Kb\$	
mcb	db –	List of memory control blocks:\$	
typeMCB	определяет	MCB type: 00h\$	
adrPSP	байт	PSP adress: 0000h\$	
size_s		Size: b\$	
endl		13,10 '\$'	
tab		9, '\$'	

<sup>&#</sup>x27;\$'-осуществляет переход на другую строку.

#### Функции:

Название	Описание
TETR_TO_HEX	осуществляет перевод половины
	байта в символ.
BYTE_TO_HEX	осуществляет перевод байта,
	помещенного в al, в два символа в

	шестнадцатеричной системе	
	счисления, помещая результат в ах.	
WRD_TO_HEX	осуществляет перевод числового	
	значения, помещенного в регистр	
	АХ, в символьную строку в	
	шестнадцатеричной системе	
	счисления, помещая результат в	
	регистр di.	
BYTE_TO_DEC	осуществляет перевод байта,	
	помещенного в AL, в два символа в	
	десятичной системе счисления,	
	помещая результат в SI.	
WRD_TO_DEC	осуществляет перевод слова,	
	помещенного в АХ, в	
	последовательность символов в	
	десятичной системе счисления,	
	помещая результат в SI. (по аналогии	
	c BYTE_TO_DEC)	
PRINT	осуществляет вывод строки на экран.	
PRINT_SYMB	осуществляет вывод символа на	
	экран (используя функцию DOS 02h,	
	прерывания 21).	

#### Ответы на контрольные вопросы:

#### 1. Что означает "доступный объем памяти"?

Доступный объем памяти – область основной памяти, выделенная программе.

#### 2. Где МСВ блок вашей программы в списке?

В первой версии программы он расположен в конце списка, т.к. программа была последней загружена в память и обладает всем объемом свободной ранее памяти.

Во второй версии, МСВ блок - предпоследний. Последним является блок, освобожденной программой памяти.

В третьей версии, блок так же пятый в списке, но после него располагаются блок, памяти в 64Кб, выделенный по запросу программы и после – блок свободной памяти.

В четвертой ситуация как во второй версии, так как запрос выделения памяти завершился неудачей.

### 3. Какой размер памяти занимает программа в каждом случае?

В первой программе, она занимает всю выделенную ей память: 648912б.

Во второй – только объем, занимаемый самой программой: 800 б.

В третьей – объем, занимаемый самой программой и 64 Кб, выделенные ей по требованию: 65536 б.

В четвертой – только объем, занимаемый самой программой, т. к. выделить 64кБ было невозможно: 864 б.

#### ПРИЛОЖЕНИЕ А

## исходный код

#### lab3\_1.asm

```
PCinfo
          segment
          assume cs:PCinfo, ds:PCinfo, es:nothing, ss:nothing
          100h
     org
start:
     jmp
               begin
     ;data
               db 'Amount of available memory:
                                                         b$'
     av_mem
               db 'Size of extended memory:
                                                      Kb$'
     ex mem
     mcb db 'List of memory control blocks:$'
     typeMCB db 'MCB type: 00h$'
     adrPSP db 'PSP adress: 0000h$'
                                 b$'
     size_s db 'Size:
          db 13, 10, '$'
   endl
               db 9,'$'
   tab
tetr_to_hex proc near
        al, 0Fh
   and
        al, 09
   cmp
   jbe
        next
          al, 07
   add
next:
          al, 30h
   add
   ret
  tetr_to_hex endp
```

```
;Байт в al переводится в два символа 16-ричного числа в ах
byte_to_hex proc near
    push
           \mathsf{C}\mathsf{X}
         ah, al
    mov
    call tetr_to_hex
    xchg al, ah
         cl, 4
    mov
    shr
         al, cl
           tetr_to_hex ;В al старшая цифра, в ah младшая
    call
    pop
           \mathsf{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
           bh, ah
    mov
    call
           byte_to_hex
          [di], ah
    mov
          di
    dec
    mov
         [di], al
    dec
           di
           al, bh
    mov
    call
           byte_to_hex
          [di], ah
    mov
           di
    dec
           [di], al
    mov
    pop
           bx
```

```
wrd_to_hex endp
;Перевод в 10 cc, si - адрес поля младшей цифры
byte_to_dec proc near
    push
           \mathsf{cx}
    push dx
    xor ah, ah
    xor
        dx, dx
          cx, 10
    mov
loop_bd:
    div
           \mathsf{cx}
                 dl, 30h
    or
           [si], dl
    mov
         si
    dec
         dx, dx
    xor
         ax, 10
    cmp
         loop_bd
    jae
           al, 00h
    cmp
                 end_1
    je
                 al, 30h
    or
           [si], al
    mov
end_1:
    pop
           dx
    pop
           \mathsf{cx}
    ret
   byte_to_dec endp
wrd_to_dec proc near
```

ret

push

 $\mathsf{cx}$ 

```
push
           dx
           cx, 10
    mov
wloop_bd:
    div
           \mathsf{cx}
           dl, 30h
    or
           [si], dl
    mov
    dec
           si
           dx, dx
     xor
    cmp
           ax, 10
           wloop_bd
    jae
           al, 00h
    cmp
                 wend_1
    je
                 al, 30h
    or
           [si], al
    mov
wend_1:
    pop
           dx
    pop
           СХ
    ret
   wrd_to_dec endp
;вывод строки
print proc near
    push
           ax
    push
           dx
           ah, 09h
    mov
    int
           21h
           dx
    pop
    pop
           ax
    ret
```

print endp

```
print_symb proc near
     push ax
     push dx
               ah, 02h
     mov
               21h
     int
               dx
     pop
     pop
               ax
     ret
  print_symb endp
begin:
;количество доступной памяти
     mov ah, 4Ah
          bx, 0ffffh
     mov
     int 21h
     xor dx, dx
     mov ax, bx
     mov
          cx, 10h
     mul
          СХ
     mov si, offset av_mem+37
     call wrd_to_dec
     mov dx, offset av_mem
```

;вывод символа

```
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

```
ah, 52h
   mov
   int
          21h
          ax, es:[bx-2]
   mov
   mov
          es, ax
   ;тип МСВ
tag1:
     mov al, es:[0000h]
   call byte to hex
               di, offset typeMCB+10
   mov
          [di], ax
   mov
               dx, offset typeMCB
   mov
   call
          print
   mov
              dx, offset tab
   call
          print
   ;сегментный адрес PSP владельца участка памяти
          ax, es:[0001h]
   mov
   mov di, offset adrPSP+15
   call wrd_to_hex
   mov
               dx, offset adrPSP
   call
          print
              dx, offset tab
   mov
   call
          print
   ;размер участка в параграфах
         ax, es:[0003h]
   mov
         cx, 10h
   mov
```

```
si, offset size_s+13
    mov
   call wrd to dec
        dx, offset size_s
   mov
   call
        print
       dx, offset tab
   mov
   call
        print
   ;последние 8 байт
   push ds
   push es
   pop ds
   mov dx, 08h
   mov di, dx
   mov
       cx, 8
tag2:
       cx,0
   cmp
    je
            tag3
          dl, byte PTR [di]
   mov
   call print_symb
   dec cx
   inc
            di
      tag2
   jmp
tag3:
    pop ds
    mov dx, offset endl
   call
        print
```

mul cx

```
byte ptr es:[0000h], 5ah
    cmp
    jе
                quit
    ;адрес следующего блока
          ax, es
    mov
    add
          ax, es:[0003h]
    inc
          ax
         es, ax
    mov
          tag1
    jmp
quit:
          ax, ax
    xor
    mov
         ah, 4ch
    int
          21h
PCinfo
          ENDS
          END
                 START
                              lab3_2.asm
PCinfo
          segment
          assume cs:PCinfo, ds:PCinfo, es:nothing, ss:nothing
          100h
     org
start:
     jmp
                begin
     ;data
                db 'Amount of available memory:
     av_mem
                                                           b$'
                db 'Size of extended memory:
                                                        Kb$'
     ex_mem
```

;проверка, последний блок или нет

```
db 'List of memory control blocks:$'
     typeMCB db 'MCB type: 00h$'
     adrPSP
                db 'PSP adress: 0000h$'
              db 'Size:
                                    b$'
     size s
           db 13, 10, '$'
    endl
                   9,'$'
    tab
                db
tetr_to_hex proc near
    and
         al, 0Fh
         al, 09
    cmp
          next
    jbe
        al, 07
    add
next:
           al, 30h
    add
    ret
   tetr_to_hex endp
;Байт в al переводится в два символа 16-ричного числа в ах
byte_to_hex proc near
    push
           \mathsf{cx}
          ah, al
    mov
    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
           \mathsf{C}\mathsf{X}
    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
         [di], al
   mov
          di
   dec
         al, bh
   mov
   call
          byte_to_hex
   mov
         [di], ah
          di
   dec
   mov
         [di], al
          bx
   pop
   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
          \mathsf{cx}
                dl, 30h
   or
          [si], dl
   mov
```

```
dec
            si
            dx, dx
    xor
            ax, 10
    cmp
    jae
            loop_bd
            al, 00h
    cmp
                   end_1
    jе
                   al, 30h
    or
            [si], al
    mov
end_1:
    pop
            dx
    pop
            \mathsf{C}\mathsf{X}
    ret
   byte_to_dec endp
wrd_to_dec proc near
    push
            \mathsf{C}\mathsf{X}
    push
            dx
            cx, 10
    mov
wloop_bd:
    div
            сх
            dl, 30h
    or
            [si], dl
    mov
            si
    dec
            dx, dx
      xor
            ax, 10
    cmp
            wloop_bd
    jae
            al, 00h
    cmp
                  wend_1
    je
                  al, 30h
    or
            [si], al
```

mov

```
wend_1:
         dx
   pop
   pop
         CX
   ret
  wrd_to_dec endp
;вывод строки
print proc near
   push ax
   push dx
        ah, 09h
   mov
   int
        21h
        dx
   pop
   pop
        ax
   ret
  print endp
;вывод символа
print_symb proc near
     push ax
     push dx
               ah, 02h
     mov
               21h
     int
     pop
               dx
     pop
               ax
     ret
  print_symb endp
```

#### begin:

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

;количество доступной памяти

mov ah, 4Ah

#### ;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]

```
;тип МСВ
tag1:
     mov al, es:[0000h]
          byte_to_hex
    call
                di, offset typeMCB+10
    mov
          [di], ax
    mov
                dx, offset typeMCB
    mov
          print
    call
               dx, offset tab
    mov
    call
          print
    ;сегментный адрес PSP владельца участка памяти
          ax, es:[0001h]
    mov
    mov
          di, offset adrPSP+15
    call wrd_to_hex
                dx, offset adrPSP
    mov
    call
          print
                dx, offset tab
    mov
    call
          print
    ;размер участка в параграфах
         ax, es:[0003h]
    mov
          cx, 10h
    mov
    mul
          \mathsf{cx}
```

mov es, ax

si, offset size\_s+13

mov

```
call wrd_to_dec
          dx, offset size_s
   mov
   call
         print
          dx, offset tab
   mov
   call
         print
   ;последние 8 байт
   push ds
   push es
   pop ds
   mov dx, 08h
   mov di, dx
         cx, 8
   mov
tag2:
             cx,0
    cmp
    je
              tag3
              dl, byte PTR [di]
   mov
         print_symb
   call
   dec
         \mathsf{cx}
              di
   inc
             tag2
   jmp
tag3:
    pop
         ds
        dx, offset endl
    mov
   call
         print
   ;проверка, последний блок или нет
         byte ptr es:[0000h], 5ah
   cmp
              quit
   je
```

```
ax, es
    mov
          ax, es:[0003h]
    add
    inc
          ax
          es, ax
    mov
    jmp
          tag1
quit:
    xor
          ax, ax
          ah, 4ch
    mov
    int
          21h
SegEnd:
PCinfo
          ENDS
          END
                 START
                              lab3_3.asm
PCinfo
          segment
          assume cs:PCinfo, ds:PCinfo, es:nothing, ss:nothing
          100h
     org
start:
                begin
     jmp
     ;data
                db 'Amount of available memory:
                                                            b$'
     av_mem
                db 'Size of extended memory:
                                                        Kb$'
     ex_mem
     mcb db 'List of memory control blocks:$'
     typeMCB db 'MCB type: 00h$'
```

;адрес следующего блока

```
adrPSP db 'PSP adress: 0000h$'
     size_s db 'Size:
                                  b$'
   endl db 13, 10, '$'
               db 9,'$'
   tab
   error db 'ERROR! Memory can not be allocated!$'
tetr_to_hex proc near
   and
         al, 0Fh
   cmp al, 09
        next
   jbe
        al, 07
   add
next:
        al, 30h
   add
   ret
  tetr_to_hex endp
;Байт в al переводится в два символа 16-ричного числа в ах
byte_to_hex proc near
   push cx
          ah, al
   mov
   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
          \mathsf{C}\mathsf{X}
   ret
   byte_to_hex endp
```

```
;ax - число, di - адрес последнего символа
wrd_to_hex proc near
    push
          bx
          bh, ah
    mov
    call byte_to_hex
         [di], ah
    mov
    dec
        di
         [di], al
    mov
    dec
         di
         al, bh
    mov
    call
          byte_to_hex
    mov
         [di], ah
          di
    dec
        [di], al
    mov
    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
        dx, dx
    xor
         cx, 10
    mov
loop_bd:
    div
          \mathsf{cx}
               dl, 30h
    or
          [si], dl
    mov
    dec
          si
```

```
dx, dx
    xor
            ax, 10
    cmp
            loop_bd
    jae
            al, 00h
    cmp
                  end_1
    je
                  al, 30h
    or
            [si], al
    mov
end_l:
    pop
            dx
    pop
            \mathsf{C}\mathsf{X}
    ret
   byte_to_dec endp
wrd_to_dec proc near
    push
            \mathsf{cx}
    push
            dx
            cx, 10
    mov
wloop_bd:
    div
            сх
            dl, 30h
    or
            [si], dl
    mov
    dec
            si
            dx, dx
      xor
    cmp
            ax, 10
    jae
            wloop_bd
            al, 00h
    cmp
                  wend_1
    je
                  al, 30h
    or
            [si], al
    mov
```

wend\_1:

```
pop
          dx
    pop
          \mathsf{cx}
    ret
  wrd_to_dec endp
;вывод строки
print proc near
    push ax
    push dx
        ah, 09h
    mov
         21h
    int
        dx
    pop
    pop
        ax
    ret
  print endp
;вывод символа
print_symb proc near
     push ax
     push dx
               ah, 02h
     mov
     int
                21h
     pop
                dx
     pop
                ax
     ret
   print_symb endp
```

#### ;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

```
;цепочка блоков управления памятью
              dx, offset mcb
   mov
   call print
    mov dx, offset endl
    call print
       ah, 52h
   mov
   int 21h
       ax, es:[bx-2]
   mov
       es, ax
   mov
   ;тип МСВ
tag1:
    mov al, es:[0000h]
   call byte_to_hex
              di, offset typeMCB+10
   mov
       [di], ax
   mov
          dx, offset typeMCB
   mov
   call print
   mov
             dx, offset tab
   call
         print
   ;сегментный адрес PSP владельца участка памяти
         ax, es:[0001h]
   mov
   mov di, offset adrPSP+15
   call wrd_to_hex
```

```
dx, offset adrPSP
   mov
   call
         print
           dx, offset tab
   mov
   call
         print
   ;размер участка в параграфах
       ax, es:[0003h]
   mov
       cx, 10h
   mov
   mul
         СХ
          si, offset size_s+13
    mov
   call wrd_to_dec
              dx, offset size_s
   mov
   call print
   mov
         dx, offset tab
   call
         print
   ;последние 8 байт
   push ds
   push es
   pop ds
   mov dx, 08h
         di, dx
   mov
         cx, 8
   mov
tag2:
           cx,0
    cmp
              tag3
    je
             dl, byte PTR [di]
   mov
   call
         print_symb
```

dec СX inc di jmp tag2 tag3: pop ds dx, offset endl mov call print ;проверка, последний блок или нет byte ptr es:[0000h], 5ah cmp quit je ;адрес следующего блока ax, es mov ax, es:[0003h] add inc ax es, ax mov jmp tag1 quit: ax, ax xor ah, 4ch mov int 21h SegEnd: PCinfo **ENDS** END **START** 

lab3\_4.asm

PCinfo segment

```
100h
     org
start:
     jmp
                begin
     ;data
     av_mem db 'Amount of available memory:
                                                           b$'
                db 'Size of extended memory:
                                                        Kb$'
     ex mem
     mcb db 'List of memory control blocks:$'
     typeMCB db 'MCB type: 00h$'
     adrPSP
               db 'PSP adress: 0000h$'
     size s db 'Size:
                                   b$'
         db 13, 10, '$'
    endl
   tab
                db 9,'$'
   error db 'ERROR! Memory can not be allocated!$'
tetr_to_hex proc near
        al, 0Fh
   and
        al, 09
   cmp
   jbe
        next
        al, 07
   add
next:
          al, 30h
   add
   ret
  tetr_to_hex endp
;Байт в al переводится в два символа 16-ричного числа в ах
byte_to_hex proc near
   push
          \mathsf{C}\mathsf{X}
```

assume cs:PCinfo, ds:PCinfo, es:nothing, ss:nothing

```
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
          \mathsf{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
        [di], al
   mov
   dec
        di
        al, bh
   mov
   call byte_to_hex
   mov
         [di], ah
   dec
          di
   mov
         [di], al
          bx
   pop
   ret
  wrd_to_hex endp
```

```
byte_to_dec proc near
    push
            \mathsf{cx}
    push
             dx
    xor
           ah, ah
          dx, dx
    xor
            cx, 10
    mov
loop_bd:
    div
             \mathsf{cx}
                   dl, 30h
    or
             [si], dl
    mov
            si
    dec
          dx, dx
    xor
          ax, 10
    cmp
            loop_bd
    jae
            al, 00h
    cmp
    jе
                   end_1
                   al, 30h
    or
             [si], al
    mov
end_1:
           dx
    pop
    pop
             \mathsf{C}\mathsf{X}
    ret
   byte_to_dec endp
wrd_to_dec proc near
    push
             \mathsf{C}\mathsf{X}
    push
             dx
    mov
           cx, 10
wloop_bd:
    div
             \mathsf{cx}
```

```
dl, 30h
    or
           [si], dl
    mov
           si
    dec
     xor dx, dx
         ax, 10
    cmp
         wloop_bd
    jae
           al, 00h
    cmp
                wend_l
    je
                al, 30h
    or
           [si], al
    mov
wend_1:
          dx
    pop
    pop
           \mathsf{C}\mathsf{X}
    ret
   wrd_to_dec endp
;вывод строки
print proc near
    push
           ax
    push
           dx
           ah, 09h
    mov
          21h
    int
    pop
          dx
    pop
           ax
    ret
   print endp
;вывод символа
print_symb proc near
     push ax
```

```
push dx
         ah, 02h
    mov
              21h
    int
    pop
              dx
    pop
         ax
    ret
  print_symb endp
begin:
;количество доступной памяти
    mov ah, 4Ah
    mov bx, Offffh
    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
             al, 71h ;старший байт
    in
             ah, al
    mov
             al, bl
    mov
    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
         dx, offset mcb
   mov
   call print
    mov dx, offset endl
   call print
   mov ah, 52h
   int 21h
   mov
       ax, es:[bx-2]
       es, ax
   mov
   ;тип МСВ
tag1:
    mov al, es:[0000h]
   call byte_to_hex
             di, offset typeMCB+10
   mov
```

```
mov [di], ax
          dx, offset typeMCB
mov
call
      print
        dx, offset tab
mov
call
      print
;сегментный адрес PSP владельца участка памяти
mov
     ax, es:[0001h]
     di, offset adrPSP+15
mov
call wrd_to_hex
           dx, offset adrPSP
mov
call print
mov
      dx, offset tab
call
      print
;размер участка в параграфах
   ax, es:[0003h]
mov
mov cx, 10h
   cx
mul
 mov
           si, offset size_s+13
call wrd_to_dec
           dx, offset size_s
mov
call print
     dx, offset tab
mov
call
      print
;последние 8 байт
```

```
push
          ds
    push
          es
          ds
    pop
        dx, 08h
    mov
        di, dx
    mov
          cx, 8
    mov
tag2:
               cx,0
     cmp
                tag3
     je
                dl, byte PTR [di]
    mov
    call print_symb
    dec
          \mathsf{C}\mathsf{X}
                di
    inc
               tag2
    jmp
tag3:
     pop ds
         dx, offset endl
     mov
    call
          print
    ;проверка, последний блок или нет
          byte ptr es:[0000h], 5ah
    cmp
    je
                quit
    ;адрес следующего блока
    mov
          ax, es
          ax, es:[0003h]
    add
    inc
          ax
         es, ax
    mov
          tag1
    jmp
```

# quit:

xor ax, ax

mov ah, 4ch

int 21h

SegEnd:

PCinfo ENDS

END START