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.

Студент гр. 9381 _____ Николаев А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

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 не страничная 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.

Был написан код .COM который выводит в консоль:

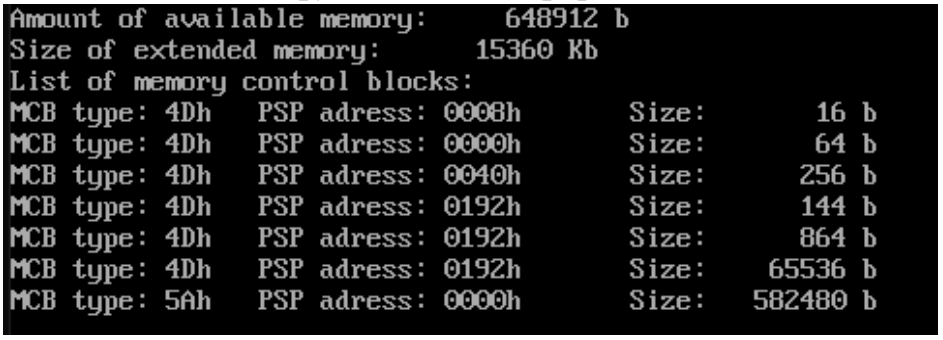
- Количество доступной памяти
- Размер расширенной памяти
- Выводит цепочку битов управления памятью

```
C:\>lab3_1.COM
Available memory:              648912b
Extended memory:             15360  Kb
List of memory control blocks:
MCB type: 4Dh   PSP adress: 0008h       Size:       16 b
MCB type: 4Dh   PSP adress: 0000h       Size:       64 b
MCB type: 4Dh   PSP adress: 0040h       Size:      256 b
MCB type: 4Dh   PSP adress: 0192h       Size:      144 b
MCB type: 5Ah   PSP adress: 0192h       Size:   648912 b        LAB3_1

C:\>
```

Был переписан код таким образом, чтобы он освобождал память, которую не занимает.

Был переписан код, таким образом чтобы он освобождал память и затем запрашивал еще 64кб памяти функцией 48h прерывания 21h.



Был изменен первоначальный вариант программы, запросив 64кб памяти функцией 48h прерывания 21h.

Функции и структуры данных.

availableMemory	Amount of available memory: b$
extendedMemory	Size of extended memory: Kb$
mcbNums	List of memory control blocks:$
mcbType	MCB type: 00h$
pspAdress	PSP adress: 0000h$
sizeS	Size: b$
endLine	13,10 '$'
tab	9, '$'

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 ax.
WRD_TO_HEX	осуществляет перевод числового значения, помещенного в регистр AX, 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 AX, в последовательность символов в десятичной системе счисления, помещая результат в SI. (по аналогии с BYTE_TO_DEC)
PRINT_STRING	осуществляет вывод строки на экран.
PRINT_SYMBOL	осуществляет вывод символа на экран (используя функцию DOS 02h, прерывания 21).

Ответы на контрольные вопросы:

1. Что означает “доступный объем памяти”?

Доступный объем памяти – максимальный объем памяти, который может быть доступен для запуска и выполнения программ.

2. Где MCB блок вашей программы в списке?

У MCB блоков после PSP address – сегментный адрес PSP владельца участка памяти – значение 0192.

3.Какой размер памяти занимает программа в каждом случае?

lab 3_ 1 - Программа заняла всю доступную память.

lab 3_2 - 784 + 144 - объем, занимаемый самой программой.

lab 3_3 - 784 + 144 - объем, занимаемый самой программой и еще 64 кб, которые мы попросили выделить.

lab 3_4 - 784 + 144 - объем, занимаемый самой программой. Выделить 64кб невозможно.

Вывод: Была исследована организация управления памятью. Рассмотрена не страничная память и способ управления динамическими разделами.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

lab3_1.asm

```
LAB   segment
            assume cs:LAB, ds:LAB, es:NOTHING, ss:NOTHING
      org   100h

START:      jmp         BEGIN
      availableMemory   db 'Amount of available memory:            b$'
      extendedMemory    db 'Size of extended memory:           Kb$'
      mcbNums     db 'List of memory control blocks:$'
      mcbType db 'MCB type: 00h$'
      pspAdress   db 'PSP adress: 0000h$'
      sizeS       db 'Size:          b$'
    endLine 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

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
    pop     cx
    ret
   BYTE_TO_HEX endp

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

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

PRINT_STRING proc near
    push    ax
    push    dx
    mov     ah, 09h
    int     21h
```

```
    pop     dx
    pop     ax
    ret
   PRINT_STRING endp

PRINT_SYMBOL proc near
      push  ax
      push  dx
      mov         ah, 02h
      int         21h
      pop         dx
      pop         ax
      ret
   PRINT_SYMBOL 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 availableMemory + 37
      call  WRD_TO_DEC

      mov   dx, offset availableMemory
      call  PRINT_STRING
      mov   dx, offset endLine
      call  PRINT_STRING

;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 extendedMemory + 34
      xor   dx, dx
      call  WRD_TO_DEC

      mov   dx, offset extendedMemory
      call  PRINT_STRING
      mov   dx, offset endLine
      call  PRINT_STRING

;цепочка блоков управления памятью
    mov           dx, offset mcbNums
```

```
    call    PRINT_STRING
      mov         dx, offset endLine
      call  PRINT_STRING

    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 mcbType + 10
    mov     [di], ax

    mov           dx, offset mcbType
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

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

    mov           dx, offset pspAdress
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

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

      mov         si, offset sizeS + 13
    call    WRD_TO_DEC
    mov           dx, offset sizeS
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

    ;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_SYMBOL
```

```
    dec     cx
    inc           di
    jmp           tag2
tag3:
      pop   ds
      mov         dx, offset endLine
    call    PRINT_STRING

    ;проверка на последний блок
    cmp     byte ptr es:[0000h], 5ah
    je            endProgramm

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

endProgramm:

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

lab3_2.asm

```
LAB   segment
            assume cs:LAB, ds:LAB, es:NOTHING, ss:NOTHING
      org   100h

START:      jmp         BEGIN
      availableMemory   db 'Amount of available memory:            b$'
      extendedMemory    db 'Size of extended memory:           Kb$'
      mcbNums     db 'List of memory control blocks:$'
      mcbType db 'MCB type: 00h$'
      pspAdress   db 'PSP adress: 0000h$'
      sizeS       db 'Size:          b$'
    endLine 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

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
    pop     cx
    ret
   BYTE_TO_HEX endp

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

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

PRINT_STRING proc near
    push    ax
    push    dx
    mov     ah, 09h
    int     21h
    pop     dx
    pop     ax
    ret
```

```
   PRINT_STRING endp

PRINT_SYMBOL proc near
      push  ax
      push  dx
      mov         ah, 02h
      int         21h
      pop         dx
      pop         ax
      ret
   PRINT_SYMBOL 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 availableMemory + 37
      call  WRD_TO_DEC

      mov   dx, offset availableMemory
      call  PRINT_STRING
      mov         dx, offset endLine
      call  PRINT_STRING

;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 extendedMemory + 34
      xor   dx, dx
```

```
      call  WRD_TO_DEC

      mov         dx, offset extendedMemory
      call  PRINT_STRING
      mov         dx, offset endLine
      call  PRINT_STRING

;цепочка блоков управления памятью
    mov           dx, offset mcbNums
    call    PRINT_STRING
      mov         dx, offset endLine
      call  PRINT_STRING

    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 mcbType + 10
    mov     [di], ax

    mov           dx, offset mcbType
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

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

    mov           dx, offset pspAdress
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

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

      mov         si, offset sizeS + 13
    call    WRD_TO_DEC
    mov           dx, offset sizeS
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

    ;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_SYMBOL
    dec     cx
    inc           di
    jmp           tag2
tag3:
      pop   ds
      mov         dx, offset endLine
    call    PRINT_STRING

    ;проверка на последний блок
    cmp     byte ptr es:[0000h], 5ah
    je            endProgramm

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

endProgramm:

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

lab3_3.asm

```
LAB   segment
            assume cs:LAB, ds:LAB, es:NOTHING, ss:NOTHING
      org   100h

START:      jmp         BEGIN
      availableMemory   db 'Amount of available memory:            b$'
      extendedMemory    db 'Size of extended memory:           Kb$'
      mcbNums     db 'List of memory control blocks:$'
      mcbType db 'MCB type: 00h$'
      pspAdress   db 'PSP adress: 0000h$'
      sizeS       db 'Size:          b$'
    endLine db  13, 10, '$'
    tab           db    9,'$'
    memoryFail    db 'Memory request failed$'

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 из al -> два символа 16-ричного числа из 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 ; старшая цифра в al,младшая в ah
    pop     cx
    ret
   BYTE_TO_HEX endp

;Перевод 16-ти разрядного числа в 16 сс.
;в ax - число,в 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,в  si - поле 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_STRING proc near
    push    ax
    push    dx
    mov     ah, 09h
    int     21h
    pop     dx
    pop     ax
    ret
   PRINT_STRING endp

;вывод символа
PRINT_SYMBOL proc near
      push  ax
      push  dx
      mov         ah, 02h
      int         21h
      pop         dx
      pop         ax
      ret
   PRINT_SYMBOL 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 availableMemory+37
      call  WRD_TO_DEC

      mov   dx, offset availableMemory
      call  PRINT_STRING
      mov         dx, offset endLine
      call  PRINT_STRING

;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 memoryFail
      call  PRINT_STRING
      mov         dx,   offset endLine
      call  PRINT_STRING
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 extendedMemory+34
      xor   dx, dx
      call  WRD_TO_DEC
```

```
      mov         dx, offset extendedMemory
      call  PRINT_STRING
      mov         dx, offset endLine
      call  PRINT_STRING

;цепочка блоков управления памятью
    mov           dx, offset mcbNums
    call    PRINT_STRING
      mov         dx, offset endLine
      call  PRINT_STRING

    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 mcbType+10
    mov     [di], ax

    mov           dx, offset mcbType
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

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

    mov           dx, offset pspAdress
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

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

      mov         si, offset sizeS+13
    call    WRD_TO_DEC
    mov           dx, offset sizeS
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

    ;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_SYMBOL
    dec     cx
    inc           di
    jmp           tag2
tag3:
      pop   ds
      mov         dx, offset endLine
    call    PRINT_STRING

    ;проверка на последний блок
    cmp     byte ptr es:[0000h], 5ah
    je            endProgramm

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

endProgramm:

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

lab3_4.asm

```
LAB   segment
            assume cs:LAB, ds:LAB, es:NOTHING, ss:NOTHING
      org   100h

START:      jmp         BEGIN
      availableMemory   db 'Amount of available memory:            b$'
      extendedMemory    db 'Size of extended memory:           Kb$'
      mcbNums     db 'List of memory control blocks:$'
      mcbType db 'MCB type: 00h$'
      pspAdress   db 'PSP adress: 0000h$'
      sizeS       db 'Size:          b$'
    endLine db  13, 10, '$'
    tab           db    9,'$'
    memoryFail    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 из al -> два символа 16-ричного числа из 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 ; старшая цифра в al, младшая в ah
    pop     cx
    ret
   BYTE_TO_HEX endp

;Перевод 16-ти разрядного числа в 16 сс.
;в ax - число, в 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,в si - поле 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_STRING proc near
    push    ax
    push    dx
    mov     ah, 09h
    int     21h
    pop     dx
    pop     ax
    ret
   PRINT_STRING endp

;вывод символа
PRINT_SYMBOL proc near
      push  ax
      push  dx
      mov         ah, 02h
      int         21h
      pop         dx
      pop         ax
      ret
   PRINT_SYMBOL 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 availableMemory+37
      call  WRD_TO_DEC

      mov   dx, offset availableMemory
      call  PRINT_STRING
      mov         dx, offset endLine
      call  PRINT_STRING


;запрос памяти
      xor         ax, ax
      mov         ah, 48h
      mov         bx, 1000h
      int         21h
      jnc         mem_ok
      mov         dx, offset memoryFail
      call  PRINT_STRING
      mov         dx,   offset endLine
      call  PRINT_STRING
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 extendedMemory+34
      xor   dx, dx
      call  WRD_TO_DEC

      mov         dx, offset extendedMemory
      call  PRINT_STRING
```

```
      mov         dx, offset endLine
      call  PRINT_STRING

;цепочка блоков управления памятью
    mov           dx, offset mcbNums
    call    PRINT_STRING
      mov         dx, offset endLine
      call  PRINT_STRING

    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 mcbType+10
    mov     [di], ax

    mov           dx, offset mcbType
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

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

    mov           dx, offset pspAdress
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

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

      mov         si, offset sizeS+13
    call    WRD_TO_DEC
    mov           dx, offset sizeS
    call    PRINT_STRING
    mov           dx, offset tab
    call    PRINT_STRING

    ;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_SYMBOL
    dec     cx
    inc           di
    jmp           tag2
tag3:
      pop   ds
      mov         dx, offset endLine
    call    PRINT_STRING

    ;проверка на последний блок
    cmp     byte ptr es:[0000h], 5ah
    je            endProgramm

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

endProgramm:

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