

**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 память**

Студентка гр. 9382

\_\_\_\_\_

Пя С.

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 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.

### **Задание.**

Шаг 1. Для выполнения лабораторной 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. Объем памяти функциями управления памятью выводится в параграфах. Необходимо преобразовать его в байты и выводить в виде десятичных чисел. Последние восемь байт MSB выводятся как символы, не следует преобразовывать их в шестнадцатеричные числа.

Запустите программу и внимательно оцените результаты. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 2. Измените программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah прерывания 21h (пример в разделе «Использование функции 4АН»). Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущем шаге. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 3. Измените программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H. Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущих шагах. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 4. Измените первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти. Обязательно обрабатывайте завершение функций ядра, проверяя флаг CF. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 5. Оцените результаты, полученные на предыдущих шагах. Ответьте на контрольные вопросы и оформите отчет.

#### **Необходимые сведения для составления программы.**

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

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 f52h "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

### **Сведения о функциях и структурах.**

TETR\_TO\_HEX: процедура перевода из 10-ой сс в символы

BYTE\_TO\_HEX: процедура перевода байта из 16-ой сс в символы

WRD\_TO\_HEX: перевод слова из 16-ой сс в символы

BYTE\_TO\_DEC: перевод байта из 16-ой сс в 10-ую и символы

print\_string: процедура вывода строки на экран

para\_to\_byte: процедура перевода из параграфа в байты

print\_n proc: процедура вывода новой строки

print\_available\_mem\_size: процедура вывода количества доступной информации

print\_extended\_mem\_size: процедура вывода размера расширенной памяти

print\_mcb: процедура вывода содержимого MCB

offset\_dec: процедура поиска символа

print\_mcb\_list: процедура вывода цепочки блоков управления памятью

delete\_free\_memory: процедура освобождения памяти

memory\_request: процедура запроса памяти

### **Выполнение работы.**

1. Выводится требуемая информация с помощью процедур print\_available\_mem\_size, print\_extended\_mem\_size, print\_mcb\_list.

```
To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT C "E:\ИИ 2021"
Drive C is mounted as local directory E:\ИИ 2021\

Z:\>C:

C:\>LB3.COM
available memory size: 648912 bytes
extended memory size: 246720 bytes
MCB #1: address: 016F PSP address: 0008 size: 16 SC/SD:
MCB #2: address: 0171 PSP address: 0000 size: 64 SC/SD:
MCB #3: address: 0176 PSP address: 0040 size: 256 SC/SD:
MCB #4: address: 0187 PSP address: 0192 size: 144 SC/SD:
MCB #5: address: 0191 PSP address: 0192 size: 648912 SC/SD: LB3

C:\>_
```

2. Программа переписана, теперь она освобождает неиспользуемую память.

```
To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT C "E:\ИИ 2021"
Drive C is mounted as local directory E:\ИИ 2021\

Z:\>C:

C:\>LB3_1.COM
available memory size: 648912 bytes
extended memory size: 246720 bytes
MCB #1: address: 016F PSP address: 0008 size: 16 SC/SD:
MCB #2: address: 0171 PSP address: 0000 size: 64 SC/SD:
MCB #3: address: 0176 PSP address: 0040 size: 256 SC/SD:
MCB #4: address: 0187 PSP address: 0192 size: 144 SC/SD:
MCB #5: address: 0191 PSP address: 0192 size: 848 SC/SD: LB3_1
MCB #6: address: 01C7 PSP address: 0000 size: 648048 SC/SD:

C:\>
```

3. Программа переписана, теперь после освобождения неиспользуемой памяти запрашивается 64КБ памяти.

```

For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT C "E:\Рлї 2021"
Drive C is mounted as local directory E:\Рлї 2021\

Z:\>C:

C:\>LB3_2.COM
available memory size: 648912 bytes
extended memory size: 246720 bytes
New memory has been added!
MCB #1: address: 016F PSP address: 0008 size: 16 SC/SD:
MCB #2: address: 0171 PSP address: 0000 size: 64 SC/SD:
MCB #3: address: 0176 PSP address: 0040 size: 256 SC/SD:
MCB #4: address: 0187 PSP address: 0192 size: 144 SC/SD:
MCB #5: address: 0191 PSP address: 0192 size: 944 SC/SD: LB3_2
MCB #6: address: 01CD PSP address: 0192 size: 65536 SC/SD: LB3_2
MCB #7: address: 11CE PSP address: 0000 size: 582400 SC/SD:

C:\>

```

4. Программа переписана, теперь до освобождения неиспользуемой памяти запрашивается 64КБ памяти.

```

To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT C "E:\Рлї 2021"
Drive C is mounted as local directory E:\Рлї 2021\

Z:\>C:

C:\>LB3_3.COM
available memory size: 648912 bytes
extended memory size: 246720 bytes
New memory hasn't been added!
MCB #1: address: 016F PSP address: 0008 size: 16 SC/SD:
MCB #2: address: 0171 PSP address: 0000 size: 64 SC/SD:
MCB #3: address: 0176 PSP address: 0040 size: 256 SC/SD:
MCB #4: address: 0187 PSP address: 0192 size: 144 SC/SD:
MCB #5: address: 0191 PSP address: 0192 size: 944 SC/SD: LB3_3
MCB #6: address: 01CD PSP address: 0000 size: 647952 SC/SD:

C:\>_

```

### Выводы.

В ходе выполнения лабораторной работы была написана программа для вывода требуемой информации о памяти и управления разделами памяти, исследованы структуры данных и работа функций управления памятью ядра операционной системы.

### **Контрольные вопросы по лабораторной работе №3**

1) Что означает "доступный объем памяти"?

Доступный объем памяти – это область памяти, которая не занята процессами системы и может выделяться для использования.

2) Где МСВ блок Вашей программы в списке?

МСВ блок программы находится в конце списка МСВ блоков, где перед ним будет находиться среда, принадлежащая программе, блок можно отследить по одинаковому сегментному адресу PSP.

3) Какой размер памяти занимает программа в каждом случае.

Программа занимает весь доступный объем памяти на 1 шаге.

Программа занимает 848 байт памяти на 2 шаге.

Программа занимает 66480 байт памяти на 3 шаге.

Программа занимает 944 байт памяти на 4 шаге.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Lb3.ASM

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN
; ДАННЫЕ
n db 0dh, 0ah, '$'
area_size db "size:      ", '$'
SC_SD db "SC/SD: ", '$'
address db "address:   ", '$'
PSP_address db "PSP address: ", '$'
extended_mem_size db "extended memory size:      bytes", 0dh, 0ah, '$'
available_mem_size db "available memory size:      bytes", 0dh, 0ah, '$'
MCB_number db "MCB #", '$'
dec_number db "      ", '$'

; ПРОЦЕДУРЫ
;-----
TETR_TO_HEX PROC near
    and AL,0Fh
    cmp AL,09
    jbe NEXT
    add AL,07
NEXT: add AL,30h
    ret
TETR_TO_HEX ENDP
;-----
BYTE_TO_HEX PROC near
; байт в AL переводится в два символа шестн. числа в AX
    push CX
    mov AH,AL
    call TETR_TO_HEX
    xchg AL,AH
    mov CL,4
    shr AL,CL
    call TETR_TO_HEX ;в AL старшая цифра
    pop CX ;в AH младшая
    ret
BYTE_TO_HEX ENDP
;-----
WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
    push BX
    mov BH,AH
    call BYTE_TO_HEX
    mov [DI],AH
    dec DI
    mov [DI],AL
    dec DI
    mov AL,BH
    call BYTE_TO_HEX
    mov [DI],AH
    dec DI
    mov [DI],AL
    pop BX
    ret
WRD_TO_HEX ENDP
;-----
```



```

BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
push CX
push DX
xor AH,AH
xor DX,DX
mov CX,10
loop_bd:
div CX
or DL,30h
mov [SI],DL
dec SI
xor DX,DX
cmp AX,10
jae loop_bd
cmp AL,00h
je end_1
or AL,30h
mov [SI],AL
end_1:
pop DX
pop CX
ret
BYTE_TO_DEC ENDP
;-----
; КОД
print_string proc near
mov ah, 09h
int 21h
ret
print_string endp

para_to_byte proc
push ax
push bx
push cx
push dx
push si

mov bx, 10h
mul bx
mov bx, 0ah
xor cx, cx

div_loop:
div bx
push dx
inc cx
sub dx, dx
cmp ax, 0h
jnz div_loop

print_sym:
pop dx
add dl, 30h
mov [si], dl
inc si
loop print_sym

pop si
pop dx
pop cx
pop bx
pop ax

```

```

ret
para_to_byte endp

print_n proc near
    push ax
    push dx

    mov dx, offset n
    mov ah, 9h
    int 21h

    pop dx
    pop ax
ret
print_n endp

print_available_mem_size proc near
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov si, offset available_mem_size
    add si, 23
    call para_to_byte
    mov dx, offset available_mem_size
    call print_string
ret
print_available_mem_size endp

print_extended_mem_size proc near
    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
    mov ah, al
    mov si, offset extended_mem_size
    add si, 22
    call para_to_byte
    mov dx, offset extended_mem_size
    call print_string
ret
print_extended_mem_size endp

print_mcb proc near
    push ax
    push dx
    push si
    push di
    push cx

    mov ax, es;MCB
    mov di, offset address
    add di, 12
    call WRD_TO_HEX
    mov dx, offset address
    call print_string
    mov ax, es:[1] ;PSP
    mov di, offset PSP_address
    add di, 16
    call WRD_TO_HEX

```

```

    mov dx, offset PSP_address
    call print_string
    mov ax, es:[3] ;size of para
    mov si, offset area_size
    add si, 6
    call para_to_byte
    mov dx, offset area_size
    call print_string
    mov bx, 8 ;SC SD
    mov dx, offset SC_SD
    call print_string
    mov cx, 7
print_sc_sd_loop:
    mov dl, es:[bx]
    mov ah, 02h
    int 21h
    inc bx
    loop print_sc_sd_loop

    pop cx
    pop di
    pop si
    pop dx
    pop ax
ret
print_mcb endp

offset_dec proc near
    offset_dec_loop:
        cmp byte ptr [si], ' '
        jne end_offset_dec
        inc si
        jmp offset_dec_loop
    end_offset_dec:
ret
offset_dec endp

print_mcb_list proc near
    push ax
    push bx
    push es
    push dx

    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    mov es, ax
    mov cl, 1
    print_list:
        mov dx, offset MCB_number
        call print_string
        mov al, cl
        mov si, offset dec_number
        add si, 2
        call BYTE_TO_DEC
        call offset_dec
        mov dx, si
        call print_string
        mov dl, ':'
        mov ah, 02h
        int 21h
        mov dl, ' '
        mov ah, 02h
        int 21h

```

```

        call print_mcb
        call print_n
        mov al, es:[0]
        cmp al, 5ah
        je end_mcb_list

        mov bx, es:[3]
        mov ax, es
        add ax, bx
        inc ax
        mov es, ax
        inc cl
        jmp print_list

end_mcb_list:
        pop dx
        pop es
        pop bx
        pop ax
ret
print_mcb_list endp

BEGIN:
; . . . . .
        call print_available_mem_size
        call print_extended_mem_size
        call print_mcb_list
; . . . . .
; Выход в DOS
xor AL,AL
mov AH,4Ch
int 21H
TESTPC ENDS
END START ;конец модуля, START - точка входа

```

## Название файла: Lb3\_1.ASM

```

TESTPC SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG 100H
START: JMP BEGIN
; ДАННЫЕ
n db 0dh, 0ah, '$'
area_size db "size:      ", '$'
SC_SD db "SC/SD: ", '$'
address db "address:    ", '$'
PSP_address db "PSP address: ", '$'
extended_mem_size db "extended memory size:      bytes", 0dh, 0ah, '$'
available_mem_size db "available memory size:      bytes", 0dh, 0ah, '$'
MCB_number db "MCB #", '$'
dec_number db " ", '$'

; ПРОЦЕДУРЫ
;-----
TETR_TO_HEX PROC near
        and AL,0Fh
        cmp AL,09
        jbe NEXT
        add AL,07
NEXT: add AL,30h
        ret
TETR_TO_HEX ENDP
;-----

```

```

BYTE_TO_HEX PROC near
; байт в AL переводится в два символа шестн. числа в AX
push CX
mov AH,AL
call TETR_TO_HEX
xchg AL,AH
mov CL,4
shr AL,CL
call TETR_TO_HEX ;в AL старшая цифра
pop CX ;в AH младшая
ret
BYTE_TO_HEX ENDP
;-----
WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
push BX
mov BH,AH
call BYTE_TO_HEX
mov [DI],AH
dec DI
mov [DI],AL
dec DI
mov AL,BH
call BYTE_TO_HEX
mov [DI],AH
dec DI
mov [DI],AL
pop BX
ret
WRD_TO_HEX ENDP
;-----
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
push CX
push DX
xor AH,AH
xor DX,DX
mov CX,10
loop_bd:
div CX
or DL,30h
mov [SI],DL
dec SI
xor DX,DX
cmp AX,10
jae loop_bd
cmp AL,00h
je end_1
or AL,30h
mov [SI],AL
end_1:
pop DX
pop CX
ret
BYTE_TO_DEC ENDP
;-----
; КОД
print_string proc near
mov ah, 09h
int 21h
ret
print_string endp

```

```

para_to_byte proc
    push ax
    push bx
    push cx
    push dx
    push si

    mov bx, 10h
    mul bx
    mov bx, 0ah
    xor cx, cx

div_loop:
    div bx
    push dx
    inc cx
    sub dx, dx
    cmp ax, 0h
    jnz div_loop

print_sym:
    pop dx
    add dl, 30h
    mov [si], dl
    inc si
loop print_sym

    pop si
    pop dx
    pop cx
    pop bx
    pop ax
ret
para_to_byte endp

print_n proc near
    push ax
    push dx

    mov dx, offset n
    mov ah, 9h
    int 21h

    pop dx
    pop ax
ret
print_n endp

print_available_mem_size proc near
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov si, offset available_mem_size
    add si, 23
    call para_to_byte
    mov dx, offset available_mem_size
    call print_string
ret
print_available_mem_size endp

print_extended_mem_size proc near
    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
    mov ah, al
    mov si, offset extended_mem_size
    add si, 22
    call para_to_byte
    mov dx, offset extended_mem_size
    call print_string
ret
print_extended_mem_size endp

print_mcb proc near
    push ax
    push dx
    push si
    push di
    push cx

    mov ax, es;MCB
    mov di, offset address
    add di, 12
    call WRD_TO_HEX
    mov dx, offset address
    call print_string
    mov ax, es:[1] ;PSP
    mov di, offset PSP_address
    add di, 16
    call WRD_TO_HEX
    mov dx, offset PSP_address
    call print_string
    mov ax, es:[3] ;size of para
    mov si, offset area_size
    add si, 6
    call para_to_byte
    mov dx, offset area_size
    call print_string
    mov bx, 8 ;SC SD
    mov dx, offset SC_SD
    call print_string
    mov cx, 7
print_sc_sd_loop:
    mov dl, es:[bx]
    mov ah, 02h
    int 21h
    inc bx
    loop print_sc_sd_loop

    pop cx
    pop di
    pop si
    pop dx
    pop ax
ret
print_mcb endp

offset_dec proc near
    offset_dec_loop:
        cmp byte ptr [si], ' '
        jne end_offset_dec
        inc si

```

```

        jmp offset_dec_loop
    end_offset_dec:
ret
offset_dec endp

print_mcb_list proc near
    push ax
    push bx
    push es
    push dx

    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    mov es, ax
    mov cl, 1
    print_list:
        mov dx, offset MCB_number
        call print_string
        mov al, cl
        mov si, offset dec_number
        add si, 2
        call BYTE_TO_DEC
        call offset_dec
        mov dx, si
        call print_string
        mov dl, ':'
        mov ah, 02h
        int 21h
        mov dl, ' '
        mov ah, 02h
        int 21h
        call print_mcb
        call print_n
        mov al, es:[0]
        cmp al, 5ah
    je end_mcb_list

    mov bx, es:[3]
    mov ax, es
    add ax, bx
    inc ax
    mov es, ax
    inc cl
    jmp print_list

end_mcb_list:
    pop dx
    pop es
    pop bx
    pop ax
ret
print_mcb_list endp

delete_free_memory proc near
    push ax
    push bx
    push cx
    push dx

    lea ax, final_end
    mov bx, 10h
    sub dx, dx
    div bx

```



```

        inc ax
        mov bx,ax
        mov al,0
        mov ah,4Ah
        int 21h

        pop dx
        pop cx
        pop bx
        pop ax
ret
delete_free_memory endp

BEGIN:

        call print_available_mem_size
        call print_extended_mem_size

        call delete_free_memory
        call print_mcb_list

; Выход в DOS
xor AL,AL
mov AH,4Ch
int 21H
final_end:
TESTPC ENDS
END START ;конец модуля, START - точка входа

```

Название файла: Lb3\_2.ASM

```

TESTPC SEGMENT
ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
ORG 100H
START: JMP BEGIN
; ДАННЫЕ
n db 0dh, 0ah, '$'
area_size db "size:      ", '$'
SC_SD db "SC/SD: ", '$'
address db "address:    ", '$'
PSP_address db "PSP address:      ", '$'
extended_mem_size db "extended memory size:      bytes", 0dh, 0ah, '$'
available_mem_size db "available memory size:      bytes", 0dh, 0ah, '$'
MCB_number db "MCB #", '$'
dec_number db " ", '$'
mem_accept db "New memory has been added!", 0dh, 0ah, '$'
mem_fail db "New memory hasn't been added!", '$'

;ПРОЦЕДУРЫ
;-----
TETR_TO_HEX PROC near
    and AL,0Fh
    cmp AL,09
    jbe NEXT
    add AL,07
NEXT: add AL,30h
    ret
TETR_TO_HEX ENDP
;-----
BYTE_TO_HEX PROC near
; байт в AL переводится в два символа шестн. числа в AX
    push CX
    mov AH,AL
    call TETR_TO_HEX
    xchg AL,AH

```

```

mov CL,4
shr AL,CL
call TETR_TO_HEX ;в AL старшая цифра
pop CX ;в AH младшая
ret
BYTE_TO_HEX ENDP
;-----
WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
push BX
mov BH,AH
call BYTE_TO_HEX
mov [DI],AH
dec DI
mov [DI],AL
dec DI
mov AL,BH
call BYTE_TO_HEX
mov [DI],AH
dec DI
mov [DI],AL
pop BX
ret
WRD_TO_HEX ENDP
;-----
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
push CX
push DX
xor AH,AH
xor DX,DX
mov CX,10
loop_bd:
div CX
or DL,30h
mov [SI],DL
dec SI
xor DX,DX
cmp AX,10
jae loop_bd
cmp AL,00h
je end_1
or AL,30h
mov [SI],AL
end_1:
pop DX
pop CX
ret
BYTE_TO_DEC ENDP
;-----
; КОД
print_string proc near
mov ah, 09h
int 21h
ret
print_string endp

para_to_byte proc
push ax
push bx
push cx
push dx
push si

```

```

        mov bx, 10h
        mul bx
        mov bx, 0ah
        xor cx, cx

div_loop:
        div bx
        push dx
        inc cx
        sub dx, dx
        cmp ax, 0h
        jnz div_loop

print_sym:
        pop dx
        add dl, 30h
        mov [si], dl
        inc si
loop print_sym

        pop si
        pop dx
        pop cx
        pop bx
        pop ax
ret
para_to_byte endp

print_n proc near
        push ax
        push dx

        mov dx, offset n
        mov ah, 9h
        int 21h

        pop dx
        pop ax
ret
print_n endp

print_available_mem_size proc near
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h
        mov ax, bx
        mov si, offset available_mem_size
        add si, 23
        call para_to_byte
        mov dx, offset available_mem_size
        call print_string
ret
print_available_mem_size endp

print_extended_mem_size proc near
        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

```

```

        mov ah, al
        mov si, offset extended_mem_size
        add si, 22
        call para_to_byte
        mov dx, offset extended_mem_size
        call print_string
ret
print_extended_mem_size endp

print_mcb proc near
    push ax
    push dx
    push si
    push di
    push cx

    mov ax, es;MCB
    mov di, offset address
    add di, 12
    call WRD_TO_HEX
    mov dx, offset address
    call print_string
    mov ax, es:[1] ;PSP
    mov di, offset PSP_address
    add di, 16
    call WRD_TO_HEX
    mov dx, offset PSP_address
    call print_string
    mov ax, es:[3] ;size of para
    mov si, offset area_size
    add si, 6
    call para_to_byte
    mov dx, offset area_size
    call print_string
    mov bx, 8 ;SC SD
    mov dx, offset SC_SD
    call print_string
    mov cx, 7
    print_sc_sd_loop:
        mov dl, es:[bx]
        mov ah, 02h
        int 21h
        inc bx
        loop print_sc_sd_loop

    pop cx
    pop di
    pop si
    pop dx
    pop ax
ret
print_mcb endp

offset_dec proc near
    offset_dec_loop:
        cmp byte ptr [si], ' '
        jne end_offset_dec
        inc si
        jmp offset_dec_loop
    end_offset_dec:
ret
offset_dec endp

print_mcb_list proc near

```

```

push ax
push bx
push es
push dx

mov ah, 52h
int 21h
mov ax, es:[bx-2]
mov es, ax
mov cl, 1
print_list:
    mov dx, offset MCB_number
    call print_string
    mov al, cl
    mov si, offset dec_number
    add si, 2
    call BYTE_TO_DEC
    call offset_dec
    mov dx, si
    call print_string
    mov dl, ':'
    mov ah, 02h
    int 21h
    mov dl, ' '
    mov ah, 02h
    int 21h
    call print_mcb
    call print_n
    mov al, es:[0]
    cmp al, 5ah
    je end_mcb_list

    mov bx, es:[3]
    mov ax, es
    add ax, bx
    inc ax
    mov es, ax
    inc cl
    jmp print_list

end_mcb_list:
    pop dx
    pop es
    pop bx
    pop ax
ret
print_mcb_list endp

delete_free_memory proc near
    push ax
    push bx
    push cx
    push dx

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

```

```

        pop dx
        pop cx
        pop bx
        pop ax
ret
delete_free_memory endp

memory_request proc near
    push ax
    push bx
    push dx

    mov bx, 1000h
    mov ah, 48h
    int 21h
    jc mem_failed
    jmp mem_accepted

mem_failed:
    mov dx, offset mem_fail
    call print_string
    jmp end_memory_request

mem_accepted:
    mov dx, offset mem_accept
    call print_string

end_memory_request:
    pop dx
    pop bx
    pop ax
ret
memory_request endp

BEGIN:

    call print_available_mem_size
    call print_extended_mem_size

    call delete_free_memory
    call memory_request
    call print_mcb_list

; Выход в DOS
xor AL,AL
mov AH,4Ch
int 21H
final_end:
TESTPC ENDS
END START ;конец модуля, START - точка входа

```

## Название файла: Lb3\_3.ASM

```

TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN
; ДАННЫЕ
n db 0dh, 0ah, '$'
area_size db "size:      ", '$'
SC_SD db "SC/SD: ", '$'
address db "address:    ", '$'
PSP_address db "PSP address:      ", '$'
extended_mem_size db "extended memory size:      bytes", 0dh, 0ah, '$'
available_mem_size db "available memory size:      bytes", 0dh, 0ah, '$'

```

```

MCB_number db "MCB #", '$'
dec_number db " ", '$'
mem_accept db "New memory has been added!", 0dh, 0ah, '$'
mem_fail db "New memory hasn't been added!", '$'

```

```
;ПРОЦЕДУРЫ
```

```
;-----
```

```
TETR_TO_HEX PROC near
```

```
    and AL,0Fh
```

```
    cmp AL,09
```

```
    jbe NEXT
```

```
    add AL,07
```

```
NEXT: add AL,30h
```

```
    ret
```

```
TETR_TO_HEX ENDP
```

```
;-----
```

```
BYTE_TO_HEX PROC near
```

```
; байт в AL переводится в два символа шестн. числа в AX
```

```
    push CX
```

```
    mov AH,AL
```

```
    call TETR_TO_HEX
```

```
    xchg AL,AH
```

```
    mov CL,4
```

```
    shr AL,CL
```

```
    call TETR_TO_HEX ;в AL старшая цифра
```

```
    pop CX ;в AH младшая
```

```
    ret
```

```
BYTE_TO_HEX ENDP
```

```
;-----
```

```
WRD_TO_HEX PROC near
```

```
;перевод в 16 с/с 16-ти разрядного числа
```

```
; в AX - число, DI - адрес последнего символа
```

```
    push BX
```

```
    mov BH,AH
```

```
    call BYTE_TO_HEX
```

```
    mov [DI],AH
```

```
    dec DI
```

```
    mov [DI],AL
```

```
    dec DI
```

```
    mov AL,BH
```

```
    call BYTE_TO_HEX
```

```
    mov [DI],AH
```

```
    dec DI
```

```
    mov [DI],AL
```

```
    pop BX
```

```
    ret
```

```
WRD_TO_HEX ENDP
```

```
;-----
```

```
BYTE_TO_DEC PROC near
```

```
; перевод в 10с/с, SI - адрес поля младшей цифры
```

```
    push CX
```

```
    push DX
```

```
    xor AH,AH
```

```
    xor DX,DX
```

```
    mov CX,10
```

```
loop_bd:
```

```
    div CX
```

```
    or DL,30h
```

```
    mov [SI],DL
```

```
    dec SI
```

```
    xor DX,DX
```

```
    cmp AX,10
```

```
    jae loop_bd
```

```
    cmp AL,00h
```

```

    je end_1
    or AL, 30h
    mov [SI], AL
end_1:
    pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP
;-----
; КОД
print_string proc near
    mov ah, 09h
    int 21h
    ret
print_string endp

para_to_byte proc
    push ax
    push bx
    push cx
    push dx
    push si

    mov bx, 10h
    mul bx
    mov bx, 0ah
    xor cx, cx

div_loop:
    div bx
    push dx
    inc cx
    sub dx, dx
    cmp ax, 0h
    jnz div_loop

print_sym:
    pop dx
    add dl, 30h
    mov [si], dl
    inc si
loop print_sym

    pop si
    pop dx
    pop cx
    pop bx
    pop ax
ret
para_to_byte endp

print_n proc near
    push ax
    push dx

    mov dx, offset n
    mov ah, 9h
    int 21h

    pop dx
    pop ax
ret
print_n endp

```



```

print_available_mem_size proc near
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov si, offset available_mem_size
    add si, 23
    call para_to_byte
    mov dx, offset available_mem_size
    call print_string
ret
print_available_mem_size endp

print_extended_mem_size proc near
    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
    mov ah, al
    mov si, offset extended_mem_size
    add si, 22
    call para_to_byte
    mov dx, offset extended_mem_size
    call print_string
ret
print_extended_mem_size endp

print_mcb proc near
    push ax
    push dx
    push si
    push di
    push cx

    mov ax, es;MCB
    mov di, offset address
    add di, 12
    call WRD_TO_HEX
    mov dx, offset address
    call print_string
    mov ax, es:[1] ;PSP
    mov di, offset PSP_address
    add di, 16
    call WRD_TO_HEX
    mov dx, offset PSP_address
    call print_string
    mov ax, es:[3] ;size of para
    mov si, offset area_size
    add si, 6
    call para_to_byte
    mov dx, offset area_size
    call print_string
    mov bx, 8 ;SC SD
    mov dx, offset SC_SD
    call print_string
    mov cx, 7
    print_sc_sd_loop:
        mov dl, es:[bx]
        mov ah, 02h
        int 21h

```

```

        inc bx
        loop print_sc_sd_loop

    pop cx
    pop di
    pop si
    pop dx
    pop ax
ret
print_mcb endp

offset_dec proc near
    offset_dec_loop:
        cmp byte ptr [si], ' '
        jne end_offset_dec
        inc si
        jmp offset_dec_loop
    end_offset_dec:
ret
offset_dec endp

print_mcb_list proc near
    push ax
    push bx
    push es
    push dx

    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    mov es, ax
    mov cl, 1
    print_list:
        mov dx, offset MCB_number
        call print_string
        mov al, cl
        mov si, offset dec_number
        add si, 2
        call BYTE_TO_DEC
        call offset_dec
        mov dx, si
        call print_string
        mov dl, ':'
        mov ah, 02h
        int 21h
        mov dl, ' '
        mov ah, 02h
        int 21h
        call print_mcb
        call print_n
        mov al, es:[0]
        cmp al, 5ah
        je end_mcb_list

        mov bx, es:[3]
        mov ax, es
        add ax, bx
        inc ax
        mov es, ax
        inc cl
        jmp print_list

end_mcb_list:
    pop dx

```

```

        pop es
        pop bx
        pop ax
ret
print_mcb_list endp

delete_free_memory proc near
    push ax
    push bx
    push cx
    push dx

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

    pop dx
    pop cx
    pop bx
    pop ax
ret
delete_free_memory endp

memory_request proc near
    push ax
    push bx
    push dx

    mov bx, 1000h
    mov ah, 48h
    int 21h
    jc mem_failed
    jmp mem_accepted

mem_failed:
    mov dx, offset mem_fail
    call print_string
    jmp end_memory_request

mem_accepted:
    mov dx, offset mem_accept
    call print_string

end_memory_request:
    pop dx
    pop bx
    pop ax
ret
memory_request endp

BEGIN:

    call print_available_mem_size
    call print_extended_mem_size

    call memory_request
    call delete_free_memory
    call print_mcb_list

```

```
; Выход в DOS
xor AL,AL
mov AH,4Ch
int 21H
final_end:
TESTPC ENDS
END START ;конец модуля, START - точка входа
```