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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Операционные системы»

Тема: Исследование организации управления основной памятью.

Студентка гр. 0382 _____ Михайлова О.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

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 MCB 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 4AH»). 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.

Выполнение работы.

Для выполнения задания был использован шаблон из методических указаний с процедурами перевода двоичных кодов в символы шестнадцатеричных чисел и десятичное число: TETR_TO_HEX, BYTE_TO_HEX, WRD_TO_HEX, BYTE_TO_DEC.

Также в программу были добавлены следующие процедуры:

- PRINT_STRING – процедура вывода строки на экран;
- PRINT_SYMB – процедура вывода символа на экран;
- WRD_TO_DEC – перевод в десятичную систему счисления;
- AVMEM – вывод в консоль размера доступной памяти;
- EXMEM – вывод в консоль размера расширенной памяти;
- MCB – вывод в консоль цепочки блоков управления памятью;
- FREE_MEM – освобождение памяти, не используемой программой;
- REQUEST_MEM – запрос 64Кб памяти и проверка флага CF.

Шаг 1. Был написан и отлажен программный модуль .COM, который выводит на экран количество доступной памяти, размер расширенной памяти и цепочку блоков управления памятью.

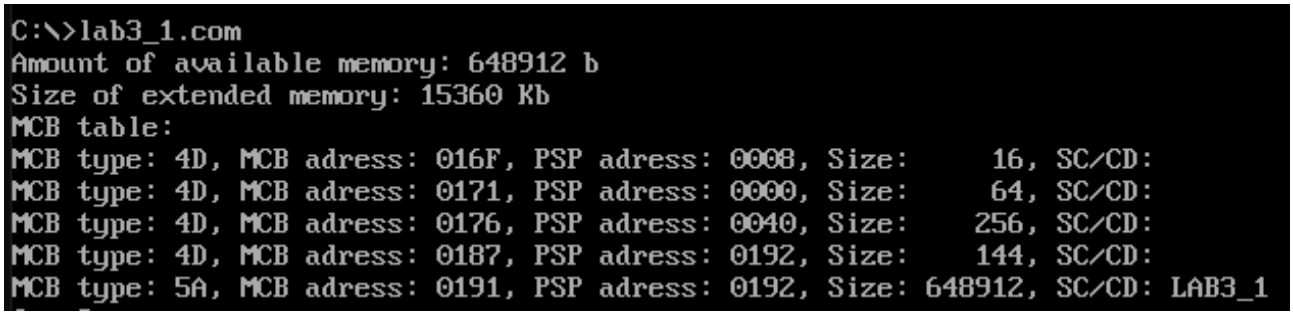

Рисунок 1 - результат запуска модуля lab3_1.com

Шаг 2. Было добавлено освобождение памяти, которую не занимает программа (процедура FREE_MEM). Для этого была использована функция 4Ah прерывания 21h.

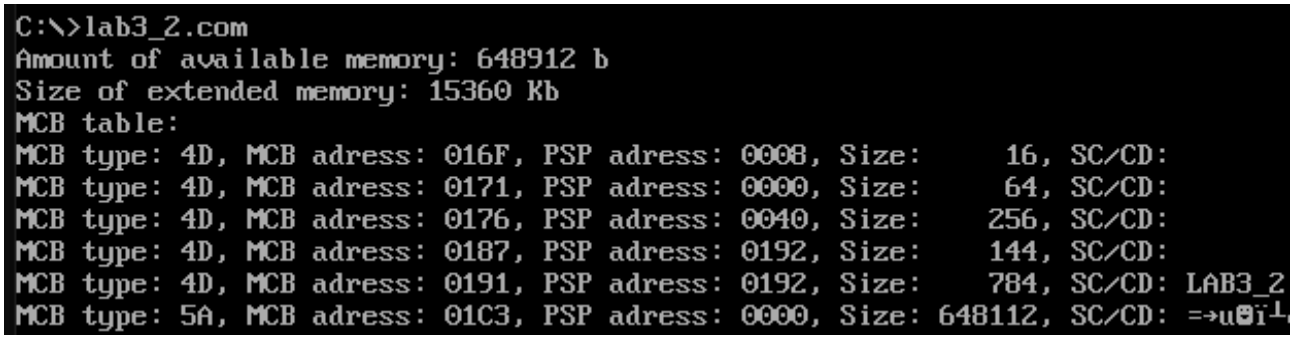

Рисунок 2 - результат запуска модуля lab3_2.com

По результатам запуска второго модуля можно заметить, что программа занимает ту память, которая ей необходима в отличие от первого пункта, где она занимала всю доступную память

Шаг 3. Была добавлена процедура REQUEST_MEM, с помощью которой после освобождения памяти запрашивается 64Кб памяти функцией 48H прерывания 21H.

```
C:\>lab3_3.com
Amount of available memory: 648912 b
Size of extended memory: 15360 Kb
MCB table:
MCB type: 4D, MCB adress: 016F, PSP adress: 0008, Size:     16, SC/CD:

MCB type: 4D, MCB adress: 0171, PSP adress: 0000, Size:     64, SC/CD:

MCB type: 4D, MCB adress: 0176, PSP adress: 0040, Size:    256, SC/CD:

MCB type: 4D, MCB adress: 0187, PSP adress: 0192, Size:    144, SC/CD:

MCB type: 4D, MCB adress: 0191, PSP adress: 0192, Size:    816, SC/CD:    LAB3_3

MCB type: 4D, MCB adress: 01C5, PSP adress: 0192, Size:  65536, SC/CD:    LAB3_3

MCB type: 5A, MCB adress: 11C6, PSP adress: 0000, Size: 582528, SC/CD:
```

Рисунок 3 - Результат запуска модуля lab3_3.com

Шаг 4. Программа была изменена так, что 64Кб памяти запрашивается до освобождения памяти.

```
C:\>lab3_4.com
Amount of available memory: 648912 b
Size of extended memory: 15360 Kb
Memory can not be allocated.
MCB table:
MCB type: 4D, MCB adress: 016F, PSP adress: 0008, Size:     16, SC/CD:

MCB type: 4D, MCB adress: 0171, PSP adress: 0000, Size:     64, SC/CD:

MCB type: 4D, MCB adress: 0176, PSP adress: 0040, Size:    256, SC/CD:

MCB type: 4D, MCB adress: 0187, PSP adress: 0192, Size:    144, SC/CD:

MCB type: 4D, MCB adress: 0191, PSP adress: 0192, Size:    848, SC/CD:   LAB3_4

MCB type: 5A, MCB adress: 01C7, PSP adress: 0000, Size: 648048, SC/CD:   ì>? ╣
```

Рисунок 4 - Результат запуска модуля lab3_4.com

В результате запуска последнего модуля видно, что память не была выделена. Это связано с тем, что на данном этапе программе уже принадлежит вся свободная память.

Исходный код программы см. в приложении А.

Ответы на контрольные вопросы.

1. Что означает "доступный объем памяти"?

Доступный объем памяти – часть оперативной памяти, которую занимает и использует программа.

2. Где MCB блок Вашей программы в списке?

На шагах 1, 2 и 4 MCB блок на 5-ом месте в списке MCB. На 3-м шаге MCB блок на 5-м и 6-м местах в списке MCB, так как на этом шаге в программе был выделен блок памяти 64Кб.

3. Какой размер памяти занимает программа в каждом случае?

Шаг 1: 648912 байт (всю выделенную память)

Шаг 2: 784 байт (только объем памяти, занимаемый программой)

Шаг 3: 65536 байт (объем памяти, занимаемый программой + выделенные по запросу 64Кб памяти)

Шаг 4: 848 байт (только объем памяти, занимаемый программой, так как выделение 64Кб памяти было невозможно)

Выводы.

В ходе работы были исследованы структуры данных и работа функций управления памятью ядра операционной системы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3_1.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN

; Данные
AV_MEM db 'Amount of available memory:        b$'
EX_MEM db 'Size of extended memory:       Kb$'
MCB_TABLE db 'MCB table:$'
MCB_STRING db 'MCB type:   , MCB adress:     , PSP adress:     ,
Size:       , SC/CD: $'
ENDL db 0DH, 0AH, '$'

; 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 шест. числа в AX
    push CX
    mov AH,AL
    call TETR_TO_HEX
    xchg AL,AH
    mov CL,4
    shr AL,CL
    call TETR_TO_HEX ;в AL старшая цифра
    pop CX ;в AH младшая
    ret
BYTE_TO_HEX ENDP

WRD_TO_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_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
    mov ah, 09h
```

```
    int 21h
    pop ax
    ret
PRINT_STRING ENDP

PRINT_SYMB PROC near
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
PRINT_SYMB ENDP

AVMEM PROC near
    mov ah, 4Ah
    mov bx, 0FFFFh
    int 21h

    mov ax, bx
    mov cx, 10h
    mul cx
    mov si, offset AV_MEM+33
    call WRD_TO_DEC
    mov dx, offset AV_MEM
    call PRINT_STRING
    mov dx, offset ENDL
    call PRINT_STRING
    ret
AVMEM ENDP

EXMEM PROC near
    mov    al, 30h
    out    70h, al
    in     al, 71h
    mov    bl, al
    mov    al, 31h
    out    70h, al
    in     al, 71h
    mov    ah, al
    mov    al, bl
    mov    si, offset EX_MEM+29
    xor dx, dx
    call WRD_TO_DEC
    mov    dx, offset EX_MEM
    call PRINT_STRING
    mov    dx, offset ENDL
    call PRINT_STRING
    ret
EXMEM ENDP

MCB PROC near
    mov    dx, offset MCB_TABLE
    call PRINT_STRING
    mov    dx, offset endl
    call PRINT_STRING
```

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

circle:
    ;type
    mov al, es:[0000h]
    call BYTE_TO_HEX
    mov    di, offset MCB_STRING+10
    mov [di], ax

    ;adress
    mov di, offset MCB_STRING+29
    mov ax, es
    call WRD_TO_HEX

    ;PSP adress
    mov ax, es:[0001h]
    mov di, offset MCB_STRING+47
    call WRD_TO_HEX

    ;size
    mov ax, es:[0003h]
    mov cx, 10h
    mul cx
    mov    si, offset MCB_STRING+61
    call WRD_TO_DEC

    mov dx, offset MCB_STRING
    call PRINT_STRING

    ;SC_CD
    mov bx, 8
    mov cx, 7
l_loop:
    mov dl, es:[bx]
    call PRINT_SYMB
    inc bx
    loop l_loop

    mov al, es:[0000h]
    cmp al, 5ah
    je final

    mov ax, es
    add ax, es:[0003h]
    inc ax
    mov es, ax
    mov dx, offset ENDL
    call PRINT_STRING
    jmp circle

final:
     ret
```

```
MCB ENDP

BEGIN:
    call AVMEM
    call EXMEM
    call MCB
    xor AL, AL
    mov AH, 4Ch
    int 21h
TESTPC ENDS
END START
```

Название файла: lab3_2.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN

; Данные
AV_MEM db 'Amount of available memory:        b$'
EX_MEM db 'Size of extended memory:       Kb$'
MCB_TABLE db 'MCB table:$'
MCB_STRING db 'MCB type:   , MCB adress:     , PSP adress:     ,
Size:       , SC/CD: $'
ENDL db 0DH, 0AH, '$'

; 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 шест. числа в AX
    push CX
    mov AH,AL
    call TETR_TO_HEX
    xchg AL,AH
    mov CL,4
    shr AL,CL
    call TETR_TO_HEX ;в AL старшая цифра
    pop CX ;в AH младшая
    ret
BYTE_TO_HEX ENDP

WRD_TO_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_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
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT_STRING ENDP

PRINT_SYMB PROC near
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
PRINT_SYMB ENDP

AVMEM PROC near
    mov ah, 4Ah
    mov bx, 0FFFFh
    int 21h

    mov ax, bx
    mov cx, 10h
    mul cx
    mov si, offset AV_MEM+33
    call WRD_TO_DEC
    mov dx, offset AV_MEM
    call PRINT_STRING
    mov dx, offset ENDL
    call PRINT_STRING
    ret
AVMEM ENDP

EXMEM PROC near
    mov    al, 30h
    out    70h, al
    in     al, 71h
    mov    bl, al
    mov    al, 31h
    out    70h, al
    in     al, 71h
    mov    ah, al
    mov    al, bl
    mov    si, offset EX_MEM+29
    xor dx, dx
    call WRD_TO_DEC
    mov    dx, offset EX_MEM
    call PRINT_STRING
    mov    dx, offset ENDL
    call PRINT_STRING
    ret
```

```
EXMEM ENDP

MCB PROC near
    mov    dx, offset MCB_TABLE
    call PRINT_STRING
    mov    dx, offset endl
    call PRINT_STRING

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

circle:
    ;type
    mov al, es:[0000h]
    call BYTE_TO_HEX
    mov    di, offset MCB_STRING+10
    mov [di], ax

    ;adress
    mov di, offset MCB_STRING+29
    mov ax, es
    call WRD_TO_HEX

    ;PSP adress
    mov ax, es:[0001h]
    mov di, offset MCB_STRING+47
    call WRD_TO_HEX

    ;size
    mov ax, es:[0003h]
    mov cx, 10h
    mul cx
    mov    si, offset MCB_STRING+61
    call WRD_TO_DEC

    mov dx, offset MCB_STRING
    call PRINT_STRING

    ;SC_CD
    mov bx, 8
    mov cx, 7
l_loop:
    mov dl, es:[bx]
    call PRINT_SYMB
    inc bx
    loop l_loop

    mov al, es:[0000h]
    cmp al, 5ah
    je final

    mov ax, es
    add ax, es:[0003h]
    inc ax
```

```
        mov es, ax
        mov dx, offset ENDL
        call PRINT_STRING
        jmp circle

    final:
         ret

    MCB ENDP

    FREE_MEM PROC near
        push ax
        push bx
        push dx

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

        pop dx
        pop bx
        pop ax
        ret
    FREE_MEM ENDP

    BEGIN:
        call AVMEM
        call EXMEM
        call FREE_MEM
        call MCB
        xor al, al
        mov ah, 4Ch
        int 21h
    prog_end:
    TESTPC ENDS
    END START
```

Название файла: lab3_3.asm

```
    TESTPC SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG 100H
    START: JMP BEGIN

    ; Данные
    AV_MEM db 'Amount of available memory:        b$'
    EX_MEM db 'Size of extended memory:       Kb$'
    MCB_TABLE db 'MCB table:$'
    MCB_STRING db 'MCB type:   , MCB adress:     , PSP adress:     ,
Size:       , SC/CD:   $'
```

```
ENDL db 0DH, 0AH, '$'

; 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 шест. числа в AX
    push CX
    mov AH,AL
    call TETR_TO_HEX
    xchg AL,AH
    mov CL,4
    shr AL,CL
    call TETR_TO_HEX ;в AL старшая цифра
    pop CX ;в AH младшая
    ret
BYTE_TO_HEX ENDP

WRD_TO_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_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
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT_STRING ENDP

PRINT_SYMB PROC near
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
PRINT_SYMB ENDP

AVMEM PROC near
    mov ah, 4Ah
    mov bx, 0FFFFh
    int 21h
```

```
    mov ax, bx
    mov cx, 10h
    mul cx
    mov si, offset AV_MEM+33
    call WRD_TO_DEC
    mov dx, offset AV_MEM
    call PRINT_STRING
    mov dx, offset ENDL
    call PRINT_STRING
    ret
AVMEM ENDP

EXMEM PROC near
    mov    al, 30h
    out    70h, al
    in     al, 71h
    mov    bl, al
    mov    al, 31h
    out    70h, al
    in     al, 71h
    mov    ah, al
    mov    al, bl
    mov    si, offset EX_MEM+29
    xor dx, dx
    call WRD_TO_DEC
    mov    dx, offset EX_MEM
    call PRINT_STRING
    mov    dx, offset ENDL
    call PRINT_STRING
    ret
EXMEM ENDP

MCB PROC near
    mov    dx, offset MCB_TABLE
    call PRINT_STRING
    mov    dx, offset endl
    call PRINT_STRING

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

circle:
    ;type
    mov al, es:[0000h]
    call BYTE_TO_HEX
    mov    di, offset MCB_STRING+10
    mov [di], ax

    ;adress
    mov di, offset MCB_STRING+29
    mov ax, es
    call WRD_TO_HEX
```

```
    ;PSP adress
    mov ax, es:[0001h]
    mov di, offset MCB_STRING+47
    call WRD_TO_HEX

    ;size
    mov ax, es:[0003h]
    mov cx, 10h
    mul cx
    mov    si, offset MCB_STRING+61
    call WRD_TO_DEC

    mov dx, offset MCB_STRING
    call PRINT_STRING

    ;SC_CD
    ;SC_CD
    mov bx, 8
    mov cx, 7
l_loop:
    mov dl, es:[bx]
    call PRINT_SYMB
    inc bx
    loop l_loop

    mov al, es:[0000h]
    cmp al, 5ah
    je final

    mov ax, es
    add ax, es:[0003h]
    inc ax
    mov es, ax
    mov dx, offset ENDL
    call PRINT_STRING
    jmp circle

final:
     ret

MCB ENDP

FREE_MEM PROC near
    push ax
    push bx
    push dx
    lea ax, prog_end
    mov bx, 10h
    xor dx, dx
    div bx
    inc ax
    mov bx, ax
    mov al, 0
    mov ah, 4Ah
    int 21h
    pop dx
```

```
        pop bx
        pop ax
        ret
    FREE_MEM ENDP

    REQEST_MEM PROC near
        push ax
        push bx
        push dx
        mov bx, 1000h
        mov ah, 48h
        int 21h
        pop dx
        pop bx
        pop ax
        ret
    REQEST_MEM ENDP

    BEGIN:
        call AVMEM
        call EXMEM
        call FREE_MEM
        call REQEST_MEM
        call MCB
        xor al, al
        mov ah, 4Ch
        int 21h
    prog_end:
    TESTPC ENDS
    END START
```

Название файла: lab3_4.asm

```
    TESTPC SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG 100H
    START: JMP BEGIN

    ; Данные
    AV_MEM db 'Amount of available memory:        b$'
    EX_MEM db 'Size of extended memory:       Kb$'
    MCB_TABLE db 'MCB table:$'
    MCB_STRING db 'MCB type:   , MCB adress:     , PSP adress:     ,
Size:       , SC/CD:   $'
    ENDL db 0DH, 0AH, '$'
    REQ_ERROR db 'Memory can not be allocated.$'


    ; 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 шест. числа в AX
    push CX
    mov AH,AL
    call TETR_TO_HEX
    xchg AL,AH
    mov CL,4
    shr AL,CL
    call TETR_TO_HEX ;в AL старшая цифра
    pop CX ;в AH младшая
    ret
BYTE_TO_HEX ENDP

WRD_TO_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_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
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT_STRING ENDP

PRINT_SYMB PROC near
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
PRINT_SYMB ENDP

AVMEM PROC near
    mov ah, 4Ah
    mov bx, 0FFFFh
    int 21h

    mov ax, bx
    mov cx, 10h
    mul cx
    mov si, offset AV_MEM+33
    call WRD_TO_DEC
    mov dx, offset AV_MEM
    call PRINT_STRING
    mov dx, offset ENDL
    call PRINT_STRING
```

```
    ret
AVMEM ENDP

EXMEM PROC near
    mov    al, 30h
    out    70h, al
    in     al, 71h
    mov    bl, al
    mov    al, 31h
    out    70h, al
    in     al, 71h
    mov    ah, al
    mov    al, bl
    mov    si, offset EX_MEM+29
    xor dx, dx
    call WRD_TO_DEC
    mov    dx, offset EX_MEM
    call PRINT_STRING
    mov    dx, offset ENDL
    call PRINT_STRING
    ret
EXMEM ENDP

MCB PROC near
    mov    dx, offset MCB_TABLE
    call PRINT_STRING
    mov    dx, offset endl
    call PRINT_STRING

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

circle:
    ;type
    mov al, es:[0000h]
    call BYTE_TO_HEX
    mov    di, offset MCB_STRING+10
    mov [di], ax

    ;adress
    mov di, offset MCB_STRING+29
    mov ax, es
    call WRD_TO_HEX

    ;PSP adress
    mov ax, es:[0001h]
    mov di, offset MCB_STRING+47
    call WRD_TO_HEX

    ;size
    mov ax, es:[0003h]
    mov cx, 10h
    mul cx
    mov    si, offset MCB_STRING+61
```

```
    call WRD_TO_DEC

    mov dx, offset MCB_STRING
    call PRINT_STRING

    ;SC_CD
    mov bx, 8
    mov cx, 7
l_loop:
    mov dl, es:[bx]
    call PRINT_SYMB
    inc bx
    loop l_loop

    mov al, es:[0000h]
    cmp al, 5ah
    je final

    mov ax, es
    add ax, es:[0003h]
    inc ax
    mov es, ax
    mov dx, offset ENDL
    call PRINT_STRING
    jmp circle


final:
     ret

MCB ENDP

FREE_MEM PROC near
    push ax
    push bx
    push dx
    lea ax, prog_end
    mov bx, 10h
    xor dx, dx
    div bx
    inc ax
    mov bx, ax
    mov al, 0
    mov ah, 4Ah
    int 21h
    pop dx
    pop bx
    pop ax
    ret
FREE_MEM ENDP

REQUEST_MEM PROC near
    push ax
    push bx
    push dx
    mov bx, 1000h
```

```
    mov ah, 48h
    int 21h
    jnc request_end

    mov dx, offset REQ_ERROR
    call PRINT_STRING
    mov    dx, offset endl
    call PRINT_STRING

request_end:
    pop dx
    pop bx
    pop ax
    ret
REQUEST_MEM ENDP

BEGIN:
    call AVMEM
    call EXMEM
    call REQUEST_MEM
    call FREE_MEM
    call MCB
    xor al, al
    mov ah, 4Ch
    int 21h
prog_end:
TESTPC ENDS
END START
```