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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Операционные системы»

Тема: Исследование организации управления основной памятью

Студент гр. 9383 _____ Арутюнян С.Н.

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.

Выполнение работы

Шаг 1. Был написан модуль типа **.COM**, который выбирает и распечатывает следующую информацию:

1. Количетсов доступной памяти.
2. Размер расширенной памяти.
3. Выводит цепочку блоков управления памятью.

Адреса при выводе представляются шестнадцатеричными числами. Объем памяти выводится в виде десятичных чисел. Последние 8 байт MCB выводятся как символы. Текст программы приведен в приложении А.

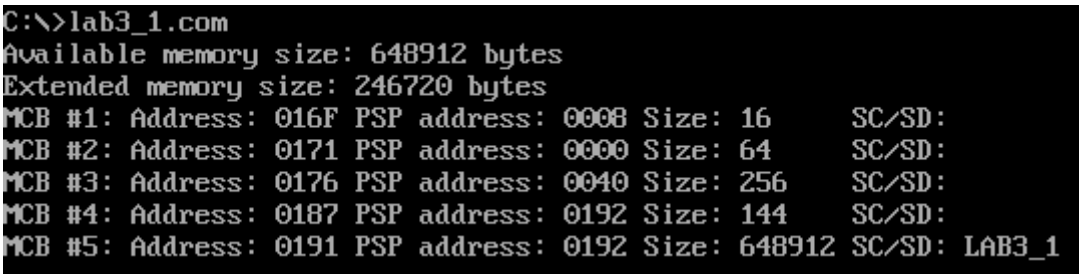

Рис. 1. Вывод программы №1

Шаг 2. Программа была изменена таким образом, чтобы она освобождала память, которую она не занимает. Текст программы приведен в приложении Б.

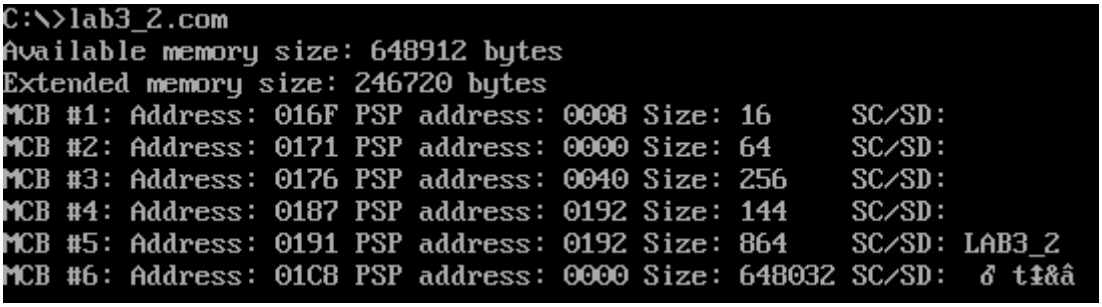

Рис. 2. Вывод программы №2

Шаг 3. Программа была изменена таким образом, чтобы после освобождения памяти программа запрашивала 64Кб памяти функцией 48h прерывания 21h. Текст программы приведен в приложении В.

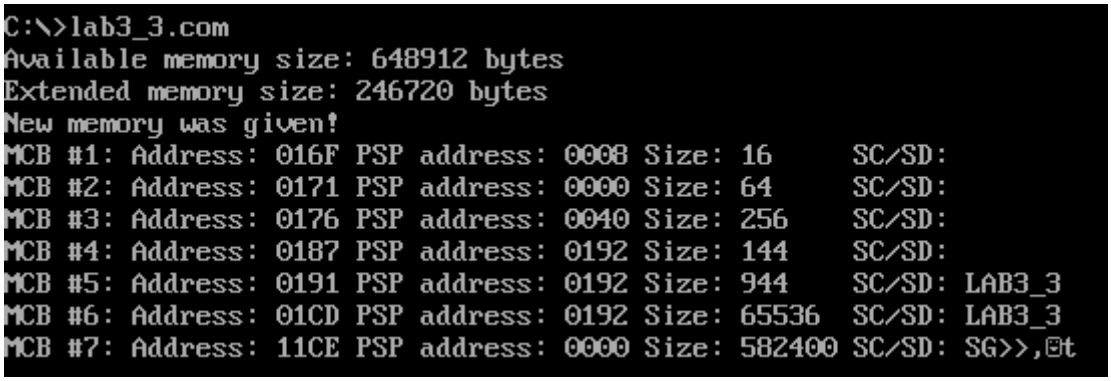

Рис. 3. Вывод программы №3

Шаг 4. Программа была изменена таким образом, чтобы она запрашивала 64Кб памяти функцией 48h прерывания 21h до освобождения памяти. Текст программы приведен в приложении Г.

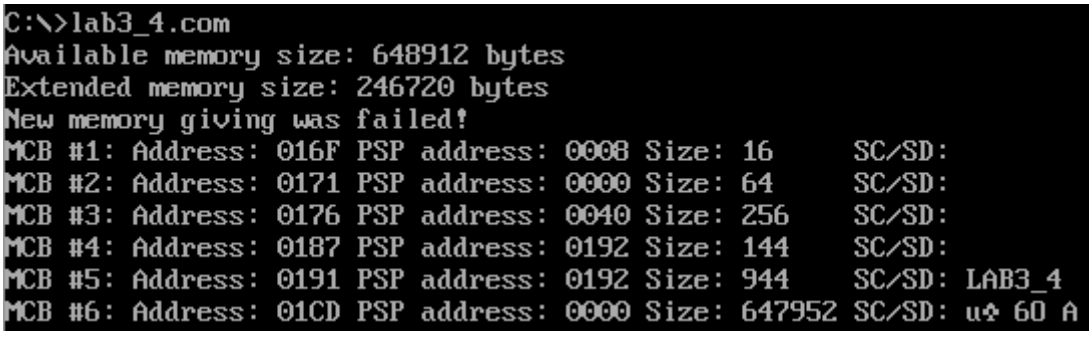

Рис. 4. Вывод программы №4

Контрольные вопросы

1. Что означает «доступный объем памяти»?

Доступный объем памяти — это область оперативной памяти, которая отдается программе для использования.

2. Где MCB блок Вашей программы в списке?

На первом рисунке MCB блок моей программы — пятый в спике.

На втором рисунке MCB блок моей программы — пятый в списке. Шестой MCB блок — блок с высвобожденной неиспользуемой памятью.

На третьем рисунке MCB блок моей программы — пятый и шестой в списке. Далее мы выделяем 64Кб памяти и делим выделенный блок на две части — шестой и седьмой MCB блок.

На четвертом рисунке MCB блок моей программы — пятый в списке.

3. Какой размер памяти занимает программа в каждом случае?

В первом случае программа занимает всю доступную память.

Во втором случае программа занимает только необходимый размер программы — 864 байт.

В третьем случае программа занимает необходимый размер программы + выделенный блок размером 64Кб, то есть 944+65536 = 66480 байт.

В четвертом случае программа занимает только необходимый вопрос памяти — 944 байт, т. к. мы выделили 64Кб памяти и сразу после этого освободили неиспользуемую память.

Заключение

В процессе выполнения лабораторной работы был исследован механизм управления памятью в операционной системе DOS.

Приложение А

```
TESTPC  SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
org 100h

start: jmp begin

; DATA
SC_OR_SD_MSG db "SC/SD: ", '$'
RECORD_SIZE db "Size:        ", '$'
ADDRESS db "Address:      ", '$'
PSP_ADDRESS db "PSP address:      ", '$'
MCB_NUMBER db "MCB #", '$'
DECIMAL_NUMBER db "    ", '$'
EXTENDED_MEMORY_SIZE db "Extended memory size:        bytes", 0dh, 0ah, '$'
AVAILABLE_MEM_SIZE db "Available memory size:        bytes", 0dh, 0ah, '$'
NEWLINE db 0dh, 0ah, '$'



;-------------------------------------------------
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 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 цифра
        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


PARAGRAPHS_TO_BYTES PROC

    push ax
    push bx
    push cx
    push dx
    push si

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

division:
          div bx
          push dx
          inc cx
          xor dx, dx
          cmp ax, 0h
          jnz division
```

```
write_symbol:
        pop dx
        or dl, 30h
        mov [si], dl
        inc si
        loop write_symbol

    pop si
    pop dx
    pop cx
    pop bx
    pop ax
        ret

PARAGRAPHS_TO_BYTES ENDP


PRINT_NEWLINE proc near

    push ax
    push dx

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

    pop dx
    pop ax

    ret

PRINT_NEWLINE endp

; Печатает строку в dx
```

```
WRITE_STRING proc near

    push ax
    mov ah, 9h
    int 21h
    pop ax

    ret

WRITE_STRING endp



PRINT_AVAILABLE_MEM_SIZE proc near

    push ax
    push bx
    push si

    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, bx
    mov si, offset AVAILABLE_MEM_SIZE
    add si, 23
    call PARAGRAPHS_TO_BYTES

    mov dx, offset AVAILABLE_MEM_SIZE
    call WRITE_STRING

    pop si
    pop bx
    pop ax

    ret
```

```
PRINT_AVAILABLE_MEM_SIZE endp


PRINT_EXTENDED_MEM_SIZE proc near

    push ax
    push bx
    push si

    mov al, 30h
    out 70h, al
    in al, 71h

    mov al, 31h
    out 70h, al
    in al, 71h
    mov ah, al

    ; теперь в ax размер расширенной памяти
    mov si, offset EXTENDED_MEMORY_SIZE
    add si, 22
    call PARAGRAPHS_TO_BYTES

    mov dx, offset EXTENDED_MEMORY_SIZE
    call WRITE_STRING

    pop si
    pop bx
    pop ax

    ret

PRINT_EXTENDED_MEM_SIZE endp
```

```
PRINT_MCB_RECORD proc near

    push ax
    push dx
    push si
    push di
    push cx

    ; адрес записи
    mov ax, es
    mov di, offset ADDRESS
    add di, 12
    call WRD_TO_HEX
    mov dx, offset ADDRESS
    call WRITE_STRING

    ; адрес PSP
    mov ax, es:[1]
    mov di, offset PSP_ADDRESS
    add di, 16
    call WRD_TO_HEX
    mov dx, offset PSP_ADDRESS
    call WRITE_STRING

    ; размер записи
    mov ax, es:[3]
    mov si, offset RECORD_SIZE
    add si, 6
    call PARAGRAPHS_TO_BYTES
    mov dx, offset RECORD_SIZE
    call WRITE_STRING

    ; SC или SD
    mov bx, 8
```

```
    mov dx, offset SC_OR_SD_MSG
    call WRITE_STRING
    mov cx, 7

    print_scsd_loop:
        mov dl, es:[bx]
        mov ah, 02h
        int 21h
        inc bx
        loop print_scsd_loop



    pop cx
    pop di
    pop si
    pop dx
    pop ax

    ret

PRINT_MCB_RECORD endp



OFFSET_DECIMAL_NUMBER proc near

    offset_loop:
        cmp byte ptr [si], ' '
        jne exit_offset_decimal
        inc si
        jmp offset_loop

exit_offset_decimal:
    ret

OFFSET_DECIMAL_NUMBER endp
```

```
PRINT_MCB_TABLE proc near

    push ax
    push bx
    push es
    push dx

    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    ; вернуть es обратно?
    mov es, ax
    mov cl, 1

    print_mcb_info:
        ; вывод "MCB #"
        mov dx, offset MCB_NUMBER
        call WRITE_STRING

        ; вывод порядкового номера
        mov al, cl
        mov si, offset DECIMAL_NUMBER
        add si, 2
        call BYTE_TO_DEC
        call OFFSET_DECIMAL_NUMBER
        mov dx, si
        call WRITE_STRING

        ; вывод ': '
        mov dl, ':'
        mov ah, 02h
        int 21h
        mov dl, ' '
```

```
        mov ah, 02h
        int 21h


        call PRINT_MCB_RECORD
        call PRINT_NEWLINE

        mov al, es:[0]
        cmp al, 5ah
        je exit

        ; берем размер MCB записи
        mov bx, es:[3]
        ; и сдвигаем адрес на начало записи к началу следующей
        mov ax, es
        add ax, bx
        inc ax
        mov es, ax

        inc cl
        jmp print_mcb_info

exit:
    pop dx
    pop es
    pop bx
    pop ax

    ret

PRINT_MCB_TABLE endp



begin:
```

```
    ; Вывести количество доступной памяти, размер расш. памяти, цепочку блоков управления памятью
    call PRINT_AVAILABLE_MEM_SIZE
    call PRINT_EXTENDED_MEM_SIZE
    call PRINT_MCB_TABLE

    ; выход в DOS
    xor al, al
    mov ah, 4ch
    int 21h

TESTPC  ENDS
        END start
```

Приложение Б

```
TESTPC  SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
org 100h

start: jmp begin

; DATA
SC_OR_SD_MSG db "SC/SD: ", '$'
RECORD_SIZE db "Size:        ", '$'
ADDRESS db "Address:      ", '$'
PSP_ADDRESS db "PSP address:      ", '$'
MCB_NUMBER db "MCB #", '$'
DECIMAL_NUMBER db "    ", '$'
EXTENDED_MEMORY_SIZE db "Extended memory size:        bytes", 0dh, 0ah, '$'
AVAILABLE_MEM_SIZE db "Available memory size:        bytes", 0dh, 0ah, '$'
NEWLINE db 0dh, 0ah, '$'


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

```
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 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 цифра
        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


PARAGRAPHS_TO_BYTES PROC

    push ax
    push bx
    push cx
```

```
    push dx
    push si

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

division:
        div bx
        push dx
        inc cx
        xor dx, dx
        cmp ax, 0h
        jnz division

write_symbol:
        pop dx
        or dl, 30h
        mov [si], dl
        inc si
        loop write_symbol

    pop si
    pop dx
    pop cx
    pop bx
    pop ax
        ret

PARAGRAPHS_TO_BYTES ENDP


PRINT_NEWLINE proc near
```

```
    push ax
    push dx

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

    pop dx
    pop ax

    ret

PRINT_NEWLINE endp

; Печатает строку в dx
WRITE_STRING proc near

    push ax
    mov ah, 9h
    int 21h
    pop ax

    ret

WRITE_STRING endp


PRINT_AVAILABLE_MEM_SIZE proc near

    push ax
    push bx
    push si

    mov ah, 4ah
    mov bx, 0ffffh
```

```
    int 21h

    mov ax, bx
    mov si, offset AVAILABLE_MEM_SIZE
    add si, 23
    call PARAGRAPHS_TO_BYTES

    mov dx, offset AVAILABLE_MEM_SIZE
    call WRITE_STRING

    pop si
    pop bx
    pop ax

    ret

PRINT_AVAILABLE_MEM_SIZE endp


PRINT_EXTENDED_MEM_SIZE proc near

    push ax
    push bx
    push si

    mov al, 30h
    out 70h, al
    in al, 71h

    mov al, 31h
    out 70h, al
    in al, 71h
    mov ah, al

    ; теперь в ax размер расширенной памяти
```

```
    mov si, offset EXTENDED_MEMORY_SIZE
    add si, 22
    call PARAGRAPHS_TO_BYTES

    mov dx, offset EXTENDED_MEMORY_SIZE
    call WRITE_STRING

    pop si
    pop bx
    pop ax

    ret

PRINT_EXTENDED_MEM_SIZE endp


PRINT_MCB_RECORD proc near

    push ax
    push dx
    push si
    push di
    push cx

    ; адрес записи
    mov ax, es
    mov di, offset ADDRESS
    add di, 12
    call WRD_TO_HEX
    mov dx, offset ADDRESS
    call WRITE_STRING

    ; адрес PSP
    mov ax, es:[1]
    mov di, offset PSP_ADDRESS
```

```
add di, 16
call WRD_TO_HEX
mov dx, offset PSP_ADDRESS
call WRITE_STRING

; размер записи
mov ax, es:[3]
mov si, offset RECORD_SIZE
add si, 6
call PARAGRAPHS_TO_BYTES
mov dx, offset RECORD_SIZE
call WRITE_STRING

; SC или SD
mov bx, 8
mov dx, offset SC_OR_SD_MSG
call WRITE_STRING
mov cx, 7

print_scsd_loop:
    mov dl, es:[bx]
    mov ah, 02h
    int 21h
    inc bx
    loop print_scsd_loop


pop cx
pop di
pop si
pop dx
pop ax

ret
```

```
PRINT_MCB_RECORD endp


OFFSET_DECIMAL_NUMBER proc near

    offset_loop:
        cmp byte ptr [si], ' '
        jne exit_offset_decimal
        inc si
        jmp offset_loop

exit_offset_decimal:
    ret

OFFSET_DECIMAL_NUMBER endp


PRINT_MCB_TABLE proc near

    push ax
    push bx
    push es
    push dx

    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    ; вернуть es обратно?
    mov es, ax
    mov cl, 1

    print_mcb_info:
        ; вывод "MCB #"
        mov dx, offset MCB_NUMBER
        call WRITE_STRING
```

```
; вывод порядкового номера
mov al, cl
mov si, offset DECIMAL_NUMBER
add si, 2
call BYTE_TO_DEC
call OFFSET_DECIMAL_NUMBER
mov dx, si
call WRITE_STRING

; вывод ': '
mov dl, ':'
mov ah, 02h
int 21h
mov dl, ' '
mov ah, 02h
int 21h



call PRINT_MCB_RECORD
call PRINT_NEWLINE

mov al, es:[0]
cmp al, 5ah
je exit

; берем размер MCB записи
mov bx, es:[3]
; и сдвигаем адрес на начало записи к началу следующей
mov ax, es
add ax, bx
inc ax
mov es, ax

inc cl
```

```
        jmp print_mcb_info

exit:
    pop dx
    pop es
    pop bx
    pop ax

    ret

PRINT_MCB_TABLE endp



FREE_UNUSED_MEMORY proc near

    push ax
    push bx
    push cx
    push dx

    lea ax, global_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 cx
    pop bx
    pop ax
```

```
    ret

FREE_UNUSED_MEMORY endp



begin:
    ; Вывести количество доступной памяти, размер расш. памяти, цепочку блоков управления памятью
    call PRINT_AVAILABLE_MEM_SIZE
    call PRINT_EXTENDED_MEM_SIZE

    call FREE_UNUSED_MEMORY
    call PRINT_MCB_TABLE

    ; выход в DOS
    xor al, al
    mov ah, 4ch
    int 21h

global_end:
TESTPC  ENDS
        END start
```

Приложение В

```
TESTPC  SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
org 100h

start: jmp begin

; DATA
SC_OR_SD_MSG db "SC/SD: ", '$'
RECORD_SIZE db "Size:        ", '$'
ADDRESS db "Address:      ", '$'
```

```
PSP_ADDRESS db "PSP address:      ", '$'
MCB_NUMBER db "MCB #", '$'
DECIMAL_NUMBER db "   ", '$'
EXTENDED_MEMORY_SIZE db "Extended memory size:        bytes", 0dh, 0ah, '$'
AVAILABLE_MEM_SIZE db "Available memory size:        bytes", 0dh, 0ah, '$'
MEMORY_SUCCESS_MSG db "New memory was given!", 0dh, 0ah, '$'
MEMORY_FAIL_MSG db "New memory giving was failed!", '$'
NEWLINE db 0dh, 0ah, '$'


;-------------------------------------------------
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 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 цифра
        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


PARAGRAPHS_TO_BYTES PROC

   push ax
   push bx
   push cx
   push dx
   push si

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

division:
        div bx
        push dx
        inc cx
        xor dx, dx
        cmp ax, 0h
        jnz division

write_symbol:
        pop dx
        or dl, 30h
        mov [si], dl
        inc si
        loop write_symbol

   pop si
```

```
    pop dx
    pop cx
    pop bx
    pop ax
        ret

PARAGRAPHS_TO_BYTES ENDP


PRINT_NEWLINE proc near

    push ax
    push dx

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

    pop dx
    pop ax

    ret

PRINT_NEWLINE endp

; Печатает строку в dx
WRITE_STRING proc near

    push ax
    mov ah, 9h
    int 21h
    pop ax

    ret
```

```
WRITE_STRING endp


PRINT_AVAILABLE_MEM_SIZE proc near

    push ax
    push bx
    push si

    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, bx
    mov si, offset AVAILABLE_MEM_SIZE
    add si, 23
    call PARAGRAPHS_TO_BYTES

    mov dx, offset AVAILABLE_MEM_SIZE
    call WRITE_STRING

    pop si
    pop bx
    pop ax

    ret

PRINT_AVAILABLE_MEM_SIZE endp


PRINT_EXTENDED_MEM_SIZE proc near

    push ax
    push bx
    push si
```

```
    mov al, 30h
    out 70h, al
    in al, 71h

    mov al, 31h
    out 70h, al
    in al, 71h
    mov ah, al

    ; теперь в ax размер расширенной памяти
    mov si, offset EXTENDED_MEMORY_SIZE
    add si, 22
    call PARAGRAPHS_TO_BYTES

    mov dx, offset EXTENDED_MEMORY_SIZE
    call WRITE_STRING

    pop si
    pop bx
    pop ax

    ret

PRINT_EXTENDED_MEM_SIZE endp


PRINT_MCB_RECORD proc near

    push ax
    push dx
    push si
    push di
    push cx
```

```
; адрес записи
mov ax, es
mov di, offset ADDRESS
add di, 12
call WRD_TO_HEX
mov dx, offset ADDRESS
call WRITE_STRING

; адрес PSP
mov ax, es:[1]
mov di, offset PSP_ADDRESS
add di, 16
call WRD_TO_HEX
mov dx, offset PSP_ADDRESS
call WRITE_STRING

; размер записи
mov ax, es:[3]
mov si, offset RECORD_SIZE
add si, 6
call PARAGRAPHS_TO_BYTES
mov dx, offset RECORD_SIZE
call WRITE_STRING

; SC или SD
mov bx, 8
mov dx, offset SC_OR_SD_MSG
call WRITE_STRING
mov cx, 7

print_scsd_loop:
    mov dl, es:[bx]
    mov ah, 02h
    int 21h
    inc bx
```

```
        loop print_scsd_loop


    pop cx
    pop di
    pop si
    pop dx
    pop ax

    ret

PRINT_MCB_RECORD endp


OFFSET_DECIMAL_NUMBER proc near

    offset_loop:
        cmp byte ptr [si], ' '
        jne exit_offset_decimal
        inc si
        jmp offset_loop

exit_offset_decimal:
    ret

OFFSET_DECIMAL_NUMBER endp


PRINT_MCB_TABLE proc near

    push ax
    push bx
    push es
    push dx
```

```
mov ah, 52h
int 21h
mov ax, es:[bx-2]
; вернуть es обратно?
mov es, ax
mov cl, 1

print_mcb_info:
    ; вывод "MCB #"
    mov dx, offset MCB_NUMBER
    call WRITE_STRING

    ; вывод порядкового номера
    mov al, cl
    mov si, offset DECIMAL_NUMBER
    add si, 2
    call BYTE_TO_DEC
    call OFFSET_DECIMAL_NUMBER
    mov dx, si
    call WRITE_STRING

    ; вывод ': '
    mov dl, ':'
    mov ah, 02h
    int 21h
    mov dl, ' '
    mov ah, 02h
    int 21h


    call PRINT_MCB_RECORD
    call PRINT_NEWLINE

    mov al, es:[0]
    cmp al, 5ah
```

```
        je exit

        ; берем размер MCB записи
        mov bx, es:[3]
        ; и сдвигаем адрес на начало записи к началу следующей
        mov ax, es
        add ax, bx
        inc ax
        mov es, ax

        inc cl
        jmp print_mcb_info

exit:
    pop dx
    pop es
    pop bx
    pop ax

    ret

PRINT_MCB_TABLE endp



FREE_UNUSED_MEMORY proc near

    push ax
    push bx
    push cx
    push dx

    lea ax, global_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 cx
    pop bx
    pop ax
    ret

FREE_UNUSED_MEMORY endp



ASK_FOR_MEMORY proc near

   push ax
   push bx
   push dx

   mov bx, 1000h
   mov ah, 48h
   int 21h

   ; Проверяем CF
   jc memory_failed
   jmp memory_success

memory_failed:
   mov dx, offset MEMORY_FAIL_MSG
   call WRITE_STRING
   jmp memory_ask_exit
```

```
memory_success:
  mov dx, offset MEMORY_SUCCESS_MSG
  call WRITE_STRING

memory_ask_exit:
  pop dx
  pop bx
  pop ax
  ret

ASK_FOR_MEMORY endp



begin:
   ; Вывести количество доступной памяти, размер расш. памяти, цепочку блоков управления памятью
   call PRINT_AVAILABLE_MEM_SIZE
   call PRINT_EXTENDED_MEM_SIZE

   call FREE_UNUSED_MEMORY
   call ASK_FOR_MEMORY

   call PRINT_MCB_TABLE

   ; выход в DOS
   xor al, al
   mov ah, 4ch
   int 21h

global_end:
TESTPC  ENDS
       END start
```

Приложение Г

```
TESTPC  SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
org 100h

start: jmp begin

; DATA
SC_OR_SD_MSG db "SC/SD: ", '$'
RECORD_SIZE db "Size:        ", '$'
ADDRESS db "Address:      ", '$'
PSP_ADDRESS db "PSP address:      ", '$'
MCB_NUMBER db "MCB #", '$'
DECIMAL_NUMBER db "    ", '$'
EXTENDED_MEMORY_SIZE db "Extended memory size:        bytes", 0dh, 0ah, '$'
AVAILABLE_MEM_SIZE db "Available memory size:        bytes", 0dh, 0ah, '$'
MEMORY_SUCCESS_MSG db "New memory was given!", 0dh, 0ah, '$'
MEMORY_FAIL_MSG db "New memory giving was failed!", '$'
NEWLINE db 0dh, 0ah, '$'


;-------------------------------------------------
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 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 цифра
        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


PARAGRAPHS_TO_BYTES PROC

    push ax
    push bx
    push cx
    push dx
    push si

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

division:
            div bx
            push dx
            inc cx
            xor dx, dx
```

```
        cmp ax, 0h
        jnz division

write_symbol:
        pop dx
        or dl, 30h
        mov [si], dl
        inc si
        loop write_symbol

    pop si
    pop dx
    pop cx
    pop bx
    pop ax
        ret

PARAGRAPHS_TO_BYTES ENDP


PRINT_NEWLINE proc near

    push ax
    push dx

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

    pop dx
    pop ax

    ret

PRINT_NEWLINE endp
```

```
; Печатает строку в dx
WRITE_STRING proc near

    push ax
    mov ah, 9h
    int 21h
    pop ax

    ret

WRITE_STRING endp


PRINT_AVAILABLE_MEM_SIZE proc near

    push ax
    push bx
    push si

    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, bx
    mov si, offset AVAILABLE_MEM_SIZE
    add si, 23
    call PARAGRAPHS_TO_BYTES

    mov dx, offset AVAILABLE_MEM_SIZE
    call WRITE_STRING

    pop si
    pop bx
    pop ax
```

```
    ret

PRINT_AVAILABLE_MEM_SIZE endp


PRINT_EXTENDED_MEM_SIZE proc near

    push ax
    push bx
    push si

    mov al, 30h
    out 70h, al
    in al, 71h

    mov al, 31h
    out 70h, al
    in al, 71h
    mov ah, al

    ; теперь в ax размер расширенной памяти
    mov si, offset EXTENDED_MEMORY_SIZE
    add si, 22
    call PARAGRAPHS_TO_BYTES

    mov dx, offset EXTENDED_MEMORY_SIZE
    call WRITE_STRING

    pop si
    pop bx
    pop ax

    ret
```

```
PRINT_EXTENDED_MEM_SIZE endp


PRINT_MCB_RECORD proc near

    push ax
    push dx
    push si
    push di
    push cx

    ; адрес записи
    mov ax, es
    mov di, offset ADDRESS
    add di, 12
    call WRD_TO_HEX
    mov dx, offset ADDRESS
    call WRITE_STRING

    ; адрес PSP
    mov ax, es:[1]
    mov di, offset PSP_ADDRESS
    add di, 16
    call WRD_TO_HEX
    mov dx, offset PSP_ADDRESS
    call WRITE_STRING

    ; размер записи
    mov ax, es:[3]
    mov si, offset RECORD_SIZE
    add si, 6
    call PARAGRAPHS_TO_BYTES
    mov dx, offset RECORD_SIZE
    call WRITE_STRING
```

```
    ; SC или SD
    mov bx, 8
    mov dx, offset SC_OR_SD_MSG
    call WRITE_STRING
    mov cx, 7

    print_scsd_loop:
        mov dl, es:[bx]
        mov ah, 02h
        int 21h
        inc bx
        loop print_scsd_loop



    pop cx
    pop di
    pop si
    pop dx
    pop ax

    ret

PRINT_MCB_RECORD endp



OFFSET_DECIMAL_NUMBER proc near

    offset_loop:
        cmp byte ptr [si], ' '
        jne exit_offset_decimal
        inc si
        jmp offset_loop

exit_offset_decimal:
    ret
```

```
OFFSET_DECIMAL_NUMBER endp


PRINT_MCB_TABLE proc near

    push ax
    push bx
    push es
    push dx

    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    ; вернуть es обратно?
    mov es, ax
    mov cl, 1

    print_mcb_info:
        ; вывод "MCB #"
        mov dx, offset MCB_NUMBER
        call WRITE_STRING

        ; вывод порядкового номера
        mov al, cl
        mov si, offset DECIMAL_NUMBER
        add si, 2
        call BYTE_TO_DEC
        call OFFSET_DECIMAL_NUMBER
        mov dx, si
        call WRITE_STRING

        ; вывод ': '
        mov dl, ':'
        mov ah, 02h
```

```
        int 21h
        mov dl, ' '
        mov ah, 02h
        int 21h


        call PRINT_MCB_RECORD
        call PRINT_NEWLINE

        mov al, es:[0]
        cmp al, 5ah
        je exit

        ; берем размер MCB записи
        mov bx, es:[3]
        ; и сдвигаем адрес на начало записи к началу следующей
        mov ax, es
        add ax, bx
        inc ax
        mov es, ax

        inc cl
        jmp print_mcb_info

exit:
    pop dx
    pop es
    pop bx
    pop ax

    ret

PRINT_MCB_TABLE endp
```

```
FREE_UNUSED_MEMORY proc near

    push ax
    push bx
    push cx
    push dx

    lea ax, global_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 cx
    pop bx
    pop ax
    ret

FREE_UNUSED_MEMORY endp



ASK_FOR_MEMORY proc near

   push ax
   push bx
   push dx

   mov bx, 1000h
```

```
  mov ah, 48h
  int 21h

  ; Проверяем CF
  jc memory_failed
  jmp memory_success

memory_failed:
  mov dx, offset MEMORY_FAIL_MSG
  call WRITE_STRING
  jmp memory_ask_exit

memory_success:
  mov dx, offset MEMORY_SUCCESS_MSG
  call WRITE_STRING

memory_ask_exit:
  pop dx
  pop bx
  pop ax
  ret

ASK_FOR_MEMORY endp



begin:
   ; Вывести количество доступной памяти, размер расш. памяти, цепочку блоков управления памятью
   call PRINT_AVAILABLE_MEM_SIZE
   call PRINT_EXTENDED_MEM_SIZE

   call ASK_FOR_MEMORY
   call FREE_UNUSED_MEMORY
```

```
    call PRINT_MCB_TABLE

    ; выход в DOS
    xor al, al
    mov ah, 4ch
    int 21h

global_end:
TESTPC  ENDS
        END start
```