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 _____ Орлов Д.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.

Выполнение работы

Шаг 1. Был написан 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 выводится в виде десятичных чисел. Последние 8 байт MCB выводятся как символы. Текст программы приведен в приложении А.

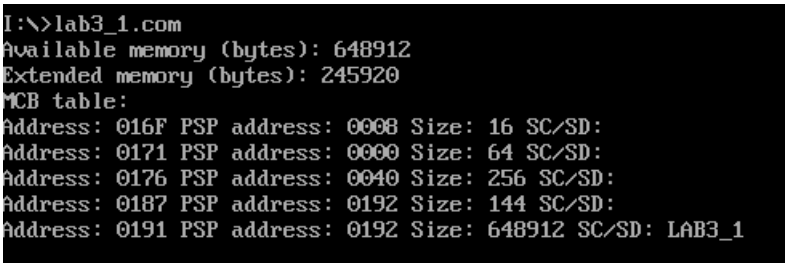

Рис. 1. Вывод программы №1

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

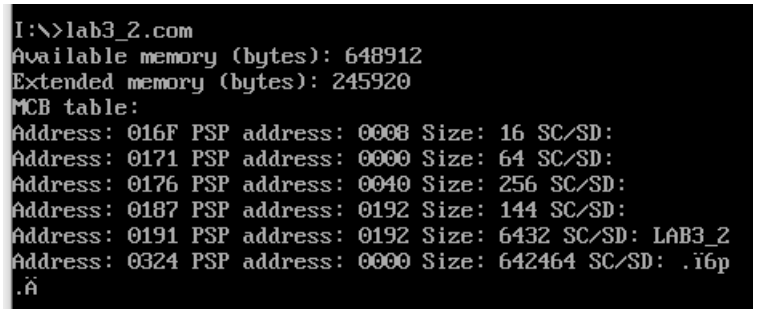

Рис. 2. Вывод программы №2

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.

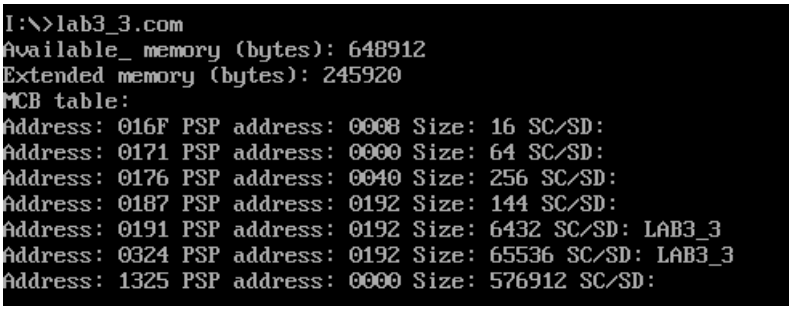

Рис. 3. Вывод программы №3

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.

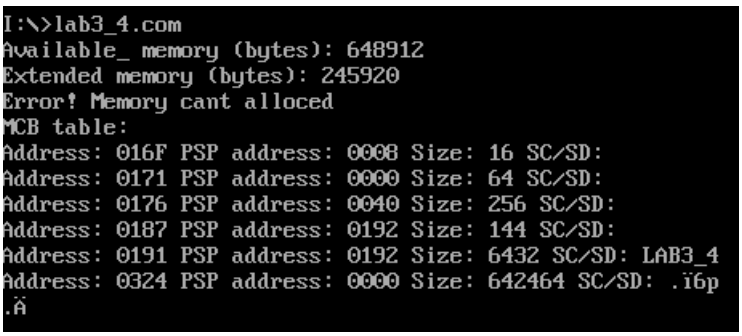

Рис. 4. Вывод программы №4

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?

MCB блок моей программы подписан названием исполняемого файла в столбце SC/SD: на первом рисунке – 5 в списке, на втором рисунке – 5 в списке, на третьем рисунке – 5 и 6 в списке, на четвертом рисунке – 5 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 — 6432 байт.

В третьем случае программа занимает необходимый размер программы + выделенный блок размером 64Кб, то есть 6432 + 65536 = 71968 байт.

В четвертом случае программа занимает только необходимый вопрос памяти — 6432 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

; Данные
AVAILABLE_MEMORY DB 'Available memory (bytes): ', '$'
EXTENDED_MEMORY DB 'Extended memory (bytes): ', '$'
MCB_TABLE DB 'MCB table: ', 0DH, 0AH, '$'
ADDRESS DB 'Address:     ', '$'
PSP DB 'PSP address:      ', '$'
STRING_SIZE DB 'Size: ', '$'
SC_SD DB 'SC/SD: ', '$'
LN DB 0DH,0AH,'$'
SPACE_STRING 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 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 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 10 с/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

;-----------------------------;

PRINT_STR PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_STR endp


PRINT_BYTE PROC
        mov bx, 10
      mov cx, 0
loop_1:
        div bx
        push dx
        inc cx
        mov dx, 0
```

```
        cmp ax, 0
        jne loop_1
print:
        pop dx
        add dl,30h
        mov ah,02h
        int 21h
        loop print
        ret
PRINT_BYTE endp


MEMORY_AVAILABLE PROC near
        mov dx, offset AVAILABLE_MEMORY
        call PRINT_STR

        mov ah, 4ah
        mov bx, 0ffffh
        int 21h
        mov ax, bx
        mov bx, 16
        mul bx
        call PRINT_BYTE

        mov dx, offset LN
        call PRINT_STR

        ret
MEMORY_AVAILABLE endp


MEMORY_EXTENDED proc near
        mov dx, offset EXTENDED_MEMORY
        call PRINT_STR
```

```
    mov al, 30h
    out 70h, al
        in al, 71h
        mov al, 31h
    out 70h, al
    in al, 71h

    mov ah, al
        mov bh, al
        mov ax, bx

        mov bx, 16
        mul bx

        call PRINT_BYTE

        mov dx, offset LN
        call PRINT_STR

        ret
MEMORY_EXTENDED endp



MCB PROC near
        mov ah, 52h
        int 21h
        mov ax, es:[bx-2]
        mov es, ax
        mov dx, offset MCB_TABLE
        call PRINT_STR

MCB_loop:
    mov ax, es              ;адрес
    mov di, offset ADDRESS
    add di, 12
```

```
    call WRD_TO_HEX
    mov dx, offset ADDRESS
    call PRINT_STR
        mov dx, offset SPACE_STRING
        call PRINT_STR

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

        mov dx, offset STRING_SIZE   ;размер
        call PRINT_STR
        mov ax, es:[3]
        mov di, offset STRING_SIZE
        add di, 6
        mov bx, 16
        mul bx
        call PRINT_BYTE
        mov dx, offset SPACE_STRING
        call PRINT_STR

        mov bx, 8               ;SC/SD
        mov dx, offset SC_SD
        call PRINT_STR
        mov cx, 7

loop_2:
        mov dl, es:[bx]
        mov ah, 02h
        int 21h
        inc bx
        loop loop_2
```

```
        mov dx, offset LN
        call PRINT_STR

        mov bx, es:[3h]
        mov al, es:[0h]
        cmp al, 5ah
        je FOR_END

        mov ax, es
        inc ax
        add ax, bx
        mov es, ax
        jmp MCB_loop

FOR_END:
        ret
MCB endp

BEGIN:
        call MEMORY_AVAILABLE
        call MEMORY_EXTENDED
        call MCB

        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

; Данные
AVAILABLE_MEMORY DB 'Available memory (bytes): ', '$'
EXTENDED_MEMORY DB 'Extended memory (bytes): ', '$'
MCB_TABLE DB 'MCB table: ', 0DH, 0AH, '$'
ADDRESS DB 'Address:     ', '$'
PSP DB 'PSP address:      ', '$'
STRING_SIZE DB 'Size: ', '$'
SC_SD DB 'SC/SD: ', '$'
LN DB 0DH,0AH,'$'
SPACE_STRING 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 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 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 10 с/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

;-----------------------------;

PRINT_STR PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_STR endp


PRINT_BYTE PROC
        mov bx, 10
      mov cx, 0
loop_1:
        div bx
        push dx
        inc cx
        mov dx, 0
        cmp ax, 0
        jne loop_1
print:
        pop dx
```

```
        add dl,30h
        mov ah,02h
        int 21h
        loop print
        ret
PRINT_BYTE endp


MEMORY_AVAILABLE PROC near
        mov dx, offset AVAILABLE_MEMORY
        call PRINT_STR

        mov ah, 4ah
        mov bx, 0ffffh
        int 21h
        mov ax, bx
        mov bx, 16
        mul bx
        call PRINT_BYTE

        mov dx, offset LN
        call PRINT_STR

        ret
MEMORY_AVAILABLE endp


MEMORY_EXTENDED proc near
        mov dx, offset EXTENDED_MEMORY
        call PRINT_STR

   mov al, 30h
   out 70h, al
        in al, 71h
        mov al, 31h
```

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

    mov ah, al
        mov bh, al
        mov ax, bx

        mov bx, 16
        mul bx

        call PRINT_BYTE

        mov dx, offset LN
        call PRINT_STR

        ret
MEMORY_EXTENDED endp



MCB PROC near
        mov ah, 52h
        int 21h
        mov ax, es:[bx-2]
        mov es, ax
        mov dx, offset MCB_TABLE
        call PRINT_STR

MCB_loop:
    mov ax, es              ;адрес
    mov di, offset ADDRESS
    add di, 12
    call WRD_TO_HEX
    mov dx, offset ADDRESS
    call PRINT_STR
        mov dx, offset SPACE_STRING
```

```
        call PRINT_STR

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

        mov dx, offset STRING_SIZE   ;размер
        call PRINT_STR
        mov ax, es:[3]
        mov di, offset STRING_SIZE
        add di, 6
        mov bx, 16
        mul bx
        call PRINT_BYTE
        mov dx, offset SPACE_STRING
        call PRINT_STR

        mov bx, 8               ;SC/SD
        mov dx, offset SC_SD
        call PRINT_STR
        mov cx, 7

loop_2:
        mov dl, es:[bx]
        mov ah, 02h
        int 21h
        inc bx
        loop loop_2

        mov dx, offset LN
        call PRINT_STR
```

```
        mov bx, es:[3h]
        mov al, es:[0h]
        cmp al, 5ah
        je FOR_END

        mov ax, es
        inc ax
        add ax, bx
        mov es, ax
        jmp MCB_loop

FOR_END:
        ret
MCB endp

TASK2 PROC near
   mov    ax, cs
   mov    es, ax
   mov    bx, offset END_CODE
   mov    ax, es
   mov    bx, ax
   mov    ah, 4ah
   int    21h
   ret
TASK2 endp

BEGIN:
        call MEMORY_AVAILABLE
        call MEMORY_EXTENDED
        call TASK2
        call MCB

        xor al, al
        mov ah, 4ch
        int 21h
```

```
END_CODE:
TESTPC ENDS

END START
```

Приложение В

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

; Данные
AVAILABLE_MEMORY DB 'Available_ memory (bytes): ', '$'
EXTENDED_MEMORY DB 'Extended memory (bytes): ', '$'
MCB_TABLE DB 'MCB table: ', 0DH, 0AH, '$'
ADDRESS DB 'Address:     ', '$'
PSP DB 'PSP address:      ', '$'
STRING_SIZE DB 'Size: ', '$'
SC_SD DB 'SC/SD: ', '$'
LN DB 0DH,0AH,'$'
SPACE_STRING DB ' ', '$'
MEMORY_ERROR DB 'Error! Memory cant alloced', 0DH, 0AH, '$'

; 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 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 10 с/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

;-----------------------------;

PRINT_STR PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_STR endp
```

```
PRINT_BYTE PROC
        mov bx, 10
      mov cx, 0
loop_1:
        div bx
        push dx
        inc cx
        mov dx, 0
        cmp ax, 0
        jne loop_1
print:
        pop dx
        add dl,30h
        mov ah,02h
        int 21h
        loop print
        ret
PRINT_BYTE endp


MEMORY_AVAILABLE PROC near
        mov dx, offset AVAILABLE_MEMORY
        call PRINT_STR

        mov ah, 4ah
        mov bx, 0ffffh
        int 21h
        mov ax, bx
        mov bx, 16
        mul bx
        call PRINT_BYTE

        mov dx, offset LN
        call PRINT_STR

        ret
MEMORY_AVAILABLE endp


MEMORY_EXTENDED proc near
        mov dx, offset EXTENDED_MEMORY
        call PRINT_STR

   mov al, 30h
   out 70h, al
        in al, 71h
        mov al, 31h
   out 70h, al
   in al, 71h

   mov ah, al
```

```
        mov bh, al
        mov ax, bx

        mov bx, 16
        mul bx

        call PRINT_BYTE

        mov dx, offset LN
        call PRINT_STR

        ret
MEMORY_EXTENDED endp


MCB PROC near
        mov ah, 52h
        int 21h
        mov ax, es:[bx-2]
        mov es, ax
        mov dx, offset MCB_TABLE
        call PRINT_STR

MCB_loop:
   mov ax, es                ;адрес
   mov di, offset ADDRESS
   add di, 12
   call WRD_TO_HEX
   mov dx, offset ADDRESS
   call PRINT_STR
        mov dx, offset SPACE_STRING
        call PRINT_STR

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

        mov dx, offset STRING_SIZE   ;размер
        call PRINT_STR
        mov ax, es:[3]
        mov di, offset STRING_SIZE
        add di, 6
        mov bx, 16
        mul bx
        call PRINT_BYTE
        mov dx, offset SPACE_STRING
        call PRINT_STR

        mov bx, 8                 ;SC/SD
```

```
        mov dx, offset SC_SD
        call PRINT_STR
        mov cx, 7

loop_2:
        mov dl, es:[bx]
        mov ah, 02h
        int 21h
        inc bx
        loop loop_2

        mov dx, offset LN
        call PRINT_STR

        mov bx, es:[3h]
        mov al, es:[0h]
        cmp al, 5ah
        je FOR_END

        mov ax, es
        inc ax
        add ax, bx
        mov es, ax
        jmp MCB_loop

FOR_END:
        ret
MCB endp

TASK2 PROC near
   mov    ax, cs
   mov    es, ax
   mov    bx, offset END_CODE
   mov    ax, es
   mov    bx, ax
   mov    ah, 4ah
   int    21h
   ret
TASK2 endp

TASK3 PROC near
   mov    bx, 1000h        ;64kb
   mov    ah, 48h
   int    21h

   jb     memory_er      ;cf = 1
   jmp    FOR_END2

memory_er:
   mov    dx, offset MEMORY_ERROR
   call   PRINT_STR
```

```
FOR_END2:
   ret

TASK3 endp

BEGIN:
        call MEMORY_AVAILABLE
        call MEMORY_EXTENDED
        call TASK3
        call TASK2
        call MCB

        xor al, al
        mov ah, 4ch
        int 21h

END_CODE:
TESTPC ENDS

END START
```

Приложение Г

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

; Данные
AVAILABLE_MEMORY DB 'Available_ memory (bytes): ', '$'
EXTENDED_MEMORY DB 'Extended memory (bytes): ', '$'
MCB_TABLE DB 'MCB table: ', 0DH, 0AH, '$'
ADDRESS DB 'Address:     ', '$'
PSP DB 'PSP address:      ', '$'
STRING_SIZE DB 'Size: ', '$'
SC_SD DB 'SC/SD: ', '$'
LN DB 0DH,0AH,'$'
SPACE_STRING DB ' ', '$'
MEMORY_ERROR DB 'Error! Memory cant alloced', 0DH, 0AH, '$'

; 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 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 10 с/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

;-----------------------------;

PRINT_STR PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_STR endp


PRINT_BYTE PROC
        mov bx, 10
      mov cx, 0
loop_1:
        div bx
        push dx
        inc cx
        mov dx, 0
        cmp ax, 0
        jne loop_1
print:
        pop dx
        add dl,30h
        mov ah,02h
        int 21h
        loop print
        ret
PRINT_BYTE endp


MEMORY_AVAILABLE PROC near
        mov dx, offset AVAILABLE_MEMORY
        call PRINT_STR

        mov ah, 4ah
        mov bx, 0ffffh
        int 21h
        mov ax, bx
        mov bx, 16
        mul bx
        call PRINT_BYTE

        mov dx, offset LN
        call PRINT_STR

        ret
MEMORY_AVAILABLE endp
```

```
MEMORY_EXTENDED proc near
        mov dx, offset EXTENDED_MEMORY
        call PRINT_STR

    mov al, 30h
    out 70h, al
        in al, 71h
        mov al, 31h
    out 70h, al
    in al, 71h

    mov ah, al
        mov bh, al
        mov ax, bx

        mov bx, 16
        mul bx

        call PRINT_BYTE

        mov dx, offset LN
        call PRINT_STR

        ret
MEMORY_EXTENDED endp


MCB PROC near
        mov ah, 52h
        int 21h
        mov ax, es:[bx-2]
        mov es, ax
        mov dx, offset MCB_TABLE
        call PRINT_STR

MCB_loop:
    mov ax, es              ;адрес
    mov di, offset ADDRESS
    add di, 12
    call WRD_TO_HEX
    mov dx, offset ADDRESS
    call PRINT_STR
        mov dx, offset SPACE_STRING
        call PRINT_STR

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

```
        call PRINT_STR

        mov dx, offset STRING_SIZE   ;размер
        call PRINT_STR
        mov ax, es:[3]
        mov di, offset STRING_SIZE
        add di, 6
        mov bx, 16
        mul bx
        call PRINT_BYTE
        mov dx, offset SPACE_STRING
        call PRINT_STR

        mov bx, 8                ;SC/SD
        mov dx, offset SC_SD
        call PRINT_STR
        mov cx, 7

loop_2:
        mov dl, es:[bx]
        mov ah, 02h
        int 21h
        inc bx
        loop loop_2

        mov dx, offset LN
        call PRINT_STR

        mov bx, es:[3h]
        mov al, es:[0h]
        cmp al, 5ah
        je FOR_END

        mov ax, es
        inc ax
        add ax, bx
        mov es, ax
        jmp MCB_loop

FOR_END:
        ret
MCB endp

TASK2 PROC near
   mov    ax, cs
   mov    es, ax
   mov    bx, offset END_CODE
   mov    ax, es
   mov    bx, ax
   mov    ah, 4ah
   int    21h
   ret
```

```
TASK2 endp

TASK3 PROC near
   mov     bx, 1000h         ;64kb
   mov     ah, 48h
   int     21h

   jb      memory_er       ;cf = 1
   jmp     FOR_END2

memory_er:
   mov     dx, offset MEMORY_ERROR
   call    PRINT_STR

FOR_END2:
   ret

TASK3 endp

BEGIN:
        call MEMORY_AVAILABLE
        call MEMORY_EXTENDED
        call TASK2
        call TASK3
        call MCB

        xor al, al
        mov ah, 4ch
        int 21h

END_CODE:
TESTPC ENDS

END START
```