

**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**

Студент гр. 9382

\_\_\_\_\_

Кузьмин Д. И.

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 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.**

1) Были созданы необходимые функции для вывода

Информации о доступной памяти, расширенной памяти, а также всех блоков MSB.

2) Затем изменено распределение памяти таким образом, что память, которую программа не занимает – освобождается.

3) Далее было сделано распределение памяти с помощью функции 48h прерывания int 21h, но после освобождения памяти

4) Затем аналогичным образом распределение, но до освобождения памяти

Демонстрацию работы программы см. в приложении А

Исходный код см. в приложении Б

#### **Контрольные вопросы.**

1) Что означает "доступный объем памяти"?

Эта память, выделяемая операционной системой, которую может использовать программа

2) Где MCB блок Вашей программы в списке?

В первом случае к программе относится последний и предпоследний блок.

Во втором - предпоследний и предыдущий за ним.

В третьем - предпоследний и 2 предыдущих за ним.

В четвертом – предпоследний и предыдущий за ним.

3) Какой размер памяти занимает программа в каждом случае?

В первом случае она занимает 649056 байт.

Во втором – 304 байта

В третьем – 304 байт + 64кб

В четвертом – 304 байт

### **Выводы.**

Был изучен принцип организации управления памятью. Разработана программа, выводящая информацию о памяти.

## ПРИЛОЖЕНИЕ А. ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
H:\>lb3.com
Available memory: 648912
Extended memory: 15360
----MCB Type:004D
PSP Address/Extra data:0008
Memory size: 16
Data:
----MCB Type:004D
PSP Address/Extra data:0000
Memory size: 64
Data:
----MCB Type:004D
PSP Address/Extra data:0040
Memory size: 256
Data:
----MCB Type:004D
PSP Address/Extra data:0192
Memory size: 144
Data:
----MCB Type:005A
PSP Address/Extra data:0192
Memory size: 648912
Data:LB3
```

Вывод размера доступной памяти.

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
H:\>exe2bin lb3_2.exe, lb3_2.com
H:\>lb3_2.com
Memory for program: 160
Extended memory: 15360
----MCB Type:004D
PSP Address/Extra data:0008 Memory size: 16
Data:
----MCB Type:004D
PSP Address/Extra data:0000 Memory size: 64
Data:
----MCB Type:004D
PSP Address/Extra data:0040 Memory size: 256
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 144
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 160
Data:LB3_2
----MCB Type:005A
PSP Address/Extra data:0000 Memory size: 648736
Data:
```

Освобождение памяти. На программу выделяется только 10 параграфов (160 байт).

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
Memory for program: 65536
Extended memory: 15360
----MCB Type:004D
PSP Address/Extra data:0008 Memory size: 16
Data:
----MCB Type:004D
PSP Address/Extra data:0000 Memory size: 64
Data:
----MCB Type:004D
PSP Address/Extra data:0040 Memory size: 256
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 144
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 160
Data:LB3_3
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 65536
Data:LB3_3
----MCB Type:005A
PSP Address/Extra data:0000 Memory size: 583184
Data: LINK
```

Распределение памяти после освобождения памяти.

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
H:\>exe2bin lb3_4.exe, lb3_4.com
H:\>lb3_4.com
Memory for program: Memory error
Extended memory: 15360
----MCB Type:004D
PSP Address/Extra data:0008 Memory size: 16
Data:
----MCB Type:004D
PSP Address/Extra data:0000 Memory size: 64
Data:
----MCB Type:004D
PSP Address/Extra data:0040 Memory size: 256
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 144
Data:
----MCB Type:004D
PSP Address/Extra data:0192 Memory size: 160
Data:LB3_4
----MCB Type:005A
PSP Address/Extra data:0000 Memory size: 648736
Data:
```

Распределение памяти до освобождения. Был обработан carry flag.

## ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД

### Файл lb3.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:TESTPC
    ORG 100H
    .386
START: JMP BEGIN
; Данные
availablemem db 'Available memory: ', '$'
extendedmem db 'Extended memory: ', '$'
divisor dd 10
mcctype db '----MCB Type: ', 13, 10, '$'
pspadress db 'PSP Address/Extra data: ', '$'
memsize db ' Memory size: ', '$'
data db 'Data:', '$'
; 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
;-----
WRITE_MSG PROC near
    mov AH, 09h
    int 21h
    ret
WRITE_MSG 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, в AH - младшая
    pop CX
    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 10чную с/с, SI - адрес 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

        xor cx, cx

divide:

        xor dx, dx
        div divisor
        push dx
        inc cx
        cmp ax, 0
        jne divide

print:

        pop dx
        add dl, '0'
        mov al, dl
        int 29h
        loop print

        mov al, 13
        int 29h
        mov al, 10
        int 29h
        ret
WRD_TO_DEC ENDP

PRINT_MEM PROC NEAR

```



```

        mov DX, offset availablemem
        call WRITE_MSG

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

        mov eax, ebx
        mov ebx, 16      ;параметр
        mul ebx

        call WRD_TO_DEC

        ret
PRINT_MEM ENDP
;-----

PRINT_EXTENDED PROC NEAR

        mov DX, offset extendedmem
        call WRITE_MSG

        mov al, 30h
        out 70h, al
        in al, 71h
        mov bl, al
        mov al, 31h
        out 70h, al
        in al, 71h
        xor dx, dx

        mov ah, al
        mov al, bl
        call WRD_TO_DEC

        ret

PRINT_EXTENDED ENDP

PRINT_CURRENT_MCB PROC NEAR

        mov al, es:[00h]
        xor ah, ah
        mov di, offset mcbtype
        add di, 16
        call WRD_TO_HEX
        mov dx, offset mcbtype
        call WRITE_MSG

        mov ax, es:[01h]
        mov di, offset pspaddress
        add di, 25
        call WRD_TO_HEX
        mov dx, offset pspaddress
        call WRITE_MSG

        mov dx, offset memsize

```

```

        call WRITE_MSG
        mov ax, es:[03h]
        mov ebx, 16      ;парарпаф
        mul ebx
        call WRD_TO_DEC

        mov dx, offset data
        call WRITE_MSG
        mov bx, 0h

print_sym:
        cmp bx, 7h
        jge endthis
        mov al, es:[08h + bx]
        int 29h
        inc bx
        jmp print_sym

endthis:
        mov al, 13
        int 29h
        mov al, 10
        int 29h
        ret

PRINT_CURRENT_MCB ENDP

PRINT_MCBS PROC NEAR
        mov ah, 52h
        int 21h
        mov ax, es:[bx - 2]
        mov es, ax
        print_mcb:
        call PRINT_CURRENT_MCB
        mov bh, es:[0h]
        cmp bh, 05Ah
        je exit
        mov ax, es:[3h]
        inc ax
        mov bx, es
        add ax, bx
        mov es, ax

        jmp print_mcb
exit:
        ret

PRINT_MCBS ENDP
;КОД
BEGIN:
        call PRINT_MEM
        call PRINT_EXTENDED
        call PRINT_MCBS
        xor AL, AL
        mov AH, 4Ch
        int 21H

TESTPC ENDS
END START

```

## Файл lb3\_2.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:TESTPC
    ORG 100H
    .386
START: JMP BEGIN
; Данные
availablemem db 'Memory for program: ', '$'
extendedmem db 'Extended memory: ', '$'
divisor dd 10
mcctype db '----MCB Type:      ',13, 10, '$'
pspadress db 'PSP Address/Extra data:  ', '$'
memsize db ' Memory size: ', '$'
data db 'Data:', '$'
; 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
;-----
WRITE_MSG PROC near
    mov AH, 09h
    int 21h
    ret
WRITE_MSG 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, в AH - младшая
    pop CX
    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 10чную с/с, SI - адрес 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
;-----
WRD_TO_DEC PROC near

        xor cx, cx

divide:

        xor dx, dx
        div divisor
        push dx
        inc cx
        cmp ax, 0
        jne divide

print:

        pop dx
        add dl, '0'
        mov al, dl
        int 29h
        loop print

        mov al, 13
        int 29h
        mov al, 10
        int 29h
        ret
WRD_TO_DEC ENDP

```

PRINT\_MEM PROC NEAR

mov DX, offset availablemem  
call WRITE\_MSG

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

mov eax, ebx  
mov ebx, 16 ;параграф  
mul ebx

call WRD\_TO\_DEC

ret

PRINT\_MEM ENDP

;-----

PRINT\_EXTENDED PROC NEAR

mov DX, offset extendedmem  
call WRITE\_MSG

mov al, 30h  
out 70h, al  
in al, 71h  
mov bl, al  
mov al, 31h  
out 70h, al  
in al, 71h  
xor dx, dx

mov ah, al  
mov al, bl  
call WRD\_TO\_DEC

ret

PRINT\_EXTENDED ENDP

PRINT\_CURRENT\_MCB PROC NEAR

mov al, es:[00h]  
xor ah, ah  
mov di, offset mcbtype  
add di, 16  
call WRD\_TO\_HEX  
mov dx, offset mcbtype  
call WRITE\_MSG

mov ax, es:[01h]  
mov di, offset pspadress  
add di, 25  
call WRD\_TO\_HEX  
mov dx, offset pspadress  
call WRITE\_MSG

```

        mov dx, offset memsize
        call WRITE_MSG
        mov ax, es:[03h]
        mov ebx, 16      ;параграф
        mul ebx
        call WRD_TO_DEC

        mov dx, offset data
        call WRITE_MSG
        mov bx, 0h

print_sym:
        cmp bx, 7h
        jge endthis
        mov al, es:[08h + bx]
        int 29h
        inc bx
        jmp print_sym

endthis:
        mov al, 13
        int 29h
        mov al, 10
        int 29h
        ret

PRINT_CURRENT_MCB ENDP

PRINT_MCBS PROC NEAR
        mov ah, 52h
        int 21h
        mov ax, es:[bx - 2]
        mov es, ax
        print_mcb:
        call PRINT_CURRENT_MCB
        mov bh, es:[0h]
        cmp bh, 05Ah
        je exit
        mov ax, es:[3h]
        inc ax
        mov bx, es
        add ax, bx
        mov es, ax

        jmp print_mcb
exit:
        ret

PRINT_MCBS ENDP
;КОД
BEGIN:
        call PRINT_MEM
        call PRINT_EXTENDED
        call PRINT_MCBS
        xor AL, AL
        mov AH, 4Ch
        int 21H

```

```
TESTPC ENDS
END START
```

### Файл lb3\_3.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:TESTPC
    ORG 100H
    .386
START: JMP BEGIN
; Данные
availablemem db 'Memory for program: ', '$'
extendedmem db 'Extended memory: ', '$'
divisor dd 10
mcctype db '----MCB Type:      ',13, 10, '$'
pspadress db 'PSP Address/Extra data:      ', '$'
memsize db ' Memory size: ', '$'
data db 'Data:', '$'
memfail db 'Memory error', 13, 10, '$'
; 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
;-----
WRITE_MSG PROC near
    mov AH, 09h
    int 21h
    ret
WRITE_MSG 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, в AH - младшая
    pop CX
    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 10чную с/с, SI - адрес 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
;-----
WRD_TO_DEC PROC near

        xor cx, cx

divide:

        xor dx, dx
        div divisor
        push dx
        inc cx
        cmp ax, 0
        jne divide

print:

        pop dx
        add dl, '0'
        mov al, dl
        int 29h
        loop print

        mov al, 13
        int 29h
        mov al, 10
        int 29h
        ret
WRD_TO_DEC ENDP

```



PRINT\_MEM PROC NEAR

```
    mov DX, offset availablemem
    call WRITE_MSG
```

```
    mov bx, 0Ah
    mov ah, 4ah
    int 21h
```

```
    jc carry
    mov ah, 48h
    mov bx, 1000h
    int 21h
```

```
    mov eax, ebx
    mov ebx, 16      ;параграф
    mul ebx
```

```
    call WRD_TO_DEC
    jmp tothend
```

```
carry:
    mov dx, offset memfail
    call WRITE_MSG
tothend:
```

```
    ret
```

PRINT\_MEM ENDP

-----

PRINT\_EXTENDED PROC NEAR

```
    mov DX, offset extendedmem
    call WRITE_MSG
```

```
    mov al, 30h
    out 70h, al
    in al, 71h
    mov bl, al
    mov al, 31h
    out 70h, al
    in al, 71h
    xor dx, dx
```

```
    mov ah, al
    mov al, bl
    call WRD_TO_DEC
```

```
    ret
```

PRINT\_EXTENDED ENDP

PRINT\_CURRENT\_MCB PROC NEAR

```
    mov al, es:[00h]
    xor ah, ah
    mov di, offset mcbtype
    add di, 16
    call WRD_TO_HEX
    mov dx, offset mcbtype
    call WRITE_MSG

    mov ax, es:[01h]
    mov di, offset pspaddress
    add di, 25
    call WRD_TO_HEX
    mov dx, offset pspaddress
    call WRITE_MSG

    mov dx, offset memsize
    call WRITE_MSG
    mov ax, es:[03h]
    mov ebx, 16      ;параграф
    mul ebx
    call WRD_TO_DEC
```

```
    mov dx, offset data
    call WRITE_MSG
    mov bx, 0h
```

print\_sym:

```
    cmp bx, 7h
    jge endthis
    mov al, es:[08h + bx]
    int 29h
    inc bx
    jmp print_sym
```

endthis:

```
    mov al, 13
    int 29h
    mov al, 10
    int 29h
    ret
```

PRINT\_CURRENT\_MCB ENDP

PRINT\_MCBS PROC NEAR

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

print\_mcb:

```
    call PRINT_CURRENT_MCB
    mov bh, es:[0h]
    cmp bh, 05Ah
    je exit
    mov ax, es:[3h]
    inc ax
    mov bx, es
```

```

        add ax,bx
        mov es,ax

        jmp print_mcb
exit:
        ret

PRINT_MCBS ENDP
;КОД
BEGIN:

        call PRINT_MEM
        call PRINT_EXTENDED
        call PRINT_MCBS
        xor AL,AL
        mov AH,4Ch
        int 21H

TESTPC ENDS
END START

```

### Файл lb3\_4.asm

```

TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:TESTPC
    ORG 100H
    .386
START: JMP BEGIN
; Данные
availablemem db 'Memory for program: ', '$'
extendedmem db 'Extended memory: ', '$'
divisor dd 10
mcctype db '----MCB Type:      ',13, 10, '$'
pspadress db 'PSP Address/Extra data:      ', '$'
memsize db '  Memory size: ', '$'
data db 'Data:', '$'
memfail db 'Memory error', 13, 10, '$'

; 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
;-----
WRITE_MSG PROC near
    mov AH, 09h
    int 21h
    ret
WRITE_MSG 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, в AH - младшая
        pop CX
        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 10чную с/с, SI - адрес 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
;-----
WRD_TO_DEC PROC near

        xor cx, cx
divide:
        xor dx, dx
        div divisor

```

```

        push    dx
        inc     cx
        cmp     ax, 0
        jne     divide

print:
        pop     dx
        add     dl, '0'
        mov     al, dl
        int     29h
        loop    print

        mov     al, 13
        int     29h
        mov     al, 10
        int     29h
        ret

WRD_TO_DEC ENDP

PRINT_MEM PROC NEAR

        mov     DX, offset availablemem
        call    WRITE_MSG

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

        jc      carry

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

        mov     eax, ebx
        mov     ebx, 16      ;параграф
        mul     ebx

        call    WRD_TO_DEC
        jmp     tothend

        carry:
        mov     dx, offset memfail
        call    WRITE_MSG
        tothend:
        mov     bx, 0Ah
        mov     ah, 4ah
        int     21h

        ret

PRINT_MEM ENDP
;-----

PRINT_EXTENDED PROC NEAR

        mov     DX, offset extendedmem
        call    WRITE_MSG

```

```

        mov al, 30h
        out 70h, al
        in al, 71h
        mov bl, al
        mov al, 31h
        out 70h, al
        in al, 71h
        xor dx, dx

        mov ah, al
        mov al, bl
        call WRD_TO_DEC

        ret

PRINT_EXTENDED ENDP

PRINT_CURRENT_MCB PROC NEAR

        mov al, es:[00h]
        xor ah, ah
        mov di, offset mcbtype
        add di, 16
        call WRD_TO_HEX
        mov dx, offset mcbtype
        call WRITE_MSG

        mov ax, es:[01h]
        mov di, offset pspadress
        add di, 25
        call WRD_TO_HEX
        mov dx, offset pspadress
        call WRITE_MSG

        mov dx, offset memsize
        call WRITE_MSG
        mov ax, es:[03h]
        mov ebx, 16      ;параметр
        mul ebx
        call WRD_TO_DEC

        mov dx, offset data
        call WRITE_MSG
        mov bx, 0h

print_sym:
        cmp bx, 7h
        jge endthis
        mov al, es:[08h + bx]
        int 29h
        inc bx
        jmp print_sym

endthis:
        mov al, 13
        int 29h
        mov al, 10

```

```

        int 29h
        ret

PRINT_CURRENT_MCB ENDP

PRINT_MCBS PROC NEAR
        mov ah,52h
        int 21h
        mov ax,es:[bx - 2]
        mov es,ax
    print_mcb:
        call PRINT_CURRENT_MCB
        mov bh,es:[0h]
        cmp bh, 05Ah
        je exit
        mov ax,es:[3h]
        inc ax
        mov bx,es
        add ax,bx
        mov es,ax

        jmp print_mcb
    exit:
        ret

PRINT_MCBS ENDP
;КОД
BEGIN:
        call PRINT_MEM
        call PRINT_EXTENDED
        call PRINT_MCBS
        xor AL,AL
        mov AH,4Ch
        int 21H

TESTPC ENDS
END START

```