

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.8382

Синельников М.Р

Преподаватель

Ефремов М.А

Санкт-Петербург

2020

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.

Ход работы.

1)

```
C:\>os3_1.com
Available memory: 648912
Size of extended memory: 15360
MCB type: 4D :MCB seg: 0008 :MCB size: 16 ;
MCB type: 4D :MCB seg: 0000 :MCB size: 64 ;
MCB type: 4D :MCB seg: 0040 :MCB size: 256 ;
MCB type: 4D :MCB seg: 0192 :MCB size: 144 ;
MCB type: 5A :MCB seg: 0192 :MCB size: 648912 :OS3_1
```

рисунок 1 — шаг 1

2)

```
C:\>os3_2.com
Available memory: 648912
Size of extended memory: 15360
MCB type: 4D :MCB seg: 0008 :MCB size: 16 ;
MCB type: 4D :MCB seg: 0000 :MCB size: 64 ;
MCB type: 4D :MCB seg: 0040 :MCB size: 256 ;
MCB type: 4D :MCB seg: 0192 :MCB size: 144 ;
MCB type: 4D :MCB seg: 0192 :MCB size: 12576 :OS3_2
MCB type: 5A :MCB seg: 0000 :MCB size: 636320 :7hEP7.4P
```

рисунок 2 — шаг 2

3)

```
\>os3_3.com
available memory: 648912
size of extended memory: 15360
B type: 4D ;MCB seg: 0008 ;MCB size: 16 ;
B type: 4D ;MCB seg: 0000 ;MCB size: 64 ;
B type: 4D ;MCB seg: 0040 ;MCB size: 256 ;
B type: 4D ;MCB seg: 0192 ;MCB size: 144 ;
B type: 4D ;MCB seg: 0192 ;MCB size: 12688 ;OS3_3
B type: 4D ;MCB seg: 0192 ;MCB size: 65536 ;OS3_3
B type: 5A ;MCB seg: 0000 ;MCB size: 570656 ;
```

рисунок 3 — шаг 3

4)

```
C:\>os3_4.com
Available memory: 648912
ERROR
```

рисунок 4 - шаг 4

Контрольные вопросы.

1) Что означает «доступный объём памяти»?

Это объём оперативной памяти, который выделяется программе.

2) Где MCB блок Вашей программы в списке?

В первой и второй программе это четвёртый и пятый блоки, в третьей это четвёртый, пятый и шестой блоки так как у них один адрес сегментного владельца.

В данном случае это подтверждает ещё и то, что в пятом блоке название строки совпадает с названием запускаемой программы.

Во второй программе неиспользуемая память была освобождена, поэтому созданся новый свободный блок. В третьей программе запросили дополнительные 64кб, поэтому появляется второй MCB блок.

В четвёртом случае возникает ошибка, когда программа запрашивает память до того, как освободили неиспользуемую.

3) Какой размер памяти занимает программа в каждом случае?

Первая — все доступные 648912 байта + 144 байта = 649056 байт

Вторая — 12576 байта + 144 байта = 12720 байт, так как остальное освободили

Третья — 12688 байта + 144 байта + запрошенные 65536 байта = 78368 байт

В четвёртом случае произошла ошибка

Вывод.

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

Приложение А

Исходный код файла OS3_1.asm

```
TESTPC      SEGMENT
              ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
              ORG 100H ; резервирование места для PSP
START: JMP begin

Available_Memory db 'Available memory:      ',0DH,0AH,'$'
Extended_Memory_Size db 'Size of extended memory:      ',0DH,0AH,'$'
MCB_TYPE db 'MCB type:      ','$'
MCB_SIZE db 'MCB size:      ','$'
MCB_SEG db 'MCB seg:      ','$'
MCB_last_bytes db '      ',0DH,0AH,'$'

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 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

```

```

WRD_TO_DEC proc near
    push ax
    push CX
    push DX
    mov CX,10
loop_wd:
    div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp ax,0
    jnz loop_wd
end_l1:
    pop DX
    pop CX
    pop ax
    ret
WRD_TO_DEC ENDP

```

```

BYTE_TO_DEC PROC near

```

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

```
push CX
push DX
xor AH,AH
xor DX,DX
mov CX,10
loop_bd:
div CX
or DL,30h
mov [SI],DL
dec SI
xor DX,DX
cmp AX,10
jae loop_bd
cmp AL,00h
je end_1
or AL,30h
mov [SI],AL
end_1:
pop DX
pop CX
ret
BYTE_TO_DEC ENDP
```

```
print_msg proc near
    push ax
    mov ah,09h
    int 21h
    pop ax
    ret
```

```
print_msg endp
```

```
GET_MEMORY PROC near
    push ax
    push bx
    push cx
    push dx
    mov bx, 0ffffh
    mov ah, 4Ah
```

```

int 21h
mov ax, bx
mov cx, 10h
mul cx
lea si, Available_memory + 25
call WRD_TO_DEC
lea dx, Available_memory
call print_msg
pop dx
pop cx
pop bx
pop ax
ret

```

GET_MEMORY ENDP

Memory_Size proc near

```

push ax
push bx
push dx
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
sub dx, dx
lea si, Extended_Memory_Size + 29
call WRD_TO_DEC
lea dx, Extended_Memory_Size
call print_msg
pop dx
pop bx
pop ax
ret

```

Memory_Size endp


```

GET_last_bytes PROC    near
    push si
    push cx
    push bx
    push ax
    mov bx,0008h
    mov cx,4
    RE:
        mov ax,es:[bx]
        mov [si],ax
        add bx,2h
        add si,2h
        loop RE
    pop ax
    pop bx
    pop cx
    pop si
    ret
GET_last_bytes ENDP

```

```

Get_MCB proc near
    push ax
    push bx
    push cx
    push dx
    mov ah,52h
    int 21h
    mov es,es:[bx-2]
    mov bx,1
    repeat:
        sub ax,ax
        sub cx,cx
        sub di,di
        sub si,si
        mov al,es:[0000h]
        call BYTE_TO_HEX
        lea di,MCB_Type+12
        mov [di],ax
        cmp ax,4135h
        je MEN_BX

```

```

step:
    lea di,MCB_Seg+12
    mov ax,es:[0001h]
    call WRD_TO_HEX
    mov ax,es:[0003h]
    mov cx,10h
    mul cx
    lea si,MCB_Size+15
    call WRD_TO_DEC
    lea dx,MCB_Type
    call print_msg
    lea dx,MCB_Seg
    call print_msg
    lea dx,MCB_Size
    call print_msg

    lea si,MCB_last_bytes
    call GET_last_bytes
    lea dx,MCB_last_bytes
    call print_msg

    cmp bx,0
    jz END_P
    xor ax, ax
mov ax, es
add ax, es:[0003h]
inc ax
mov es, ax
    jmp repeat
END_P:
    pop dx
    pop cx
    pop bx
    pop ax
    ret
MEN_BX:
    mov     bx,0
    jmp step

```

Get_MCB ENDP

```
begin:
    call Get_Memory
    call Memory_Size
    call Get_MCB
    ret

TESTPC      ENDS
END START
```

Приложение В

Исходный код файла OS3_1.asm

```
TESTPC      SEGMENT
              ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
              ORG 100H ; резервирование места для PSP
```

```
START: JMP begin
```

```
Available_Memory db 'Available memory:      ',0DH,0AH,'$'
Extended_Memory_Size db 'Size of extended memory:      ',0DH,0AH,'$'
MCB_TYPE db 'MCB type:      ','$'
MCB_SIZE db 'MCB size:      ','$'
MCB_SEG db 'MCB seg:      ','$'
MCB_last_bytes db '      ',0DH,0AH,'$'
```

```
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 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
```

WRD_TO_DEC proc near

```
push ax
push CX
push DX
mov CX,10
loop_wd:
div CX
or DL,30h
mov [SI],DL
dec SI
xor DX,DX
cmp ax,0
jnz loop_wd
end_11:
pop DX
pop CX
pop ax
ret
WRD_TO_DEC ENDP
```

BYTE_TO_DEC PROC near

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

```

push CX
push DX
xor AH,AH
xor DX,DX
mov CX,10
loop_bd:
div CX
or DL,30h
mov [SI],DL
dec SI
xor DX,DX
cmp AX,10
jae loop_bd
cmp AL,00h
je end_1
or AL,30h
mov [SI],AL
end_1:
pop DX
pop CX
ret
BYTE_TO_DEC ENDP

```

```

print_msg proc near
    push ax
    mov ah,09h
    int 21h
    pop ax
    ret

```

```

print_msg endp

```

```

GET_MEMORY PROC near
    push ax
    push bx
    push cx
    push dx
    mov bx, 0ffffh
    mov ah, 4Ah
    int 21h

```

```

mov ax, bx
mov cx, 10h
mul cx
lea si, Available_memory + 25
call WRD_TO_DEC
lea dx, Available_memory
call print_msg
lea ax, end_of_prog
inc ax
mov bx, ax
mov al, 0
mov ah, 4Ah
int 21h
pop dx
pop cx
pop bx
pop ax
ret

```

GET_MEMORY ENDP

Memory_Size proc near

```

push ax
push bx
push dx
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
sub dx, dx
lea si, Extended_Memory_Size + 29
call WRD_TO_DEC
lea dx, Extended_Memory_Size
call print_msg
pop dx
pop bx

```

```
pop ax
ret
```

```
Memory_Size endp
```

```
GET_last_bytes PROC    near
    push si
    push cx
    push bx
    push ax
    mov bx,0008h
    mov cx,4
    RE:
        mov ax,es:[bx]
        mov [si],ax
        add bx,2h
        add si,2h
        loop RE
    pop ax
    pop bx
    pop cx
    pop si
    ret
GET_last_bytes ENDP
```

```
Get_MCB proc near
push ax
    push bx
    push cx
    push dx
    mov ah,52h
    int 21h
    mov es,es:[bx-2]
    mov bx,1
    repeat:
        sub ax,ax
        sub cx,cx
        sub di,di
        sub si,si
        mov al,es:[0000h]
```



```

        call BYTE_TO_HEX
        lea di,MCB_Type+12
        mov [di],ax
        cmp ax,4135h
        je MEN_BX
step:
        lea di,MCB_Seg+12
        mov ax,es:[0001h]
        call WRD_TO_HEX
        mov ax,es:[0003h]
        mov cx,10h
        mul cx
        lea si,MCB_Size+15
        call WRD_TO_DEC
        lea dx,MCB_Type
        call print_msg
        lea dx,MCB_Seg
        call print_msg
        lea dx,MCB_Size
        call print_msg

        lea si,MCB_last_bytes
        call GET_last_bytes
        lea dx,MCB_last_bytes
        call print_msg

        cmp bx,0
        jz END_P
        xor ax, ax
mov ax, es
add ax, es:[0003h]
inc ax
mov es, ax
        jmp repeat
END_P:
        pop dx
        pop cx
        pop bx
        pop ax
        ret
MEN_BX:

```

```

        mov     bx,0
        jmp step

Get_MCB ENDP

begin:
    call Get_Memory
    call Memory_Size
    call Get_MCB
    ret
end_of_prog:

TESTPC      ENDS
END START

```

Приложение С

Исходный код файла OS3_3.asm

```
TESTPC      SEGMENT
              ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
              ORG 100H ; резервирование места для PSP
START: JMP begin

Available_Memory db 'Available memory:      ',0DH,0AH,'$'
Extended_Memory_Size db 'Size of extended memory:      ',0DH,0AH,'$'
MCB_TYPE db 'MCB type:      ','$'
MCB_SIZE db 'MCB size:      ','$'
MCB_SEG db 'MCB seg:      ','$'
MCB_last_bytes db '      ',0DH,0AH,'$'

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 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

```

```

WRD_TO_DEC proc near
    push ax
    push CX
    push DX
    mov CX,10
loop_wd:
    div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp ax,0
    jnz loop_wd
end_11:
    pop DX
    pop CX
    pop ax
    ret
WRD_TO_DEC ENDP

```

```

BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
    push CX
    push DX
    xor AH,AH

```

```

    xor DX,DX
    mov CX,10
loop_bd:
    div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp AX,10
    jae loop_bd
    cmp AL,00h
    je end_1
    or AL,30h
    mov [SI],AL
end_1:
    pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

```

```

print_msg proc near
    push ax
    mov ah,09h
    int 21h
    pop ax
    ret

```

```

print_msg endp

```

```

GET_MEMORY PROC near
    push ax
    push bx
    push cx
    push dx
    mov bx, 0ffffh
    mov ah, 4Ah
    int 21h
    mov ax, bx
    mov cx, 10h
    mul cx
    lea si, Available_memory + 25
    call WRD_TO_DEC

```

```

    lea dx, Available_memory
    call print_msg
    lea ax, end_of_prog
    inc ax
    mov bx, ax
    mov al, 0
    mov ah, 4Ah
    int 21h
    mov bx, 1000h
    mov ah, 48h
    int 21h
    pop dx
    pop cx
    pop bx
    pop ax
    ret

```

GET_MEMORY ENDP

Memory_Size proc near

```

    push ax
    push bx
    push dx
    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
    sub dx, dx
    lea si, Extended_Memory_Size + 29
    call WRD_TO_DEC
    lea dx, Extended_Memory_Size
    call print_msg
    pop dx
    pop bx
    pop ax
    ret

```

Memory_Size endp

GET_last_bytes PROC near

```
    push si
    push cx
    push bx
    push ax
    mov bx,0008h
    mov cx,4
    RE:
        mov ax,es:[bx]
        mov [si],ax
        add bx,2h
        add si,2h
        loop RE
    pop ax
    pop bx
    pop cx
    pop si
    ret
```

GET_last_bytes ENDP

Get_MCB proc near

```
push ax
    push bx
    push cx
    push dx
    mov ah,52h
    int 21h
    mov es,es:[bx-2]
    mov bx,1
    repeat:
        sub ax,ax
        sub cx,cx
        sub di,di
        sub si,si
        mov al,es:[0000h]
        call BYTE_TO_HEX
        lea di,MCB_Type+12
        mov [di],ax
        cmp ax,4135h
        je MEN_BX
```

```

step:
    lea di,MCB_Seg+12
    mov ax,es:[0001h]
    call WRD_TO_HEX
    mov ax,es:[0003h]
    mov cx,10h
    mul cx
    lea si,MCB_Size+15
    call WRD_TO_DEC
    lea dx,MCB_Type
    call print_msg
    lea dx,MCB_Seg
    call print_msg
    lea dx,MCB_Size
    call print_msg

    lea si,MCB_last_bytes
    call GET_last_bytes
    lea dx,MCB_last_bytes
    call print_msg

    cmp bx,0
    jz END_P
    xor ax, ax

    mov ax, es
    add ax, es:[0003h]
    inc ax
    mov es, ax
    jmp repeat
END_P:
    pop dx
    pop cx
    pop bx
    pop ax
    ret
MEN_BX:
    mov     bx,0
    jmp step

```

Get_MCB ENDP

begin:


```
call Get_Memory
call Memory_Size
call Get_MCB
ret
end_of_prog:

TESTPC      ENDS
END START
```

Приложение D

Исходный код файла OS3_4.asm

```
TESTPC      SEGMENT
              ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
              ORG 100H ; резервирование места для PSP
START: JMP begin

Available_Memory db 'Available memory:      ',0DH,0AH,'$'
Extended_Memory_Size db 'Size of extended memory:      ',0DH,0AH,'$'
MCB_TYPE db 'MCB type:      ','$'
MCB_SIZE db 'MCB size:      ','$'
MCB_SEG db 'MCB seg:      ','$'
MCB_last_bytes db '      ',0DH,0AH,'$'
ERROR db ' ERROR ',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 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

WRD_TO_DEC proc near

```
push ax
push CX
push DX
mov CX,10
loop_wd:
div CX
or DL,30h
mov [SI],DL
dec SI
xor DX,DX
cmp ax,0
jnz loop_wd
```

end_11:

```
pop DX
pop CX
pop ax
ret
```

WRD_TO_DEC ENDP

BYTE_TO_DEC PROC near

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

```
push CX
push DX
```

```

xor AH,AH
xor DX,DX
mov CX,10
loop_bd:
div CX
or DL,30h
mov [SI],DL
dec SI
xor DX,DX
cmp AX,10
jae loop_bd
cmp AL,00h
je end_1
or AL,30h
mov [SI],AL
end_1:
pop DX
pop CX
ret
BYTE_TO_DEC ENDP

```

```

print_msg proc near
    push ax
    mov ah,09h
    int 21h
    pop ax
    ret

```

```

print_msg endp

```

```

GET_MEMORY PROC near
    push ax
    push bx
    push dx
    mov bx, 0ffffh
    mov ah, 4Ah
    int 21h
    mov ax, bx
    mov cx, 10h
    mul cx
    lea si, Available_memory + 25
    call WRD_TO_DEC

```

```

    lea dx, Available_memory
    call print_msg
    mov bx, 1000h
    mov ah, 48h
    int 21h
    mov cx, 1
    jc end1
    mov cx, 2
    lea ax, end_of_prog
    inc ax
    mov bx, ax
    mov al, 0
    mov ah, 4Ah
    int 21h
end1:
    pop dx
    pop bx
    pop ax
    ret

```

GET_MEMORY ENDP

```

Memory_Size proc near
    push ax
    push bx
    push dx
    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
    sub dx, dx
    lea si, Extended_Memory_Size + 29
    call WRD_TO_DEC
    lea dx, Extended_Memory_Size
    call print_msg
    pop dx
    pop bx

```

```
    pop ax
    ret
```

```
Memory_Size endp
```

```
GET_last_bytes PROC    near
    push si
    push cx
    push bx
    push ax
    mov bx,0008h
    mov cx,4
    RE:
        mov ax,es:[bx]
        mov [si],ax
        add bx,2h
        add si,2h
        loop RE
    pop ax
    pop bx
    pop cx
    pop si
    ret
GET_last_bytes ENDP
```

```
Get_MCB proc near
    push ax
    push bx
    push cx
    push dx
    mov ah,52h
    int 21h
    mov es,es:[bx-2]
    mov bx,1
    repeat:
        sub ax,ax
        sub cx,cx
        sub di,di
        sub si,si
        mov al,es:[0000h]
        call BYTE_TO_HEX
        lea di,MCB_Type+12
```

```

        mov [di],ax
        cmp ax,4135h
        je MEN_BX
step:
        lea di,MCB_Seg+12
        mov ax,es:[0001h]
        call WRD_TO_HEX
        mov ax,es:[0003h]
        mov cx,10h
        mul cx
        lea si,MCB_Size+15
        call WRD_TO_DEC
        lea dx,MCB_Type
        call print_msg
        lea dx,MCB_Seg
        call print_msg
        lea dx,MCB_Size
        call print_msg

        lea si,MCB_last_bytes
        call GET_last_bytes
        lea dx,MCB_last_bytes
        call print_msg

        cmp bx,0
        jz END_P
        xor ax, ax
mov ax, es
add ax, es:[0003h]
inc ax
mov es, ax
        jmp repeat
END_P:
        pop dx
        pop cx
        pop bx
        pop ax
        ret
MEN_BX:
        mov     bx,0
        jmp step

```

Get_MCB ENDP

begin:

call Get_Memory

mov ax,2

cmp cx,ax

je further

lea dx,ERROR

call print_msg

jmp finish

further:

call Memory_Size

call Get_MCB

finish:

ret

end_of_prog:

TESTPC ENDS

END START