

**МИНОБРНАУКИ РОССИИ**  
**Санкт-Петербургский государственный**  
**электротехнический университет**  
**«ЛЭТИ» им. В.И. Ульянова (Ленина)**  
**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №3**  
**по дисциплине «Операционные системы»**  
**Тема: Исследование организации управления основной памятью.**

Студент гр. 8382

Терехов А.Е.

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.

Был реализован загрузочный .com модуль, выводящий на экран размер доступной и расширенной памяти и цепочку блоков управления памятью. С кодом можно ознакомиться в приложении А, а результат работы программы представлен на рисунке 1. Так как в DOSBOX отсутствует прокрутка вывод был перенаправлен в файл.

```
Доступная память: 648912
Расширенная память: 58940 H

Владелец:MS DOS
Размер: 16 байт
Последние байты:  ? ? ? ? ? ? ? ?

Владелец:свободный участок
Размер: 64 байт
Последние байты:  ? ? ? ? ? ? ? ?

Владелец:0040
Размер: 256 байт
Последние байты:  ? ? ? ? ? ? ? ?

Владелец:0192
Размер: 144 байт
Последние байты:  ? ? ? ? ? ? ? ?

Владелец:0192
Размер: 648912 байт
Последние байты:  LR3 1 ? ? ?
```

Рисунок 1. Результат работы первой программы.

Затем программа была модифицирована, добавлено освобождение неиспользуемой памяти. С текстом программы можно ознакомиться в приложении Б, а с результатом работы на рисунке 2.

```
Доступная память: 648912
Расширенная память: 58940 Н
освободил

Владелец:MS DOS
Размер:      16 байт
Последние байты:  ??????????

Владелец:свободный участок
Размер:      64 байт
Последние байты:  ??????????

Владелец:0040
Размер:      256 байт
Последние байты:  ??????????

Владелец:0192
Размер:      144 байт
Последние байты:  ??????????

Владелец:0192
Размер:      15792 байт
Последние байты:  LR3_2????

Владелец:свободный участок
Размер:      633104 байт
Последние байты:  ??????????
```

Рисунок 2. Результат работы второй программы.

Было добавлено выделение 64 Кбайт памяти. Код третьей программы находится в приложении В, а результат на рисунке 3.

```

Доступная память: 648912
Расширенная память: 58940 Н
освободил
выделил

Владелец: MS DOS
Размер:      16 байт
Последние байты:      ?????????

Владелец: свободный участок
Размер:      64 байт
Последние байты:      ?????????

Владелец: 0040
Размер:      256 байт
Последние байты:      ?????????

Владелец: 0192
Размер:      144 байт
Последние байты:      ?????????

Владелец: 0192
Размер:      16576 байт
Последние байты:      LR3_3????

Владелец: 0192
Размер:      65536 байт
Последние байты:      LR3_3????

Владелец: свободный участок
Размер:      566768 байт
Последние байты:      ЁоўТІСЛ

```

Рисунок 3. Результат работы третьей программы.

Затем исходная программа была изменена так, что сначала происходило выделение 64 Кбайт памяти, а затем освобождение неиспользуемой памяти. Текст такой программы расположен в приложении Г, а результат на рисунке 4.

```

Доступная память: 648912
Расширенная память: 58940 Н
не выделил
освободил

Владелец:MS DOS
Размер:      16 байт
Последние байты:      ?????????

Владелец:свободный участок
Размер:      64 байт
Последние байты:      ?????????

Владелец:0040
Размер:      256 байт
Последние байты:      ?????????

Владелец:0192
Размер:      144 байт
Последние байты:      ?????????

Владелец:0192
Размер:      16576 байт
Последние байты:      LR3_4????

Владелец:свободный участок
Размер:      632320 байт
Последние байты:      P1□□PH

```

Рисунок 4. Результат работы четвертой программы.

### Ответы на вопросы.

1. Что означает "доступный объем памяти"?

Максимальный объем памяти, который может использовать программа.

2. Где MSB блок Вашей программы в списке?

Там, где владельцем указан 0192.

3. Какой размер памяти занимает программа в каждом случае?

Первая:  $144 + 648912 = 649056$  байт.

Вторая:  $15792 + 144 = 15936$  байт.

Третья:  $144 + 16576 + 65536 = 82256$  байт.

Четвертая:  $144 + 16576 = 16720$  байт.

### **Вывод.**

В лабораторной работе была исследована организация управления памятью в ОС DOS. Изучены функции прерывания 21H для работы с памятью.

## ПРИЛОЖЕНИЕ А

```

CODESEG SEGMENT
    ASSUME cs:CODESEG, ds:CODESEG, es:NOTHING, ss:NOTHING
    ORG 100H
START: jmp BEGIN
AVALIABLE_MEM    db 'Доступная память:                ', 0DH, 0AH, '$'
EXT_MEM          db 'Расширенная память:                H', 0DH, 0AH, '$'
NEW_LINE         db 0DH, 0AH, '$'
OWNER            db 'Владелец:$'
FREE             db 'свободный участок', 0DH, 0AH, '$'
OSXMSUBM         db 'OS XMS UMB', 0DH, 0AH, '$'
TOP_MEM          db 'исключенная верхняя память драйверов', 0DH, 0AH, '$'
MSDOS            db 'MS DOS', 0DH, 0AH, '$'
TAKEN386         db 'блоки 386MAX UMB', 0DH, 0AH, '$'
BLOCKED386       db 'заблокирован 386MAX', 0DH, 0AH, '$'
OWNED386         db '386MAX UMB', 0DH, 0AH, '$'
LAST_BYTES       db 'Последние байты:                $'
MCB_SIZE         db 'Размер:                байт', 0DH, 0AH, '$'
OWNER_2          db '                ', 0DH, 0AH, '$'
; Процедура печати строки
WRITE PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE ENDP
;-----
TETR_TO_HEX PROC near
    and al, 0Fh
    cmp al, 09
    jbe NEXT
    add al, 07
NEXT:    add al, 30h; код нуля
    ret
TETR_TO_HEX 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
    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 байта в 10с/с, si - адрес поля младшей цифры
; al содержит исходный байт
    push ax
    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
        pop ax
        ret

```



BYTE\_TO\_DEC ENDP

;-----

BEGIN:

; вывод размера доступной памяти

mov ah, 4Ah

mov bx, 0FFFFh

int 21h

mov ax, bx

mov bx, 10h

mul bx

lea si, AVALIABLE\_MEM

add si, 23

call WRD\_TO\_DEC

lea dx, AVALIABLE\_MEM

call WRITE

; вывод размера расширенной памяти

mov al, 30h

out 70h, al

in al, 71h

mov bl, al

mov al, 31h

out 70h, al

in al, 71h

lea si, EXT\_MEM

add si, 24

xor dx, dx

call WRD\_TO\_DEC

lea dx, EXT\_MEM

call WRITE

; вывод цепочки блоков управления памятью

mov ah, 52h

int 21h

mov ax, es:[bx-2]

mov es, ax

next\_mcb:

lea dx, NEW\_LINE

call WRITE

lea dx, OWNER

call WRITE

mov bx, es:[1]

; switch owner

lea dx, FREE

cmp bx, 0000h

je EQUAL

lea dx, OSXMSUBM

cmp bx, 0006h

je EQUAL

lea dx, TOP\_MEM

cmp bx, 0007h

je EQUAL

lea dx, MSDOS

cmp bx, 0008h

je EQUAL

lea dx, TAKEN386

```

        cmp bx, 0FFFAh
        je EQUAL
        lea dx, BLOCKED386
        cmp bx, 0FFFDh
        je EQUAL
        lea dx, OWNED386
        cmp bx, 0FFFEh
        je EQUAL
            jmp NOT_EQUAL
EQUAL:
        call WRITE
        jmp MCB_SZ
NOT_EQUAL:
        lea di, OWNER_2
        add di, 6
        mov ax, bx
        call WRD_TO_HEX
        mov dx, di
        call WRITE
        lea dx, MCB_SIZE
MCB_SZ:
        mov ax, es:[3]
        mov bx, 10h
        mul bx
        lea si, MCB_SIZE
        add si, 13
        call WRD_TO_DEC
        lea dx, MCB_SIZE
        call WRITE
        ; last bytes
        lea dx, LAST_BYTES
        call WRITE
        mov cx, 8
        xor bx, bx
        mov ah, 2
        next_b:
            mov dl, es:[bx+8h]
            int 21h
            inc bx
            loop next_b
        lea dx, NEW_LINE
        call WRITE
        mov al, es:[0]
        cmp al, 5ah
        je MCB_end
        mov ax, es:[3]
        mov bx, es
        add bx, ax
        inc bx
        mov es, bx
        jmp next_mcb
MCB_end:
        xor al, al
        mov ah, 4Ch
        int 21h
CODESEG ENDS
END START ;конец модуля, START - точка входа

```

## ПРИЛОЖЕНИЕ Б

```

CODESEG SEGMENT
    ASSUME cs:CODESEG, ds:CODESEG, es:NOTHING, ss:NOTHING
    ORG 100H
START: jmp BEGIN
AVALIABLE_MEM db 'Доступная память:                ', 0DH, 0AH, '$'
EXT_MEM       db 'Расширенная память:                H', 0DH, 0AH, '$'
NEW_LINE      db 0DH, 0AH, '$'
OWNER         db 'Владелец:$'
FREE          db 'свободный участок', 0DH, 0AH, '$'
OSXMSUBM      db 'OS XMS UMB', 0DH, 0AH, '$'
TOP_MEM       db 'исключенная верхняя память драйверов', 0DH, 0AH, '$'
MSDOS         db 'MS DOS', 0DH, 0AH, '$'
TAKEN386      db 'блоки 386MAX UMB', 0DH, 0AH, '$'
BLOCKED386    db 'заблокирован 386MAX', 0DH, 0AH, '$'
OWNED386      db '386MAX UMB', 0DH, 0AH, '$'
LAST_BYTES    db 'Последние байты:                $'
MCB_SIZE      db 'Размер:                байт', 0DH, 0AH, '$'
OWNER_2       db '                ', 0DH, 0AH, '$'
FR_OK         db 'освободил', 0DH, 0AH, '$'
FR_ERR        db 'не освободил', 0DH, 0AH, '$'
; Процедура печати строки
WRITE PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE ENDP
;-----
TETR_TO_HEX PROC near
    and al, 0Fh
    cmp al, 09
    jbe NEXT
    add al, 07
NEXT:    add al, 30h; код нуля
    ret
TETR_TO_HEX 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
    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 байта в 10с/с, si - адрес поля младшей цифры
; al содержит исходный байт
    push ax
    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

```

```

        pop ax
        ret
BYTE_TO_DEC ENDP
;-----

BEGIN:
    ; вывод размера доступной памяти
    mov ah, 4Ah
    mov bx, 0FFFFh
    int 21h
    mov ax, bx
    mov bx, 10h
    mul bx
    lea si, AVAILIABLE_MEM
    add si, 23
    call WRD_TO_DEC
    lea dx, AVAILIABLE_MEM
    call WRITE

    ; вывод размера расширенной памяти

    mov al, 30h
    out 70h, al
    in al, 71h
    mov bl, al
    mov al, 31h
    out 70h, al
    in al, 71h
    lea si, EXT_MEM
    add si, 24
    xor dx, dx
    call WRD_TO_DEC
    lea dx, EXT_MEM
    call WRITE

    lea bx, PR_END
    mov ah, 4Ah
    int 21h

    jnc FREE_OK
    jmp FREE_ERR

FREE_OK:
    lea dx, FR_OK
    jmp END_FREE
FREE_ERR:
    lea dx, FR_ERR
END_FREE:
    call WRITE

    ; вывод цепочки блоков управления памятью
    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    mov es, ax
next_mcb:
    lea dx, NEW_LINE

```

```

        call WRITE
        lea dx, OWNER
        call WRITE
        mov bx, es:[1]
        ; switch owner
        lea dx, FREE
    cmp bx, 0000h
    je EQUAL
    lea dx, OSXMSUBM
    cmp bx, 0006h
    je EQUAL
    lea dx, TOP_MEM
    cmp bx, 0007h
    je EQUAL
    lea dx, MSDOS
    cmp bx, 0008h
    je EQUAL
    lea dx, TAKEN386
    cmp bx, 0FFFAh
    je EQUAL
    lea dx, BLOCKED386
    cmp bx, 0FFFDh
    je EQUAL
    lea dx, OWNED386
    cmp bx, 0FFFEh
    je EQUAL
        jmp NOT_EQUAL
EQUAL:
        call WRITE
        jmp MCB_SZ
NOT_EQUAL:
        lea di, OWNER_2
        add di, 6
        mov ax, bx
        call WRD_TO_HEX
        mov dx, di
        call WRITE
        lea dx, MCB_SIZE
MCB_SZ:
        mov ax, es:[3]
        mov bx, 10h
        mul bx
        lea si, MCB_SIZE
        add si, 13
        call WRD_TO_DEC
        lea dx, MCB_SIZE
        call WRITE
        ; last bytes
        lea dx, LAST_BYTES
        call WRITE
        mov cx, 8
        xor bx, bx
        mov ah, 2
    next_b:
        mov dl, es:[bx+8h]
        int 21h
        inc bx

```

```

                                loop next_b
                                lea dx, NEW_LINE
                                call WRITE
                                mov al, es:[0]
                                cmp al, 5ah
                                je MCB_end

                                mov ax, es:[3]
                                mov bx, es
                                add bx, ax
                                inc bx
                                mov es, bx
                                jmp next_mcb
MCB_end:
xor al,al
mov ah,4Ch
int 21h
PR_END:
CODESEG ENDS
END START

```

## ПРИЛОЖЕНИЕ В

```

CODESEG SEGMENT
    ASSUME cs:CODESEG, ds:CODESEG, es:NOTHING, ss:NOTHING
    ORG 100H
START: jmp BEGIN
AVALIABLE_MEM db 'Доступная память:                ', 0DH, 0AH, '$'
EXT_MEM       db 'Расширенная память:                H', 0DH, 0AH, '$'
NEW_LINE      db 0DH, 0AH, '$'
OWNER         db 'Владелец:$'
FREE          db 'свободный участок', 0DH, 0AH, '$'
OSXMSUBM      db 'OS XMS UMB', 0DH, 0AH, '$'
TOP_MEM       db 'исключенная верхняя память драйверов', 0DH, 0AH, '$'
MSDOS         db 'MS DOS', 0DH, 0AH, '$'
TAKEN386      db 'блоки 386MAX UMB', 0DH, 0AH, '$'
BLOCKED386    db 'заблокирован 386MAX', 0DH, 0AH, '$'
OWNED386      db '386MAX UMB', 0DH, 0AH, '$'
LAST_BYTES    db 'Последние байты:                $'
MCB_SIZE      db 'Размер:                байт', 0DH, 0AH, '$'
OWNER_2       db '                ', 0DH, 0AH, '$'
FR_OK         db 'освободил', 0DH, 0AH, '$'
FR_ERR        db 'не освободил', 0DH, 0AH, '$'
GV_OK         db 'выделил', 0DH, 0AH, '$'
GV_ERR        db 'не выделил', 0DH, 0AH, '$'
; Процедура печати строки
WRITE PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE ENDP
;-----
TETR_TO_HEX PROC near
    and al, 0Fh
    cmp al, 09
    jbe NEXT
    add al, 07
NEXT:    add al, 30h; код нуля
    ret
TETR_TO_HEX 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
    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 байта в 10с/с, si - адрес поля младшей цифры
; al содержит исходный байт
    push ax
    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_1:  pop dx
        pop cx
        pop ax
        ret
BYTE_TO_DEC ENDP
;-----

BEGIN:

        ; вывод размера доступной памяти
        mov ah, 4Ah
        mov bx, 0FFFFh
        int 21h
        mov ax, bx
        mov bx, 10h
        mul bx
        lea si, AVALIABLE_MEM
        add si, 23
        call WRD_TO_DEC
        lea dx, AVALIABLE_MEM
        call WRITE

        ; вывод размера расширенной памяти

        mov al, 30h
        out 70h, al
        in al, 71h
        mov bl, al
        mov al, 31h
        out 70h, al
        in al, 71h
        lea si, EXT_MEM
        add si, 24
        xor dx, dx
        call WRD_TO_DEC
        lea dx, EXT_MEM
        call WRITE
        ; освобождение памяти
        lea bx, PR_END
        mov ah, 4Ah
        int 21h

        jnc FREE_OK
        jmp FREE_ERR

FREE_OK:
        lea dx, FR_OK
        jmp END_FREE
FREE_ERR:
        lea dx, FR_ERR
END_FREE:
        call WRITE

        ; запрос 64 килобайт памяти

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

```

```

jnc GIVE_OK
jmp GIVE_ERR

GIVE_OK:
lea dx, GV_OK
jmp END_GIVE
GIVE_ERR:
lea dx, GV_ERR
END_GIVE:
call WRITE
; вывод цепочки блоков управления памятью
mov ah, 52h
int 21h
mov ax, es:[bx-2]
mov es, ax
next_mcb:
    lea dx, NEW_LINE
    call WRITE
    lea dx, OWNER
    call WRITE
    mov bx, es:[1]
    ; switch owner
    lea dx, FREE
    cmp bx, 0000h
    je EQUAL
    lea dx, OSXMSUBM
    cmp bx, 0006h
    je EQUAL
    lea dx, TOP_MEM
    cmp bx, 0007h
    je EQUAL
    lea dx, MSDOS
    cmp bx, 0008h
    je EQUAL
    lea dx, TAKEN386
    cmp bx, 0FFFAh
    je EQUAL
    lea dx, BLOCKED386
    cmp bx, 0FFFDh
    je EQUAL
    lea dx, OWNED386
    cmp bx, 0FFFEh
    je EQUAL
    jmp NOT_EQUAL
EQUAL:
    call WRITE
    jmp MCB_SZ
NOT_EQUAL:
    lea di, OWNER_2
    add di, 6
    mov ax, bx
    call WRD_TO_HEX
    mov dx, di
    call WRITE
    lea dx, MCB_SIZE
MCB_SZ:
    mov ax, es:[3]

```

```

        mov bx,10h
        mul bx
        lea si, MCB_SIZE
        add si, 13
        call WRD_TO_DEC
        lea dx, MCB_SIZE
        call WRITE
        ; last bytes
        lea dx, LAST_BYTES
        call WRITE
        mov cx, 8
        xor bx, bx
        mov ah, 2
next_b:
        mov dl, es:[bx+8h]
        int 21h
        inc bx
        loop next_b
        lea dx, NEW_LINE
        call WRITE
        mov al, es:[0]
        cmp al, 5ah
        je MCB_end

        mov ax, es:[3]
        mov bx, es
        add bx, ax
        inc bx
        mov es, bx
        jmp next_mcb
MCB_end:
        xor al,al
        mov ah,4Ch
        int 21h
PR_END:
CODESEG ENDS
END START

```

## ПРИЛОЖЕНИЕ Г

```

CODESEG SEGMENT
    ASSUME cs:CODESEG, ds:CODESEG, es:NOTHING, ss:NOTHING
    ORG 100H
START: jmp BEGIN
AVALIABLE_MEM    db 'Доступная память:                ', 0DH, 0AH, '$'
EXT_MEM          db 'Расширенная память:                H', 0DH, 0AH, '$'
NEW_LINE         db 0DH, 0AH, '$'
OWNER            db 'Владелец:$'
FREE             db 'свободный участок', 0DH, 0AH, '$'
OSXMSUBM         db 'OS XMS UMB', 0DH, 0AH, '$'
TOP_MEM          db 'исключенная верхняя память драйверов', 0DH, 0AH, '$'
MSDOS            db 'MS DOS', 0DH, 0AH, '$'
TAKEN386         db 'блоки 386MAX UMB', 0DH, 0AH, '$'
BLOCKED386       db 'заблокирован 386MAX', 0DH, 0AH, '$'
OWNED386         db '386MAX UMB', 0DH, 0AH, '$'
LAST_BYTES       db 'Последние байты:                $'
MCB_SIZE         db 'Размер:                байт', 0DH, 0AH, '$'
OWNER_2          db '                ', 0DH, 0AH, '$'
GV_OK            db 'выделил', 0DH, 0AH, '$'
GV_ERR           db 'не выделил', 0DH, 0AH, '$'
FR_OK            db 'освободил', 0DH, 0AH, '$'
FR_ERR           db 'не освободил', 0DH, 0AH, '$'
; Процедура печати строки
WRITE PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE ENDP
;-----
TETR_TO_HEX PROC near
    and al, 0Fh
    cmp al, 09
    jbe NEXT
    add al, 07
NEXT:    add al, 30h; код нуля
    ret
TETR_TO_HEX 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
    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 байта в 10с/с, si - адрес поля младшей цифры
; al содержит исходный байт
    push ax
    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_1:  pop dx
        pop cx
        pop ax
        ret
BYTE_TO_DEC ENDP
;-----

BEGIN:
        ; вывод размера доступной памяти
        mov ah, 4Ah
        mov bx, 0FFFFh
        int 21h
        mov ax, bx
        mov bx, 10h
        mul bx
        lea si, AVAILIABLE_MEM
        add si, 23
        call WRD_TO_DEC
        lea dx, AVAILIABLE_MEM
        call WRITE

        ; вывод размера расширенной памяти

        mov al, 30h
        out 70h, al
        in al, 71h
        mov bl, al
        mov al, 31h
        out 70h, al
        in al, 71h
        lea si, EXT_MEM
        add si, 24
        xor dx, dx
        call WRD_TO_DEC
        lea dx, EXT_MEM
        call WRITE

        ; запрос 64 килобайт памяти

        mov ah, 48h
        mov bx, 400h
        int 21h
        jnc GIVE_OK
        jmp GIVE_ERR

GIVE_OK:
        lea dx, GV_OK
        jmp END_GIVE
GIVE_ERR:
        lea dx, GV_ERR
END_GIVE:
        call WRITE
        ; освобождение памяти
        lea bx, PR_END
        mov ah, 4Ah
        int 21h

```

```

jnc FREE_OK
jmp FREE_ERR

FREE_OK:
lea dx, FR_OK
jmp END_FREE
FREE_ERR:
lea dx, FR_ERR
END_FREE:
call WRITE

; вывод цепочки блоков управления памятью
mov ah, 52h
int 21h
mov ax, es:[bx-2]
mov es, ax
next_mcb:
    lea dx, NEW_LINE
    call WRITE
    lea dx, OWNER
    call WRITE
    mov bx, es:[1]
    ; switch owner
    lea dx, FREE
    cmp bx, 0000h
    je EQUAL
    lea dx, OSXMSUBM
    cmp bx, 0006h
    je EQUAL
    lea dx, TOP_MEM
    cmp bx, 0007h
    je EQUAL
    lea dx, MSDOS
    cmp bx, 0008h
    je EQUAL
    lea dx, TAKEN386
    cmp bx, 0FFFAh
    je EQUAL
    lea dx, BLOCKED386
    cmp bx, 0FFFDh
    je EQUAL
    lea dx, OWNED386
    cmp bx, 0FFFEh
    je EQUAL
    jmp NOT_EQUAL
EQUAL:
    call WRITE
    jmp MCB_SZ
NOT_EQUAL:
    lea di, OWNER_2
    add di, 6
    mov ax, bx
    call WRD_TO_HEX
    mov dx, di
    call WRITE
    lea dx, MCB_SIZE
MCB_SZ:

```



```

        mov ax, es:[3]
        mov bx, 10h
        mul bx
        lea si, MCB_SIZE
        add si, 13
        call WRD_TO_DEC
        lea dx, MCB_SIZE
        call WRITE
        ; last bytes
        lea dx, LAST_BYTES
        call WRITE
        mov cx, 8
        xor bx, bx
        mov ah, 2
next_b:
        mov dl, es:[bx+8h]
        int 21h
        inc bx
        loop next_b
        lea dx, NEW_LINE
        call WRITE
        mov al, es:[0]
        cmp al, 5ah
        je MCB_end

        mov ax, es:[3]
        mov bx, es
        add bx, ax
        inc bx
        mov es, bx
        jmp next_mcb
MCB_end:
        xor al, al
        mov ah, 4Ch
        int 21h
PR_END:
CODESEG ENDS
END START

```