

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

Крейсманн К.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

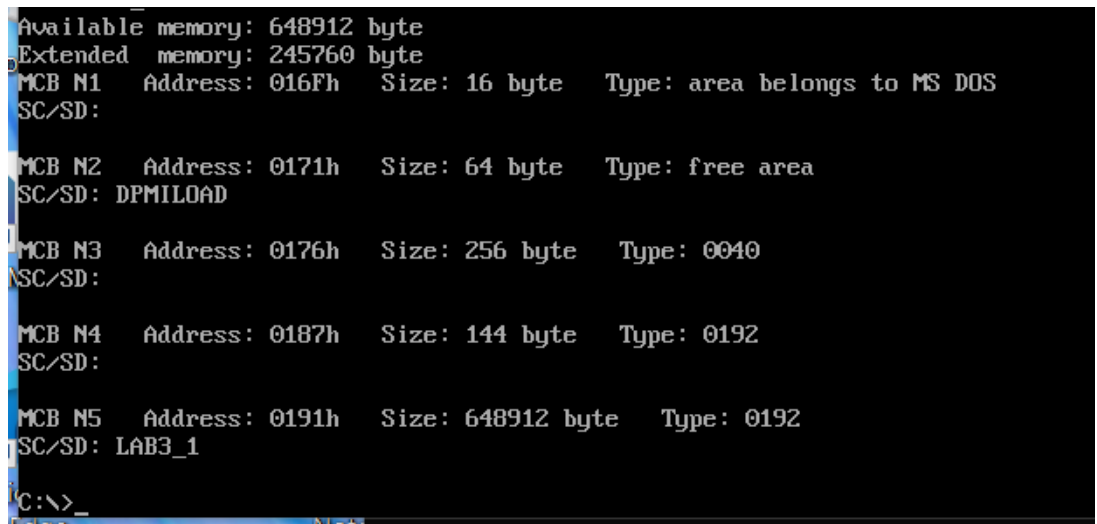
Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованный в компьютере и способ, принятый в ОС. В лабораторной работе рассматривается неограниченная память и способ управления динамическими разделами. Для реализации управления памятью в этом случае 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 функции управления памятью ядра операционной системы.

Выполнение работы

Шаг 1.

Была написана и отлажена программа типа .COM, которая выбирает и распечатывает следующую информацию:

- 1) Количество доступной памяти.
- 2) Размер расширенной памяти.
- 3) Выводит цепочку блоков управления памятью.



```
Available memory: 648912 byte
Extended memory: 245760 byte
MCB N1 Address: 016Fh Size: 16 byte Type: area belongs to MS DOS
SC/SD:
MCB N2 Address: 0171h Size: 64 byte Type: free area
SC/SD: DPMILOAD
MCB N3 Address: 0176h Size: 256 byte Type: 0040
SC/SD:
MCB N4 Address: 0187h Size: 144 byte Type: 0192
SC/SD:
MCB N5 Address: 0191h Size: 648912 byte Type: 0192
SC/SD: LAB3_1
C:\>_
```

Рисунок 1 – Пример работы программы №1

Шаг 2.

Программа была изменена так, чтобы она освобождала память, которую она не занимает.

```
Available memory: 648912 byte
Extended memory: 245760 byte
MCB N1 Address: 016Fh Size: 16 byte Type: area belongs to MS DOS
SC/SD:
MCB N2 Address: 0171h Size: 64 byte Type: free area
SC/SD: DPMILOAD
MCB N3 Address: 0176h Size: 256 byte Type: 0040
SC/SD:
MCB N4 Address: 0187h Size: 144 byte Type: 0192
SC/SD:
MCB N5 Address: 0191h Size: 1024 byte Type: 0192
SC/SD: LAB3_2
MCB N6 Address: 01D2h Size: 647872 byte Type: free area
SC/SD: or $
C:\>
```

Рисунок 2 - Пример работы программы №2

Шаг 3.

Программа была изменена так, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функций 48h прерывания 21h.

```
C:\>lab3_3
Available memory: 648912 byte
Extended memory: 245760 byte
MCB N1 Address: 016Fh Size: 16 byte Type: area belongs to MS DOS
SC/SD:
MCB N2 Address: 0171h Size: 64 byte Type: free area
SC/SD: DPMILOAD
MCB N3 Address: 0176h Size: 256 byte Type: 0040
SC/SD:
MCB N4 Address: 0187h Size: 144 byte Type: 0192
SC/SD:
MCB N5 Address: 0191h Size: 1072 byte Type: 0192
SC/SD: LAB3_3
MCB N6 Address: 01D5h Size: 65536 byte Type: 0192
SC/SD: LAB3_3
MCB N7 Address: 11D6h Size: 582272 byte Type: free area
SC/SD: í→Eú íU
C:\>
```

Рисунок 3 - Пример работы программы №3

Шаг 4.

Программа была изменена так, чтобы 64Кб запрашивались до освобождения памяти.

```
C:\>lab3_4.com
Available memory: 648912 byte
Extended memory: 245760 byte
Memory cannot be allocated!

MCB N1   Address: 016Fh   Size: 16 byte   Type: area belongs to MS DOS
SC/SD:

MCB N2   Address: 0171h   Size: 64 byte   Type: free area
SC/SD: DPMILOAD

MCB N3   Address: 0176h   Size: 256 byte   Type: 0040
SC/SD:

MCB N4   Address: 0187h   Size: 144 byte   Type: 0192
SC/SD:

MCB N5   Address: 0191h   Size: 1072 byte   Type: 0192
SC/SD: LAB3_4

MCB N6   Address: 01D5h   Size: 647824 byte   Type: free area
SC/SD: r initia

C:\>_
Not
```

Рисунок 4 - Пример работы программы №4

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ ПО ЛАБОРАТОРНОЙ №3

1) Что означает «доступный объем памяти»?

Часть оперативной памяти, которую может использовать программа.

2) Где МСВ блок Вашей памяти в списке?

Во всех кроме третьей программы – пятый по счету. В третьей программе – пятый и шестой.

3) Какой размер памяти занимает программа в каждом случае?

В первом случае : 648912 байт.

Во втором: 1024 байт.

В третьем: $1072 + 65536 = 66608$ байт.

В четвертом: 1072 байта .

Выводы.

Был исследован механизм управления память в ОС DOS.

Приложение А.

Исходный код программы lab3_1.asm:

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

;данные

AVAILABLE_MEMORY_STRING    db "Available memory: ",
    '$'
EXTENDED_MEMORY_STRING    db 0dh,0ah,"Extended memory: ",
    '$'
BYTE_STRING                db " byte",
    '$'
MCB_NUMBER_STRING          db 0dh,0ah, "MCB N",
    '$'
NEWLINE_STRING             db
    0dh,0ah,'$'
SPACE_STRING               db " "
ADDRESS_STRING             db " Address:      h",
    '$'
SIZE_STRING                db " Size: ",
    '$'
TYPE_STRING                db " Type: ",
    '$'
SCSD_STRING                db "SC/SD: " ,
    '$'
TYPE_0000_STRING           db "free area",
    0dh,0ah,'$'
TYPE_0006_STRING           db "area belongs OS XMS UMB driver",
    0dh,0ah,'$'
TYPE_0007_STRING           db "area of excluded upper driver memory",
    0dh,0ah,'$'
TYPE_0008_STRING           db "area belongs to MS DOS",
    0dh,0ah,'$'
TYPE_FFFA_STRING           db "area occupied by the control block 386MAAX
UMB",0dh,0ah,'$'
TYPE_FFFD_STRING           db "area is blocked by 386MAX",
    0dh,0ah,'$'
TYPE_FFFE_STRING           db "area belongs to 386MAX UMB",
    0dh,0ah,'$'
TYPE_EMPTY_STRING          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 преводится 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- числа, 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

WRITE_STR proc near
        push ax
        mov ah,9h
        int 21h
        pop ax
        ret
WRITE_STR ENDP

DD_DEC_PRINT proc near ;выводит число, записанное в (dx ax) в 10-ой с.с.
        push ax
        push dx
        mov bx,0ah          ;в bx 10 в 10-ой с.с.
        xor cx,cx           ;счетчик для количества цифр
metka1:
        div bx              ;теперь в ax - целая часть от деления (dx ax)/bx, в dx -
остаток
        push dx             ;сохраняем остаток от деления
        inc cx
        xor dx,dx           ;dx=0
        cmp ax,0            ;делим, пока есть что делить
        jne metka1

metka2:
        pop dx              ;вытаскиваем очередную цифру
        add dl,'0'          ;в символ
        mov ah,02h
        int 21h             ;вывод символа
        loop metka2

        pop dx
        pop ax
        ret
DD_DEC_PRINT ENDP

```



```

AVAILABLE_MEMORY proc near
    mov dx,offset AVAILABLE_MEMORY_STRING
    call WRITE_STR

    mov ah,4ah
    mov bx,0ffffh ;заведомо большой рзамер памяти
    int 21h        ;теперь в bx доступная память

    mov ax,bx
    mov bx,010h
    mul bx          ;теперь в (dx ax) количество байт
    call DD_DEC_PRINT

    mov dx,offset BYTE_STRING
    call WRITE_STR

    ret
AVAILABLE_MEMORY endp

EXTENDED_MEMORY proc near
    mov al,30h      ;запись адреса ячейки CMOS
    out 70h,al
    in al,71h       ;чтение младшего байта размера расширенной памяти
    mov bl,al
    mov al,31h      ;запись адреса ячейки CMOS
    out 70h,al
    in al,71h       ;чтение старшего байта

    mov bh,al
    mov ax,bx        ;в ax старший и младший байты
    mov dx, offset EXTENDED_MEMORY_STRING
    call WRITE_STR

    mov bx,010h
    mul bx           ;(dx ax) = ax*bx

    call DD_DEC_PRINT
    mov dx,offset BYTE_STRING
    call WRITE_STR

    ret
EXTENDED_MEMORY endp

MCB_LIST proc near
    mov ah,52h
    int 21h          ;теперь ES:BX указывает на список списков
    mov bx, ES:[BX-2] ;получаем адрес первого mcb
    mov es,bx
    mov cx,1
    push cx
metka3:
    mov dx,offset MCB_NUMBER_STRING
    call WRITE_STR
    xor dx,dx
    pop cx
    mov ax,cx
    inc cx
    push cx
    call DD_DEC_PRINT
    ;mov dx,offset NEWLINE_STRING
    ;call WRITE_STR

    mov di,offset ADDRESS_STRING
    add di,15

```

```

mov ax,es
call WRD_TO_HEX
mov dx,offset ADDRESS_STRING
call WRITE_STR

mov dx,offset SIZE_STRING
call WRITE_STR
mov ax,es:[3]
mov bx,010h
mul bx
call DD_DEC_PRINT
mov dx, offset BYTE_STRING
call WRITE_STR

mov dx,offset TYPE_STRING
call WRITE_STR

mov ax,es:[1]
cmp ax,00000h
jne metka4
mov dx,offset TYPE_0000_STRING
jmp metka10
metka4:
cmp ax,00006h
jne metka5
mov dx,offset TYPE_0006_STRING
jmp metka10
metka5:
cmp ax,00007h
jne metka6
mov dx,offset TYPE_0007_STRING
jmp metka10
metka6:
cmp ax,00008h
jne metka7
mov dx,offset TYPE_0008_STRING
jmp metka10
metka7:
cmp ax,0fffah
jne metka8
mov dx,offset TYPE_FFFA_STRING
jmp metka10
metka8:
cmp ax,0fffdh
jne metka9
mov dx,offset TYPE_FFFD_STRING
jmp metka10
metka9:
cmp ax,0fffeh
jne metka11
mov dx,offset TYPE_FFFE_STRING
jmp metka10
metka11:
mov di,offset TYPE_EMPTY_STRING
add di,3
call WRD_TO_HEX
mov dx,offset TYPE_EMPTY_STRING
metka10:
call WRITE_STR
mov di,0
metka12:
mov dx,offset SCSD_STRING
call WRITE_STR

```

;получаем размер участка в параграфах

;в (dx ax) размер участка в байтах

```

metka14:
    mov dl,es:[8+di]
    mov ah,02h
    int 21h
    inc di
    cmp di,8
    jne metka14

    mov ah,es:[0]
    push ax

    mov ax,es:[3]
    mov bx,es
    add bx,ax
    inc bx
    mov es,bx

    mov dx,offset NEWLINE_STRING
    call write_str

    pop ax
    cmp ah,05ah
    jne metka13
    jmp end_of_proc
metka13:
    jmp metka3

end_of_proc:
    pop cx
    ret
MCB_LIST ENDP

BEGIN:
    call AVAILABLE_MEMORY
    call EXTENDED_MEMORY
    call MCB_LIST
; Выход в DOS
    xor AL,AL
    mov AH,4Ch
    int 21H
TESTPC ENDS
END START ;конец модуля, START - точка входа

```

Приложение В.

Исходный код программы lab3_2.asm:

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

;данные

AVAILABLE_MEMORY_STRING    db "Available memory: ",
    '$'
EXTENDED_MEMORY_STRING     db 0dh,0ah,"Extended  memory: ",
    '$'
BYTE_STRING                db " byte",
    '$'
MCB_NUMBER_STRING          db 0dh,0ah, "MCB N",
    '$'
NEWLINE_STRING             db
    0dh,0ah,'$'
SPACE_STRING               db "  "
ADDRESS_STRING             db "    Address:      h",
    '$'
SIZE_STRING                db "    Size: ",
    '$'
TYPE_STRING               db "    Type: ",
    '$'
SCSD_STRING                db "SC/SD: "      ,
    '$'
TYPE_0000_STRING           db "free area",
    0dh,0ah,'$'
TYPE_0006_STRING           db "area belongs OS XMS UMB driver",
    0dh,0ah,'$'
TYPE_0007_STRING           db "area of excluded upper driver memory",
    0dh,0ah,'$'
TYPE_0008_STRING           db "area belongs to MS DOS",
    0dh,0ah,'$'
TYPE_FFFA_STRING           db "area occupied by the control block 386MAAX
UMB",0dh,0ah,'$'
TYPE_FFFD_STRING           db "area is blocked by 386MAX",
    0dh,0ah,'$'
TYPE_FFFE_STRING           db "area belongs to 386MAX UMB",
    0dh,0ah,'$'
TYPE_EMPTY_STRING          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 преводится 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- числа, 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

WRITE_STR proc near
        push ax
        mov ah,9h
        int 21h
        pop ax
        ret
WRITE_STR ENDP

DD_DEC_PRINT proc near ;выводит число, записанное в (dx ax) в 10-ой с.с.
        push ax
        push dx
        mov bx,0ah          ;в bx 10 в 10-ой с.с.
        xor cx,cx          ;счетчик для количества цифр
metka1:
        div bx              ;теперь в ax - целая часть от деления (dx ax)/bx, в dx -
остаток
        push dx              ;сохраняем остаток от деления
        inc cx
        xor dx,dx            ;dx=0
        cmp ax,0            ;делим, пока есть что делить
        jne metka1

metka2:
        pop dx              ;вытаскиваем очередную цифру
        add dl,'0'          ;в символ
        mov ah,02h
        int 21h              ;вывод символа
        loop metka2

        pop dx
        pop ax
        ret
DD_DEC_PRINT ENDP

```

```

AVAILABLE_MEMORY proc near
    mov dx,offset AVAILABLE_MEMORY_STRING
    call WRITE_STR

    mov ah,4ah
    mov bx,0ffffh ;заведомо большой рзамер памяти
    int 21h        ;теперь в bx доступная память

    mov ax,bx
    mov bx,010h
    mul bx          ;теперь в (dx ax) количество байт
    call DD_DEC_PRINT

    mov dx,offset BYTE_STRING
    call WRITE_STR

    ret
AVAILABLE_MEMORY endp

EXTENDED_MEMORY proc near
    mov al,30h          ;запись адреса ячейки CMOS
    out 70h,al
    in al,71h           ;чтение младшего байта размера расширенной памяти
    mov bl,al
    mov al,31h          ;запись адреса ячейки CMOS
    out 70h,al
    in al,71h           ;чтение старшего байта

    mov bh,al
    mov ax,bx            ;в ax старший и младший байты
    mov dx, offset EXTENDED_MEMORY_STRING
    call WRITE_STR

    mov bx,010h
    mul bx               ;(dx ax) = ax*bx

    call DD_DEC_PRINT
    mov dx,offset BYTE_STRING
    call WRITE_STR

    ret
EXTENDED_MEMORY endp

MCB_LIST proc near
    mov ah,52h
    int 21h              ;теперь ES:BX указывает на список списков
    mov bx, ES:[BX-2]    ;получаем адрес первого mcb
    mov es,bx
    mov cx,1
    push cx
metka3:
    mov dx,offset MCB_NUMBER_STRING
    call WRITE_STR
    xor dx,dx
    pop cx
    mov ax,cx
    inc cx
    push cx
    call DD_DEC_PRINT
    ;mov dx,offset NEWLINE_STRING
    ;call WRITE_STR

    mov di,offset ADDRESS_STRING
    add di,15

```

```

mov ax,es
call WRD_TO_HEX
mov dx,offset ADDRESS_STRING
call WRITE_STR

mov dx,offset SIZE_STRING
call WRITE_STR
mov ax,es:[3]
mov bx,010h
mul bx
call DD_DEC_PRINT
mov dx, offset BYTE_STRING
call WRITE_STR

mov dx,offset TYPE_STRING
call WRITE_STR

mov ax,es:[1]
cmp ax,00000h
jne metka4
mov dx,offset TYPE_0000_STRING
jmp metka10
metka4:
cmp ax,00006h
jne metka5
mov dx,offset TYPE_0006_STRING
jmp metka10
metka5:
cmp ax,00007h
jne metka6
mov dx,offset TYPE_0007_STRING
jmp metka10
metka6:
cmp ax,00008h
jne metka7
mov dx,offset TYPE_0008_STRING
jmp metka10
metka7:
cmp ax,0fffah
jne metka8
mov dx,offset TYPE_FFFA_STRING
jmp metka10
metka8:
cmp ax,0fffdh
jne metka9
mov dx,offset TYPE_FFFD_STRING
jmp metka10
metka9:
cmp ax,0fffeh
jne metka11
mov dx,offset TYPE_FFFE_STRING
jmp metka10
metka11:
mov di,offset TYPE_EMPTY_STRING
add di,3
call WRD_TO_HEX
mov dx,offset TYPE_EMPTY_STRING
metka10:
call WRITE_STR
mov di,0
metka12:
mov dx,offset SCSD_STRING
call WRITE_STR

```

;получаем размер участка в параграфах

;в (dx ax) размер участка в байтах

```

metka14:
    mov dl,es:[8+di]
    mov ah,02h
    int 21h
    inc di
    cmp di,8
    jne metka14

    mov ah,es:[0]
    push ax

    mov ax,es:[3]
    mov bx,es
    add bx,ax
    inc bx
    mov es,bx

    mov dx,offset NEWLINE_STRING
    call write_str

    pop ax
    cmp ah,05ah
    jne metka13
    jmp end_of_proc
metka13:
    jmp metka3

end_of_proc:
    pop cx
    ret
MCB_LIST ENDP

```

```

FREE proc near
    mov ax, offset endofcode
    mov bx, 10h
    mov dx,0
    div bx
    mov bx,ax
    inc bx
    mov ah,4ah
    int 21h
    ret
FREE ENDP

```

```

BEGIN:
    call AVAILABLE_MEMORY
    call EXTENDED_MEMORY

    call FREE
    call MCB_LIST

; Выход в DOS
    xor AL,AL
    mov AH,4Ch
    int 21H

```


endOfCode:

TESTPC ENDS

END START ;конец модуля, START - точка входа

Приложение С.

Исходный код программы lab3_3.asm:

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

;данные

AVAILABLE_MEMORY_STRING    db "Available memory: ",
                             '$'
EXTENDED_MEMORY_STRING     db 0dh,0ah,"Extended memory: ",
                             '$'
BYTE_STRING                db " byte",
                             '$'
MCB_NUMBER_STRING          db 0dh,0ah, "MCB N",
                             '$'
NEWLINE_STRING             db
                             0dh,0ah,'$'
SPACE_STRING               db " "
ADDRESS_STRING              db " Address:      h",
                             '$'
SIZE_STRING                db " Size: ",
                             '$'
TYPE_STRING                db " Type: ",
                             '$'
SCSD_STRING                db "SC/SD: " ,
                             '$'
TYPE_0000_STRING           db "free area",
                             0dh,0ah,'$'
TYPE_0006_STRING           db "area belongs OS XMS UMB driver",
                             0dh,0ah,'$'
TYPE_0007_STRING           db "area of excluded upper driver memory",
                             0dh,0ah,'$'
TYPE_0008_STRING           db "area belongs to MS DOS",
                             0dh,0ah,'$'
TYPE_FFFA_STRING           db "area occupied by the control block 386MAAX
UMB",0dh,0ah,'$'
TYPE_FFFD_STRING           db "area is blocked by 386MAX",
                             0dh,0ah,'$'
TYPE_FFFE_STRING           db "area belongs to 386MAX UMB",
                             0dh,0ah,'$'
TYPE_EMPTY_STRING          db " ",
                             0dh,0ah,'$'
ERROR_MEMORY_STRING        db 0dh,0ah,"Memory cannot be allocated!",
                             0dh,0ah,'$'

;.....

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 преобразуется 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- числа, 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

WRITE_STR proc near
        push ax
        mov ah,9h
        int 21h
        pop ax
        ret
WRITE_STR ENDP

DD_DEC_PRINT proc near ;выводит число, записанное в (dx ax) в 10-ой с.с.
        push ax
        push dx
        mov bx,0ah          ;в bx 10 в 10-ой с.с.
        xor cx,cx           ;счетчик для количества цифр
metka1:
        div bx              ;теперь в ax - целая часть от деления (dx ax)/bx, в dx -
остаток
        push dx             ;сохраняем остаток от деления
        inc cx
        xor dx,dx           ;dx=0
        cmp ax,0           ;делим, пока есть что делить
        jne metka1

metka2:
        pop dx              ;вытаскиваем очередную цифру
        add dl,'0'          ;в символ
        mov ah,02h
        int 21h             ;вывод символа
        loop metka2

        pop dx
        pop ax
        ret
DD_DEC_PRINT ENDP

```

```

AVAILABLE_MEMORY proc near
    mov dx,offset AVAILABLE_MEMORY_STRING
    call WRITE_STR

    mov ah,4ah
    mov bx,0ffffh ;заведомо большой рзамер памяти
    int 21h        ;теперь в bx доступная память

    mov ax,bx
    mov bx,010h
    mul bx          ;теперь в (dx ax) количество байт
    call DD_DEC_PRINT

    mov dx,offset BYTE_STRING
    call WRITE_STR

    ret
AVAILABLE_MEMORY endp

EXTENDED_MEMORY proc near
    mov al,30h          ;запись адреса ячейки CMOS
    out 70h,al
    in al,71h           ;чтение младшего байта размера расширенной памяти
    mov bl,al
    mov al,31h          ;запись адреса ячейки CMOS
    out 70h,al
    in al,71h           ;чтение старшего байта

    mov bh,al
    mov ax,bx            ;в ax старший и младший байты
    mov dx, offset EXTENDED_MEMORY_STRING
    call WRITE_STR

    mov bx,010h
    mul bx               ;(dx ax) = ax*bx

    call DD_DEC_PRINT
    mov dx,offset BYTE_STRING
    call WRITE_STR

    ret
EXTENDED_MEMORY endp

MCB_LIST proc near
    mov ah,52h
    int 21h              ;теперь ES:BX указывает на список списков
    mov bx, ES:[BX-2]    ;получаем адрес первого mcb
    mov es,bx
    mov cx,1
    push cx
metka3:
    mov dx,offset MCB_NUMBER_STRING
    call WRITE_STR
    xor dx,dx
    pop cx
    mov ax,cx
    inc cx
    push cx
    call DD_DEC_PRINT
    ;mov dx,offset NEWLINE_STRING
    ;call WRITE_STR

    mov di,offset ADDRESS_STRING
    add di,15

```

```

mov ax,es
call WRD_TO_HEX
mov dx,offset ADDRESS_STRING
call WRITE_STR

mov dx,offset SIZE_STRING
call WRITE_STR
mov ax,es:[3]
mov bx,010h
mul bx
call DD_DEC_PRINT
mov dx, offset BYTE_STRING
call WRITE_STR

mov dx,offset TYPE_STRING
call WRITE_STR

mov ax,es:[1]
cmp ax,00000h
jne metka4
mov dx,offset TYPE_0000_STRING
jmp metka10
metka4:
cmp ax,00006h
jne metka5
mov dx,offset TYPE_0006_STRING
jmp metka10
metka5:
cmp ax,00007h
jne metka6
mov dx,offset TYPE_0007_STRING
jmp metka10
metka6:
cmp ax,00008h
jne metka7
mov dx,offset TYPE_0008_STRING
jmp metka10
metka7:
cmp ax,0fffah
jne metka8
mov dx,offset TYPE_FFFA_STRING
jmp metka10
metka8:
cmp ax,0fffdh
jne metka9
mov dx,offset TYPE_FFFD_STRING
jmp metka10
metka9:
cmp ax,0fffeh
jne metka11
mov dx,offset TYPE_FFFE_STRING
jmp metka10
metka11:
mov di,offset TYPE_EMPTY_STRING
add di,3
call WRD_TO_HEX
mov dx,offset TYPE_EMPTY_STRING
metka10:
call WRITE_STR
mov di,0
metka12:
mov dx,offset SCSD_STRING
call WRITE_STR

```

;получаем размер участка в параграфах

;в (dx ax) размер участка в байтах

```

metka14:
    mov dl,es:[8+di]
    mov ah,02h
    int 21h
    inc di
    cmp di,8
    jne metka14

    mov ah,es:[0]
    push ax

    mov ax,es:[3]
    mov bx,es
    add bx,ax
    inc bx
    mov es,bx

    mov dx,offset NEWLINE_STRING
    call write_str

    pop ax
    cmp ah,05ah
    jne metka13
    jmp end_of_proc
metka13:
    jmp metka3

end_of_proc:
    pop cx
    ret
MCB_LIST ENDP

FREE proc near
    mov ax, offset endofcode
    mov bx, 10h
    mov dx,0
    div bx
    mov bx,ax
    inc bx
    mov ah,4ah
    int 21h

    ret
FREE ENDP

memory_request proc near
    mov bx,1000h ;2^12 параграфов
    mov ah,48h
    int 21h
    jnae metkaBAD_mem ;CF=1
    jmp METKAOK_MEM
metkaBAD_mem:
    mov dx,offset ERROR_MEMORY_STRING
    call WRITE_STR
metkaOK_MEM:
    ret
MEMORY_REQUEST ENDP

```

```

BEGIN:
    call AVAILABLE_MEMORY
    call EXTENDED_MEMORY
    call FREE
    call MEMORY_REQUEST
    call MCB_LIST

; Выход в DOS
    xor AL,AL
    mov AH,4Ch
    int 21H

    endOfCode:

TESTPC ENDS
END START ;конец модуля, START - точка входа

```

Приложение D.

Исходный код программы lab3_4.asm:

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

;данные

AVAILABLE_MEMORY_STRING    db "Available memory: ",
                             '$'
EXTENDED_MEMORY_STRING     db 0dh,0ah,"Extended memory: ",
                             '$'
BYTE_STRING                db " byte",
                             '$'
MCB_NUMBER_STRING          db 0dh,0ah, "MCB N",
                             '$'
NEWLINE_STRING             db
                             0dh,0ah,'$'
SPACE_STRING               db " "
ADDRESS_STRING              db " Address:      h",
                             '$'
SIZE_STRING                db " Size: ",
                             '$'
TYPE_STRING                db " Type: ",
                             '$'
SCSD_STRING                db "SC/SD: " ,
                             '$'
TYPE_0000_STRING           db "free area",
                             0dh,0ah,'$'
TYPE_0006_STRING           db "area belongs OS XMS UMB driver",
                             0dh,0ah,'$'
TYPE_0007_STRING           db "area of excluded upper driver memory",
                             0dh,0ah,'$'
TYPE_0008_STRING           db "area belongs to MS DOS",
                             0dh,0ah,'$'
TYPE_FFFA_STRING           db "area occupied by the control block 386MAAX
UMB",0dh,0ah,'$'
TYPE_FFFD_STRING           db "area is blocked by 386MAX",
                             0dh,0ah,'$'
TYPE_FFFE_STRING           db "area belongs to 386MAX UMB",
                             0dh,0ah,'$'
TYPE_EMPTY_STRING          db " ",
                             0dh,0ah,'$'
ERROR_MEMORY_STRING        db 0dh,0ah,"Memory cannot be allocated!",
                             0dh,0ah,'$'

;.....

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 преводится 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- числа, 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

WRITE_STR proc near
        push ax
        mov ah,9h
        int 21h
        pop ax
        ret
WRITE_STR ENDP

DD_DEC_PRINT proc near ;выводит число, записанное в (dx ax) в 10-ой с.с.
        push ax
        push dx
        mov bx,0ah          ;в bx 10 в 10-ой с.с.
        xor cx,cx           ;счетчик для количества цифр
metka1:
        div bx              ;теперь в ax - целая часть от деления (dx ax)/bx, в dx -
остаток
        push dx             ;сохраняем остаток от деления
        inc cx
        xor dx,dx           ;dx=0
        cmp ax,0           ;делим, пока есть что делить
        jne metka1

metka2:
        pop dx              ;вытаскиваем очередную цифру
        add dl,'0'          ;в символ
        mov ah,02h
        int 21h             ;вывод символа
        loop metka2

        pop dx
        pop ax
        ret
DD_DEC_PRINT ENDP

```

```

AVAILABLE_MEMORY proc near
    mov dx,offset AVAILABLE_MEMORY_STRING
    call WRITE_STR

    mov ah,4ah
    mov bx,0ffffh ;заведомо большой рзамер памяти
    int 21h        ;теперь в bx доступная память

    mov ax,bx
    mov bx,010h
    mul bx          ;теперь в (dx ax) количество байт
    call DD_DEC_PRINT

    mov dx,offset BYTE_STRING
    call WRITE_STR

    ret
AVAILABLE_MEMORY endp

EXTENDED_MEMORY proc near
    mov al,30h          ;запись адреса ячейки CMOS
    out 70h,al
    in al,71h           ;чтение младшего байта размера расширенной памяти
    mov bl,al
    mov al,31h          ;запись адреса ячейки CMOS
    out 70h,al
    in al,71h           ;чтение старшего байта

    mov bh,al
    mov ax,bx            ;в ax старший и младший байты
    mov dx, offset EXTENDED_MEMORY_STRING
    call WRITE_STR

    mov bx,010h
    mul bx               ;(dx ax) = ax*bx

    call DD_DEC_PRINT
    mov dx,offset BYTE_STRING
    call WRITE_STR

    ret
EXTENDED_MEMORY endp

MCB_LIST proc near
    mov ah,52h
    int 21h              ;теперь ES:BX указывает на список списков
    mov bx, ES:[BX-2]    ;получаем адрес первого mcb
    mov es,bx
    mov cx,1
    push cx
metka3:
    mov dx,offset MCB_NUMBER_STRING
    call WRITE_STR
    xor dx,dx
    pop cx
    mov ax,cx
    inc cx
    push cx
    call DD_DEC_PRINT
    ;mov dx,offset NEWLINE_STRING
    ;call WRITE_STR

    mov di,offset ADDRESS_STRING
    add di,15

```

```

mov ax,es
call WRD_TO_HEX
mov dx,offset ADDRESS_STRING
call WRITE_STR

mov dx,offset SIZE_STRING
call WRITE_STR
mov ax,es:[3]
mov bx,010h
mul bx
call DD_DEC_PRINT
mov dx, offset BYTE_STRING
call WRITE_STR

mov dx,offset TYPE_STRING
call WRITE_STR

mov ax,es:[1]
cmp ax,00000h
jne metka4
mov dx,offset TYPE_0000_STRING
jmp metka10
metka4:
cmp ax,00006h
jne metka5
mov dx,offset TYPE_0006_STRING
jmp metka10
metka5:
cmp ax,00007h
jne metka6
mov dx,offset TYPE_0007_STRING
jmp metka10
metka6:
cmp ax,00008h
jne metka7
mov dx,offset TYPE_0008_STRING
jmp metka10
metka7:
cmp ax,0fffah
jne metka8
mov dx,offset TYPE_FFFA_STRING
jmp metka10
metka8:
cmp ax,0fffdh
jne metka9
mov dx,offset TYPE_FFFD_STRING
jmp metka10
metka9:
cmp ax,0fffeh
jne metka11
mov dx,offset TYPE_FFFE_STRING
jmp metka10
metka11:
mov di,offset TYPE_EMPTY_STRING
add di,3
call WRD_TO_HEX
mov dx,offset TYPE_EMPTY_STRING
metka10:
call WRITE_STR
mov di,0
metka12:
mov dx,offset SCSD_STRING
call WRITE_STR

```

;получаем размер участка в параграфах

;в (dx ax) размер участка в байтах

```

metka14:
    mov dl,es:[8+di]
    mov ah,02h
    int 21h
    inc di
    cmp di,8
    jne metka14

    mov ah,es:[0]
    push ax

    mov ax,es:[3]
    mov bx,es
    add bx,ax
    inc bx
    mov es,bx

    mov dx,offset NEWLINE_STRING
    call write_str

    pop ax
    cmp ah,05ah
    jne metka13
    jmp end_of_proc
metka13:
    jmp metka3

end_of_proc:
    pop cx
    ret
MCB_LIST ENDP

FREE proc near
    mov ax, offset endofcode
    mov bx, 10h
    mov dx,0
    div bx
    mov bx,ax
    inc bx
    mov ah,4ah
    int 21h

    ret
FREE ENDP

memory_request proc near
    mov bx,1000h ;2^12 paragraphs
    mov ah,48h
    int 21h
    jnae metkaBAD_mem ;CF=1
    jmp METKAOK_MEM
metkaBAD_mem:
    mov dx,offset ERROR_MEMORY_STRING
    call WRITE_STR
metkaOK_MEM:
    ret
MEMORY_REQUEST ENDP

```

```

BEGIN:
    call AVAILABLE_MEMORY
    call EXTENDED_MEMORY
    call MEMORY_REQUEST
    call FREE
    call MCB_LIST

; Выход в DOS
    xor AL,AL
    mov AH,4Ch
    int 21H

    endOfCode:

TESTPC ENDS
END START ;конец модуля, START - точка входа

```