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. 8382 _____ Черницын П.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследование структур данных и работы функций управления памятью ядра операционной системы.

Выполнение работы.

Был написан программный модуль типа **.COM** (представлен в приложении А), который определяет и распечатывает следующую информацию:

1) Количество доступной памяти.

2) Размер расширенной памяти.

3) Выводит цепочку блоков управления памятью.

Результат исполнения COM модуля представлен на рисунке 1.

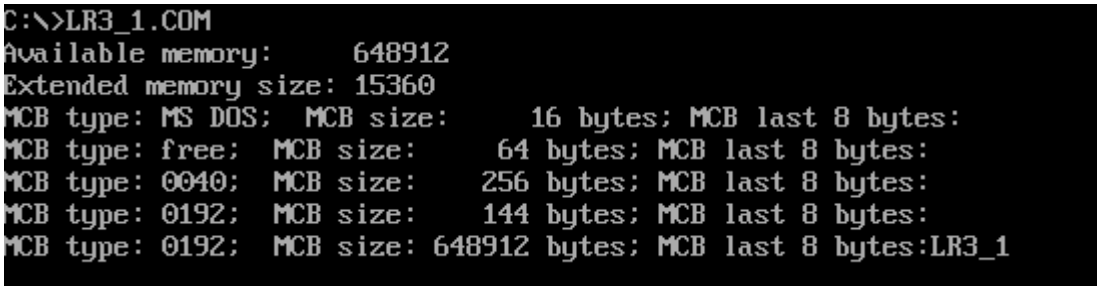

Рисунок 1 — результат исполнения первого COM модуля

Далее программа была изменена таким образом, чтобы она освобождала память, которую она не занимает. Исходный код представлен в приложении Б, а результат исполнения — на рисунке 2.

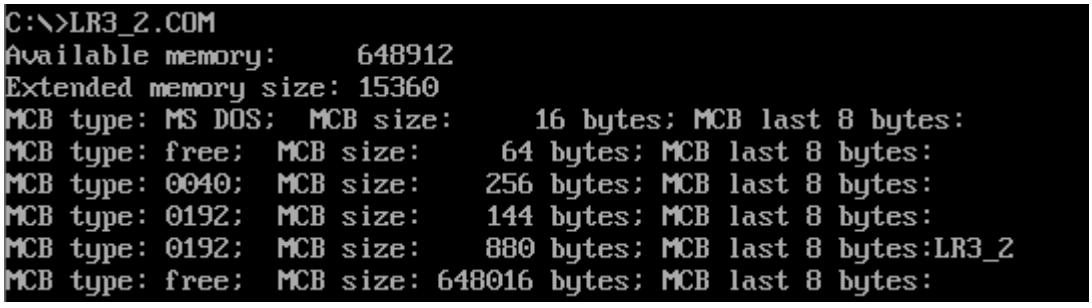

Рисунок 2 — результат исполнения второго COM модуля

Далее программа была изменена таким образом, чтобы после освобождения памяти программа запрашивала 64Кб памяти. Исходный код представлен в приложении В, а результат исполнения — на рисунке 3.

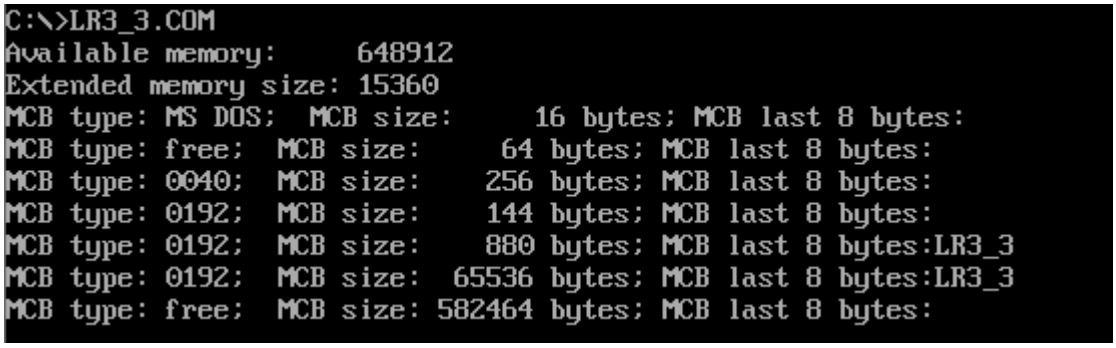

Рисунок 3 — результат исполнения третьего COM модуля

Далее программа была изменена таким образом, чтобы память запрашивалась до освобождения. Исходный код представлен в приложении Г, а результат исполнения — на рисунке 4.

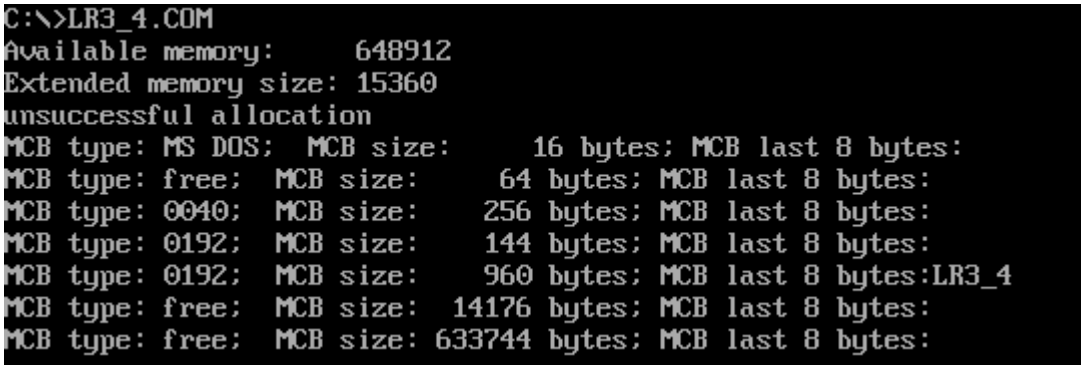

Рисунок 4 — результат исполнения четвертого COM модуля

Контрольные вопросы.

1) Что означает «доступный объём памяти»?

Это максимальный размер памяти, который может быть выделен программе.

2) Где MCB блок Вашей программы в списке?

Как видно из рис. 1-4, в первой, второй и четвёртой программах, блоками программы являются четвертый и пятый блоки. В третьей программе – блоки с

третьего по шестой.

3) Какой размер занимает программа в каждом случае?

Первая — занимает 649056 байта.

Вторая — занимает 1024 байта.

Третья — занимает 1024 байта + выделенные 64Кб.

Четвёртая — занимает 1104 байта.

Вывод.

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

ПРИЛОЖЕНИЕ А

СОДЕРЖИМОЕ ФАЙЛА LAB3_1.ASM

```
TESTPC SEGMENT
        ASSUME cs:TESTPC, ds:TESTPC, es:NOTHING, ss:NOTHING
        ORG 100H
START: jmp BEGIN
;DATA
AVAILABLE_MEM           db 'Available memory:           ',0DH,0AH,'$'
EXTENDED_MEM_SIZE       db 'Extended memory size:                ',0DH,0AH,'$'
MCB_TYPE                db 'MCB type: ','$'
MCB_SIZE                db ';  MCB size:        bytes; ','$'
MCB_LAST_BYTE           db 'MCB last 8 bytes:','$'
STR_FREE                db 'free','$'
STR_OSXMSUBM            db 'OS XMS UMB','$'
STR_TOP_MEM             db 'drivers top memory','$'
STR_MSDOS               db 'MS DOS','$'
STR_TAKEN386            db '386MAX UMBs block','$'
STR_BLOCKED386          db 'blocked by 386MAX','$'
STR_OWNED386            db '386MAX UMB','$'
ENDL                    db 0DH,0AH, '$'
;-------------------------------------------------------
WriteMsg PROC near
            push ax
        mov ah,09h
        int 21h
            pop ax
        ret
WriteMsg 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 в шест. сс ax
        push cx
        mov ah,al
        call TETR_TO_HEX
        xchg al,ah
        mov cl,4
        shr al,cl
        call TETR_TO_HEX ;al - старшая
        pop cx ;ah - младшая цифра
        ret
BYTE_TO_HEX ENDP
;-------------------------------------------------------
PRINT_BYTE      PROC near
; prints AL as two hex digits
      push BX
      push DX

      call BYTE_TO_HEX
      mov BH, AH

      mov DL, AL
      mov AH, 02h
      int 21h

      mov DL, BH
      mov AH, 02h
      int 21h
```

```
      pop DX
      pop BX
      ret
PRINT_BYTE    ENDP
;-------------------------------------------------------
WRD_TO_HEX PROC near
; перевод в 16сс 16ти разрядного числа
; 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 байта в 10сс, si - адрес поля младшей цифры
; al содержит исходный байт
        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
;ax содержит исходное слово
;si адрес поля младшей цифры
            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
            jne loop_wd
end_l1:
            pop dx
            pop cx
            pop ax
            ret
WRD_TO_DEC ENDP
;-------------------------------------------------------
BEGIN:
            ;Available memory
```

```
      mov bx, 0ffffh
      mov ah, 4Ah
int 21h
mov ax, bx
mov cx, 10h
mul cx
lea  si, AVAILABLE_MEM + 27
call WRD_TO_DEC
      lea dx, AVAILABLE_MEM
      call WriteMsg

      ;Extended memory size
      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_MEM_SIZE + 26
      call WRD_TO_DEC
      lea dx, EXTENDED_MEM_SIZE
      call WriteMsg

      ;MCB info
      ; get first mcb's address
      mov AH, 52h
      int 21h
      mov AX, ES:[BX-2]
      mov ES, AX

      MCB_LOOP:
                  lea dx, MCB_TYPE
                  call WriteMsg

                  ; get owner
                  mov BX, ES:[1h]

                  lea dx, STR_FREE
                  cmp bx, 0000h
                  je MCB_MATCH
                  lea dx, STR_OSXMSUBM
                  cmp bx, 0006h
                  je MCB_MATCH
                  lea dx, STR_TOP_MEM
                  cmp bx, 0007h
                  je MCB_MATCH
                  lea dx, STR_MSDOS
                  cmp bx, 0008h
                  je MCB_MATCH
                  lea dx, STR_TAKEN386
                  cmp bx, 0FFFAh
                  je MCB_MATCH
                  lea dx, STR_BLOCKED386
                  cmp bx, 0FFFDh
                  je MCB_MATCH
                  lea dx, STR_OWNED386
                  cmp bx, 0FFFEh
                  je MCB_MATCH
                  jmp MCB_NOT_MATCH

                  MCB_NOT_MATCH:
                        mov AL, BH
                        call PRINT_BYTE
                        mov AL, BL
                        call PRINT_BYTE
                        jmp MCB_MATCH_END

                  MCB_MATCH:
```

```
                call WriteMsg

MCB_MATCH_END:

; get size
mov ax, es:[3h]
mov bx, 10h
mul bx

lea si, MCB_SIZE
add si, 18
call WRD_TO_DEC
lea dx, MCB_SIZE
call WriteMsg

;print last 8 bytes
lea dx, MCB_LAST_BYTE
call WriteMsg

mov cx, 8
mov bx, 8h
mov ah, 02h

PRINT_BYTE_LOOP:
            mov dl, es:[bx]
            int 21h
            inc bx
            loop PRINT_BYTE_LOOP
lea dx, ENDL
call WriteMsg

;check last block
mov al, es:[0h]
cmp al, 5Ah
je _END

;get new block
mov ax, es:[3h]
mov bx, es
add bx, ax
inc bx
mov es, bx
jmp MCB_LOOP


_END:
            ret

TESTPC ENDS
            END START
```

ПРИЛОЖЕНИЕ Б

СОДЕРЖИМОЕ ФАЙЛА LAB3_2.ASM

```
TESTPC SEGMENT
        ASSUME cs:TESTPC, ds:TESTPC, es:NOTHING, ss:NOTHING
        ORG 100H
START: jmp BEGIN
;DATA
AVAILABLE_MEM           db 'Available memory:           ',0DH,0AH,'$'
EXTENDED_MEM_SIZE       db 'Extended memory size:                ',0DH,0AH,'$'
MCB_TYPE                db 'MCB type: ','$'
MCB_SIZE                db ';  MCB size:        bytes; ','$'
MCB_LAST_BYTE           db 'MCB last 8 bytes:','$'
STR_FREE                db 'free','$'
STR_OSXMSUBM            db 'OS XMS UMB','$'
STR_TOP_MEM             db 'drivers top memory','$'
STR_MSDOS               db 'MS DOS','$'
STR_TAKEN386            db '386MAX UMBs block','$'
STR_BLOCKED386          db 'blocked by 386MAX','$'
STR_OWNED386            db '386MAX UMB','$'
ENDL                    db 0DH,0AH, '$'
;--------------------------------------------------
WriteMsg PROC near
            push ax
        mov ah,09h
        int 21h
            pop ax
        ret
WriteMsg 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 в шест. сс ax
        push cx
        mov ah,al
        call TETR_TO_HEX
        xchg al,ah
        mov cl,4
        shr al,cl
        call TETR_TO_HEX ;al - старшая
        pop cx ;ah - младшая цифра
        ret
BYTE_TO_HEX ENDP
;--------------------------------------------------
PRINT_BYTE      PROC near
; prints AL as two hex digits
      push BX
      push DX

      call BYTE_TO_HEX
      mov BH, AH

      mov DL, AL
      mov AH, 02h
      int 21h

      mov DL, BH
      mov AH, 02h
      int 21h
```

```
      pop DX
      pop BX
      ret
PRINT_BYTE    ENDP
;-------------------------------------------------------
WRD_TO_HEX PROC near
; перевод в 16сс 16ти разрядного числа
; 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 байта в 10сс, si - адрес поля младшей цифры
; al содержит исходный байт
        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
;ax содержит исходное слово
;si адрес поля младшей цифры
            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
            jne loop_wd
end_l1:
            pop dx
            pop cx
            pop ax
            ret
WRD_TO_DEC ENDP
;-------------------------------------------------------
BEGIN:
            ;Available memory
```

```
      mov bx, 0ffffh
      mov ah, 4Ah
int 21h
mov ax, bx
mov cx, 10h
mul cx
lea  si, AVAILABLE_MEM + 27
call WRD_TO_DEC
      lea dx, AVAILABLE_MEM
      call WriteMsg

      ;Extended memory size
      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_MEM_SIZE + 26
      call WRD_TO_DEC
      lea dx, EXTENDED_MEM_SIZE
      call WriteMsg

      lea bx, _END
      add bx, 10h
      shr bx, 1
      shr bx, 1
      shr bx, 1
      shr bx, 1 ;Делим на 16, чтобы получить параграфы

      mov ah, 4Ah
      int 21h

      ;MCB info
      ; get first mcb's address
      mov AH, 52h
      int 21h
      mov AX, ES:[BX-2]
      mov ES, AX

      MCB_LOOP:
                  lea dx, MCB_TYPE
                  call WriteMsg

                  ; get owner
                  mov BX, ES:[1h]

                  lea dx, STR_FREE
                  cmp bx, 0000h
                  je MCB_MATCH
                  lea dx, STR_OSXMSUBM
                  cmp bx, 0006h
                  je MCB_MATCH
                  lea dx, STR_TOP_MEM
                  cmp bx, 0007h
                  je MCB_MATCH
                  lea dx, STR_MSDOS
                  cmp bx, 0008h
                  je MCB_MATCH
                  lea dx, STR_TAKEN386
                  cmp bx, 0FFFAh
                  je MCB_MATCH
                  lea dx, STR_BLOCKED386
                  cmp bx, 0FFFDh
                  je MCB_MATCH
                  lea dx, STR_OWNED386
                  cmp bx, 0FFFEh
                  je MCB_MATCH
```

```
                        jmp MCB_NOT_MATCH

                        MCB_NOT_MATCH:
                              mov AL, BH
                              call PRINT_BYTE
                              mov AL, BL
                              call PRINT_BYTE
                              jmp MCB_MATCH_END

                        MCB_MATCH:
                              call WriteMsg

                        MCB_MATCH_END:

                        ; get size
                        mov ax, es:[3h]
                        mov bx, 10h
                        mul bx

                        lea si, MCB_SIZE
                        add si, 18
                        call WRD_TO_DEC
                        lea dx, MCB_SIZE
                        call WriteMsg

                        ;print last 8 bytes
                        lea dx, MCB_LAST_BYTE
                        call WriteMsg

                        mov cx, 8
                        mov bx, 8h
                        mov ah, 02h

                        PRINT_BYTE_LOOP:
                                    mov dl, es:[bx]
                                    int 21h
                                    inc bx
                                    loop PRINT_BYTE_LOOP
                        lea dx, ENDL
                        call WriteMsg

                        ;check last block
                        mov al, es:[0h]
                        cmp al, 5Ah
                        je _END

                        ;get new block
                        mov ax, es:[3h]
                        mov bx, es
                        add bx, ax
                        inc bx
                        mov es, bx
                        jmp MCB_LOOP


_END:
                    ret

TESTPC ENDS
                    END START
```

ПРИЛОЖЕНИЕ В

СОДЕРЖИМОЕ ФАЙЛА LAB3_3.ASM

```
TESTPC SEGMENT
        ASSUME cs:TESTPC, ds:TESTPC, es:NOTHING, ss:NOTHING
        ORG 100H
START: jmp BEGIN
;DATA
AVAILABLE_MEM           db 'Available memory:           ',0DH,0AH,'$'
EXTENDED_MEM_SIZE       db 'Extended memory size:                ',0DH,0AH,'$'
MCB_TYPE                db 'MCB type: ','$'
MCB_SIZE                db ';  MCB size:        bytes; ','$'
MCB_LAST_BYTE           db 'MCB last 8 bytes:','$'
STR_FREE                db 'free','$'
STR_OSXMSUBM            db 'OS XMS UMB','$'
STR_TOP_MEM             db 'drivers top memory','$'
STR_MSDOS               db 'MS DOS','$'
STR_TAKEN386            db '386MAX UMBs block','$'
STR_BLOCKED386          db 'blocked by 386MAX','$'
STR_OWNED386            db '386MAX UMB','$'
ENDL                    db 0DH,0AH, '$'
;--------------------------------------------------
WriteMsg PROC near
            push ax
        mov ah,09h
        int 21h
            pop ax
        ret
WriteMsg 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 в шест. сс ax
        push cx
        mov ah,al
        call TETR_TO_HEX
        xchg al,ah
        mov cl,4
        shr al,cl
        call TETR_TO_HEX ;al - старшая
        pop cx ;ah - младшая цифра
        ret
BYTE_TO_HEX ENDP
;--------------------------------------------------
PRINT_BYTE      PROC near
; prints AL as two hex digits
      push BX
      push DX

      call BYTE_TO_HEX
      mov BH, AH

      mov DL, AL
      mov AH, 02h
      int 21h

      mov DL, BH
      mov AH, 02h
      int 21h
```

```
      pop DX
      pop BX
      ret
PRINT_BYTE    ENDP
;-------------------------------------------------------
WRD_TO_HEX PROC near
; перевод в 16сс 16ти разрядного числа
; 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 байта в 10сс, si - адрес поля младшей цифры
; al содержит исходный байт
        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
;ax содержит исходное слово
;si адрес поля младшей цифры
            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
            jne loop_wd
end_l1:
            pop dx
            pop cx
            pop ax
            ret
WRD_TO_DEC ENDP
;-------------------------------------------------------
BEGIN:
            ;Available memory
```

```
      mov bx, 0ffffh
      mov ah, 4Ah
int 21h
mov ax, bx
mov cx, 10h
mul cx
lea  si, AVAILABLE_MEM + 27
call WRD_TO_DEC
      lea dx, AVAILABLE_MEM
      call WriteMsg

      ;Extended memory size
      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_MEM_SIZE + 26
      call WRD_TO_DEC
      lea dx, EXTENDED_MEM_SIZE
      call WriteMsg

      lea bx, _END
      add bx, 10h
      shr bx, 1
      shr bx, 1
      shr bx, 1
      shr bx, 1 ;Делим на 16, чтобы получить параграфы

      mov ah, 4Ah
      int 21h

      mov BX, 1000h
      mov AH, 48h
      int 21h

      ;MCB info
      ; get first mcb's address
      mov AH, 52h
      int 21h
      mov AX, ES:[BX-2]
      mov ES, AX

      MCB_LOOP:
                  lea dx, MCB_TYPE
                  call WriteMsg

                  ; get owner
                  mov BX, ES:[1h]

                  lea dx, STR_FREE
                  cmp bx, 0000h
                  je MCB_MATCH
                  lea dx, STR_OSXMSUBM
                  cmp bx, 0006h
                  je MCB_MATCH
                  lea dx, STR_TOP_MEM
                  cmp bx, 0007h
                  je MCB_MATCH
                  lea dx, STR_MSDOS
                  cmp bx, 0008h
                  je MCB_MATCH
                  lea dx, STR_TAKEN386
                  cmp bx, 0FFFAh
                  je MCB_MATCH
                  lea dx, STR_BLOCKED386
                  cmp bx, 0FFFDh
```

```
                        je MCB_MATCH
                        lea dx, STR_OWNED386
                        cmp bx, 0FFFEh
                        je MCB_MATCH
                        jmp MCB_NOT_MATCH

                        MCB_NOT_MATCH:
                              mov AL, BH
                              call PRINT_BYTE
                              mov AL, BL
                              call PRINT_BYTE
                              jmp MCB_MATCH_END

                        MCB_MATCH:
                              call WriteMsg

                        MCB_MATCH_END:

                        ; get size
                        mov ax, es:[3h]
                        mov bx, 10h
                        mul bx

                        lea si, MCB_SIZE
                        add si, 18
                        call WRD_TO_DEC
                        lea dx, MCB_SIZE
                        call WriteMsg

                        ;print last 8 bytes
                        lea dx, MCB_LAST_BYTE
                        call WriteMsg

                        mov cx, 8
                        mov bx, 8h
                        mov ah, 02h

                        PRINT_BYTE_LOOP:
                                    mov dl, es:[bx]
                                    int 21h
                                    inc bx
                                    loop PRINT_BYTE_LOOP
                        lea dx, ENDL
                        call WriteMsg

                        ;check last block
                        mov al, es:[0h]
                        cmp al, 5Ah
                        je _END

                        ;get new block
                        mov ax, es:[3h]
                        mov bx, es
                        add bx, ax
                        inc bx
                        mov es, bx
                        jmp MCB_LOOP


_END:
            ret

TESTPC ENDS
            END START
```

ПРИЛОЖЕНИЕ Г

СОДЕРЖИМОЕ ФАЙЛА LAB3_4.ASM

```
TESTPC SEGMENT
        ASSUME cs:TESTPC, ds:TESTPC, es:NOTHING, ss:NOTHING
        ORG 100H
START: jmp BEGIN
;DATA
AVAILABLE_MEM          db 'Available memory:
',0DH,0AH,'$'
EXTENDED_MEM_SIZE      db 'Extended memory size:
',0DH,0AH,'$'
MCB_TYPE               db 'MCB type: ','$'
MCB_SIZE               db ';  MCB size:        bytes; ','$'
MCB_LAST_BYTE          db 'MCB last 8 bytes:','$'
STR_FREE               db 'free','$'
STR_OSXMSUBM           db 'OS XMS UMB','$'
STR_TOP_MEM            db 'drivers top memory','$'
STR_MSDOS              db 'MS DOS','$'
STR_TAKEN386           db '386MAX UMBs block','$'
STR_BLOCKED386         db 'blocked by 386MAX','$'
STR_OWNED386           db '386MAX UMB','$'
STR_ALLOCATE_SUCCESS db 'successful allocation',0DH,0AH,'$'
STR_ALLOCATE_ERROR  db 'unsuccessful allocation',0DH,0AH,'$'
ENDL                   db 0DH,0AH, '$'
;--------------------------------------------------
WriteMsg PROC near
           push ax
        mov ah,09h
        int 21h
           pop ax
        ret
WriteMsg 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 в шест. сс ax
        push cx
        mov ah,al
        call TETR_TO_HEX
        xchg al,ah
        mov cl,4
        shr al,cl
        call TETR_TO_HEX ;al - старшая
```

```
        pop cx ;ah - младшая цифра
        ret
BYTE_TO_HEX ENDP
;--------------------------------------------------
PRINT_BYTE      PROC near
; prints AL as two hex digits
     push BX
     push DX

     call BYTE_TO_HEX
     mov BH, AH

     mov DL, AL
     mov AH, 02h
     int 21h

     mov DL, BH
     mov AH, 02h
     int 21h

     pop DX
     pop BX
     ret
PRINT_BYTE    ENDP
;--------------------------------------------------
WRD_TO_HEX PROC near
; перевод в 16сс 16ти разрядного числа
; 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 байта в 10сс, si - адрес поля младшей цифры
; al содержит исходный байт
        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
;ax содержит исходное слово
;si адрес поля младшей цифры
          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
          jne loop_wd
end_l1:
          pop dx
          pop cx
          pop ax
          ret
WRD_TO_DEC ENDP
;--------------------------------------------------
BEGIN:
          ;Available memory
          mov bx, 0ffffh
          mov ah, 4Ah
     int 21h
     mov ax, bx
     mov cx, 10h
     mul cx
     lea  si, AVAILABLE_MEM + 27
     call WRD_TO_DEC
          lea dx, AVAILABLE_MEM
          call WriteMsg

          ;Extended memory size
          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_MEM_SIZE + 26
call WRD_TO_DEC
lea dx, EXTENDED_MEM_SIZE
call WriteMsg

;allocate memory
mov bx, 1000h
mov ah, 48h
int 21h

jnc ALLOCATE_SUCCESS
jmp ALLOCATE_ERROR

ALLOCATE_SUCCESS:
      lea dx, STR_ALLOCATE_SUCCESS
      jmp ALLOCATE_MEMORY_END
ALLOCATE_ERROR:
      lea dx, STR_ALLOCATE_ERROR

ALLOCATE_MEMORY_END:
      call WriteMsg

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

lea bx, _END
add bx, 10h
shr bx, 1
shr bx, 1
shr bx, 1
shr bx, 1 ;Делим на 16, чтобы получить параграфы

mov ah, 4Ah
int 21h

;MCB info
; get first mcb's address
mov AH, 52h
int 21h
mov AX, ES:[BX-2]
mov ES, AX

MCB_LOOP:
            lea dx, MCB_TYPE
```

```
call WriteMsg

; get owner
mov BX, ES:[1h]

lea dx, STR_FREE
cmp bx, 0000h
je MCB_MATCH
lea dx, STR_OSXMSUBM
cmp bx, 0006h
je MCB_MATCH
lea dx, STR_TOP_MEM
cmp bx, 0007h
je MCB_MATCH
lea dx, STR_MSDOS
cmp bx, 0008h
je MCB_MATCH
lea dx, STR_TAKEN386
cmp bx, 0FFFAh
je MCB_MATCH
lea dx, STR_BLOCKED386
cmp bx, 0FFFDh
je MCB_MATCH
lea dx, STR_OWNED386
cmp bx, 0FFFEh
je MCB_MATCH
jmp MCB_NOT_MATCH

MCB_NOT_MATCH:
      mov AL, BH
      call PRINT_BYTE
      mov AL, BL
      call PRINT_BYTE
      jmp MCB_MATCH_END

MCB_MATCH:
      call WriteMsg

MCB_MATCH_END:

; get size
mov ax, es:[3h]
mov bx, 10h
mul bx

lea si, MCB_SIZE
add si, 18
call WRD_TO_DEC
lea dx, MCB_SIZE
call WriteMsg

;print last 8 bytes
lea dx, MCB_LAST_BYTE
```

```
                        call WriteMsg

                        mov cx, 8
                        mov bx, 8h
                        mov ah, 02h

                        PRINT_BYTE_LOOP:
                                    mov dl, es:[bx]
                                    int 21h
                                    inc bx
                                    loop PRINT_BYTE_LOOP
                        lea dx, ENDL
                        call WriteMsg

                        ;check last block
                        mov al, es:[0h]
                        cmp al, 5Ah
                        je _END

                        ;get new block
                        mov ax, es:[3h]
                        mov bx, es
                        add bx, ax
                        inc bx
                        mov es, bx
                        jmp MCB_LOOP


_END:
            ret
ALLOCATION_ERR:


TESTPC ENDS
            END START
```