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

Студентка гр. 7381 _____ Машина Ю.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Постановка задачи.

Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованной 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

Ход работы.

В ходе выполнения данной работы были использованы сведения из табл. 1 (структура MCB) и был создан набор функций и структур данных, описанных в табл. 2-3.

Таблица 1 – структура MCB.

Смещение	Длина поля (байт)	Содержимое поля
00h	1	тип MCB: 5Ah, если последний в списке, 4Dh, если не последний
01h	2	Сегментный адрес PSP владельца участка памяти, либо 0000h - свободный участок, 0006h - участок принадлежит драйверу OS XMS UMB 0007h - участок является исключенной верхней памятью драйверов 0008h - участок принадлежит MS DOS FFFAh - участок занят управляющим блоком 386MAX UMB FFFDh - участок заблокирован 386MAX FFFEh - участок принадлежит 386MAX UMB
03h	2	Размер участка в параграфах
05h	3	Зарезервирован
08h	8	"SC" - если участок принадлежит MS DOS, то в нем системный код

Таблица 2 – Описание функций.

Название функции	**Назначение**
Available_Memory	Печатает количество доступной памяти
Extended_Memory	Печатает размер расширенной памяти
MCB_Data	Выводит цепочку блоков управления памятью
Print	Вызывает функцию печати строки
TETR_TO_HEX	Вспомогательная функция для работы функции BYTE_TO_HEX
BYTE_TO_HEX	Переводит число AL в коды символов 16-ой с/с, записывая получившееся в BL и BH
WRD_TO_HEX	Переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры
BYTE_TO_DEC	Переводит байт из AL в десятичнуюс/с и записывает получившееся число по адресу SI, начиная с младшей цифры

Таблица 3 - Описание структур данных.

Название	**Тип**	**Назначение**
AvailableMemory	db	Доступная память
ExtendedMemory	db	Расширенная память
MCBchain	db	Цепочка блоков управления памятью
header	db	Заголовок таблицы
FailMsg	db	Сообщение об ошибке
SuccessMsg	db	Сообщение об успехе
endline	db	Перенос строки

Была написана программа, которая выполняет следующие действия:

1) Печатает количество доступной памяти
2) Печатает размер расширенной памяти
3) Выводит цепочку блоков управления памятью

А также написаны 3 ее модификации:

1. Изменить программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используется функцию 4Ah прерывания 21h.

2. Изменить программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H.

3. Изменить первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти.

Результаты работы программы представлены на рис. 1-4.

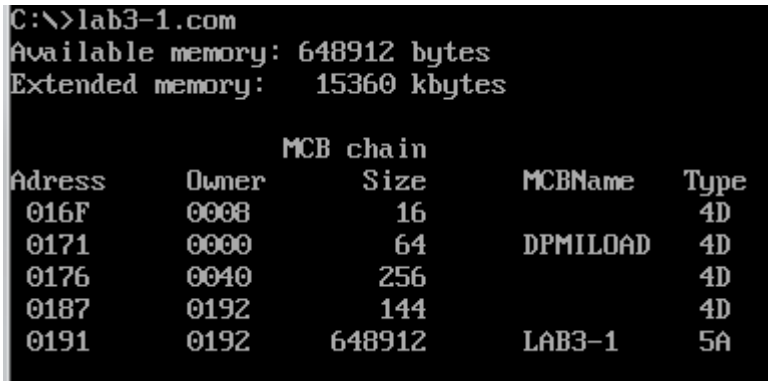

Рисунок 1 – Результат выполнения программы lr3-1.com (без модификаций)

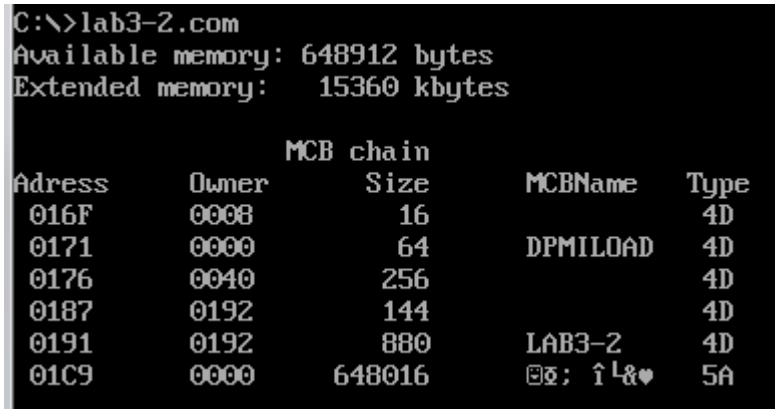

Рисунок 2 – Результат выполнения программы lr3-2.com (с модификацией 1)

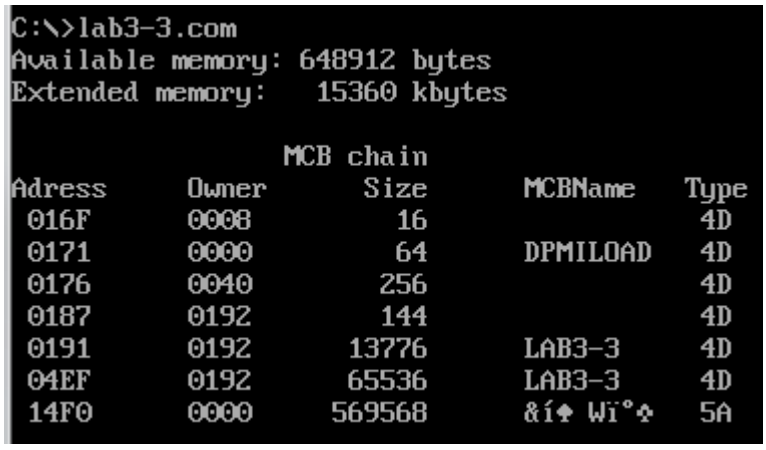

Рисунок 3 – Результат выполнения программы lr3-3.com (с модификацией 2)

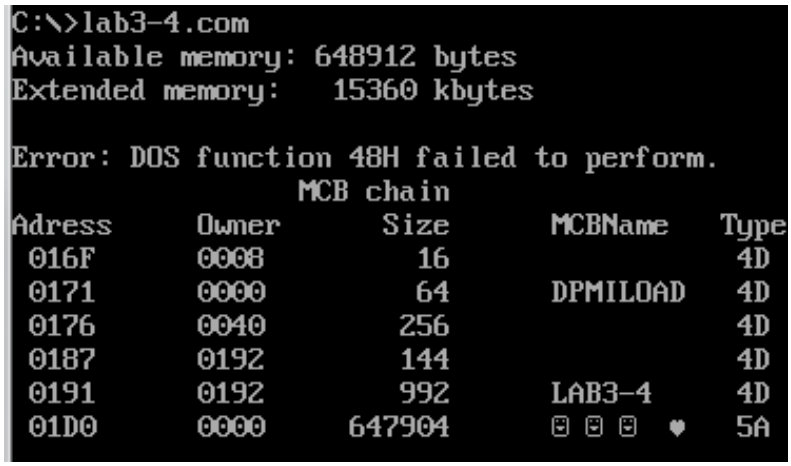

Рисунок 4 – Результат выполнения программы lr3-4.com (с модификацией 3)

Выводы.

В процессе выполнения данной лабораторной работы были исследованы структуры данных и работа функций управления памятью ядра операционной системы. Коды программ lr3-1.asm, lr3-2.asm, lr3-3.asm и lr3-4.asm представлены в приложении А

Ответы на контрольные вопросы.

1. Что означает «доступный объем памяти»?

Доступный объем памяти – это тот объем памяти, в который можно загружать пользовательские программы.

2. Где MCB блок Вашей программы в списке?

У программы есть во всех случаях не менее двух блоков: первый во всех четырех случаях имеет сегментный адрес 0187 и управляет областью переменных среды, второй во всех случаях имеет адрес 0191 и управляет областью, выделенной для программы. В третьем случае есть третий блок, который управляет выделенной областью размера 64 кб. Блок первой программы расположен в конце списка (см. рис. 1).

3. Какой размер памяти занимает программа в каждом случае?

В первом случае – всю доступную память, 648912 байт. Во втором – сколько ей необходимо, 880 байтов. В третьем – 880 минимально необходимых ей + 64 кб, выделенные по запросу. В четвертом – всю доступную память, 992 байта (сначала запросили память, но она не выделилась, а потом освободили неиспользуемую память.).

ПРИЛОЖЕНИЕ А

lr3-1.asm

```
PCMemory   SEGMENT
           ASSUME CS:PCMemory, DS:PCMemory, es:NOTHING, SS:NOTHING
           ORG 100H
START: JMP BEGIN

AvailableMemory db "Available memory:        bytes", 0dh, 0ah, '$'
FreeMemory db "Free memory:        bytes", 0dh, 0ah, '$'
endline db 0dh, 0ah, '$'
ExtendedMemory db "Extended memory:         kbytes", 0dh, 0ah, 0dh,
0ah, '$'
MCBchain db '                  MCB chain ', 0dh, 0ah, '$'
header db 'Adress     Owner      Size      MCBName   Type', 0dh,
0ah, '$'
MCB 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 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
;--------------------------------------------------
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
    push AX
      push CX
      push DX
      xor AH,AH
      xor DX,DX
      mov CX,10
loopfld: div CX
      or DL,30h
      mov [SI],DL
      dec SI
      xor DX,DX
      cmp AX,10
      jae loopfld
      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
;--------------------------------
WRD_TO_DEC PROC near
      push cx
      push dx

      mov cx,10
fl: div cx
      or DL, 30h
      mov [SI],DL
      dec SI
      xor dx,dx
      cmp ax,10
      jae fl
      cmp AL,00h
      je endl
      or AL,30h
      mov [SI],AL
endl:
      pop dx
```

```
      pop cx
      ret
WRD_TO_DEC ENDP
;--------------------------------

Print PROC near
      mov AH,09h
      int 21h
      ret
Print ENDP

Available_Memory PROC NEAR
      push AX
      push BX
      push DX
      push si

      sub ax, ax
      mov ah, 04Ah
      mov bx, 0FFFFh
      int 21h
      mov ax, 10h
      mul bx
      mov si, offset AvailableMemory
      add si, 017h
      call WRD_TO_DEC
      mov dx, offset AvailableMemory
      call Print

      pop si
      pop DX
      pop BX
      pop AX
      ret
Available_Memory ENDP

Extended_Memory PROC NEAR
      push ax
      push si
      push dx
      mov al, 31h ;размер расширенной памяти нах. в ячейках 30h, 31h
CMOS
      out 70h, al
      in al, 71h ; чтение старшего байта размера расширенной памяти
      mov ah, al ;
      mov al, 30h ; запись адреса ячейки CMOS
      out 70h, al
      in al, 71h ; чтение младшего байта
      mov bl, al ; размера расширенной памяти

      lea si, ExtendedMemory
      add si, 23
```

```
      xor dx, dx
      call WRD_TO_DEC
      lea dx, ExtendedMemory
      call Print

      pop dx
      pop si
      pop ax
      ret
Extended_Memory ENDP

MCB_Data PROC near
      push ax
      push bx
      push cx
      push dx
      push es
      push si
      push di

      lea dx, MCBchain
      call Print
      lea dx,  header
      call Print
      mov AH, 52h ; Get List of Lists
      int 21h
      mov es, es:[bx-2]          ; слово по адресу es:[bx-2] - адрес
самого первого MCB

      loopMCB:
          mov ax, es   ; текущий адрес MCB
          lea di, MCB
            add di, 4
            call WRD_TO_HEX
            mov ax, es:[01h] ; сегментный адрес PSP владельца участка
памяти
            lea di, MCB
            add di, 14
            call WRD_TO_HEX
            mov ax, es:[03h]      ; размер участка в параграфах
            mov bx, 10h
            mul bx
            lea si, MCB
            add si, 25
            call WRD_TO_DEC

            lea di, MCB
            add di, 32
            mov cx, 8
            mov bx, 0
      MCBname:
```

```
            mov al, es:[08h + bx] ;sc- в участке системный код, sd-в
нем системные данные
            mov [di + bx], al
            inc bx
      loop MCBname

      mov al, es:[00h] ; тип MCB
      lea di, MCB
      add di, 43
      call BYTE_TO_HEX
      mov [di], al
      inc di
      mov [di], ah
      lea dx, MCB
      call Print
      mov ax, es
      add ax, es:[03h]
      inc ax
      mov BL, es:[00h]
      mov es, ax ; адрес следующего MCB в списке
      cmp BL, 4Dh ; 4Dh - не последний в списке
      je loopMCB ; 5Ah - последний в списке

      pop di
      pop si
      pop es
      pop dx
      pop cx
      pop bx
      pop ax
      ret
MCB_Data ENDP

BEGIN:
      call Available_Memory
      call Extended_Memory
      call MCB_Data
      xor AL,AL  ;|
      mov AH,4Ch ;| exit to dos
      int 21H    ;|
PCMemory   ENDS
           END START
```

lr3-2.asm

```
PCMemory      SEGMENT
   ASSUME CS:PCMemory, DS:PCMemory, es:NOTHING, SS:NOTHING
   ORG 100H
START: JMP BEGIN

AvailableMemory db "Available memory:        bytes", 0dh, 0ah, '$'
FreeMemory db "Free memory:        bytes", 0dh, 0ah, '$'
endline db 0dh, 0ah, '$'
ExtendedMemory db "Extended memory:         kbytes", 0dh, 0ah, 0dh, 0ah,
'$'
MCBchain db '                 MCB chain ', 0dh, 0ah, '$'
header db 'Adress     Owner      Size      MCBName   Type', 0dh, 0ah,
'$'
MCB 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 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
;--------------------------------------------------
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
    push AX
 push CX
 push DX
 xor AH,AH
 xor DX,DX
 mov CX,10
loopfld: div CX
 or DL,30h
 mov [SI],DL
 dec SI
 xor DX,DX
 cmp AX,10
 jae loopfld
 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
;--------------------------------
WRD_TO_DEC PROC near
 push cx
 push dx

 mov cx,10
fl: div cx
 or DL, 30h
 mov [SI],DL
 dec SI
 xor dx,dx
 cmp ax,10
 jae fl
 cmp AL,00h
 je endl
 or AL,30h
 mov [SI],AL
endl:
 pop dx
 pop cx
 ret
WRD_TO_DEC ENDP
;--------------------------------
```

```
Print PROC near
 mov AH,09h
 int 21h
 ret
Print ENDP

Available_Memory PROC NEAR
 push AX
 push BX
 push DX
 push si

 sub ax, ax
 mov ah, 04Ah
 mov bx, 0FFFFh
 int 21h
 mov ax, 10h
 mul bx
 mov si, offset AvailableMemory
 add si, 017h
 call WRD_TO_DEC
 mov dx, offset AvailableMemory
 call Print

 pop si
 pop DX
 pop BX
 pop AX
 ret
Available_Memory ENDP

Extended_Memory PROC NEAR
 push ax
 push si
 push dx
 mov al, 31h ;размер расширенной памяти нах. в ячейках 30h, 31h CMOS
 out 70h, al
 in al, 71h ; чтение старшего байта размера расширенной памяти
 mov ah, al ;
 mov al, 30h ; запись адреса ячейки CMOS
 out 70h, al
 in al, 71h ; чтение младшего байта
 mov bl, al ; размера расширенной памяти

 lea si, ExtendedMemory
 add si, 23
 xor dx, dx
 call WRD_TO_DEC
 lea dx, ExtendedMemory
 call Print
```

```
 pop dx
 pop si
 pop ax
 ret
Extended_Memory ENDP

MCB_Data PROC near
 push ax
 push bx
 push cx
 push dx
 push es
 push si
 push di

 lea dx, MCBchain
 call Print
 lea dx,  header
 call Print
 mov AH, 52h ; Get List of Lists
 int 21h
 mov es, es:[bx-2]       ; слово по адресу es:[bx-2] - адрес самого
первого MCB

 loopMCB:
     mov ax, es     ; текущий адрес MCB
     lea di, MCB
    add di, 4
    call WRD_TO_HEX
    mov ax, es:[01h] ; сегментный адрес PSP владельца участка памяти
    lea di, MCB
    add di, 14
    call WRD_TO_HEX
    mov ax, es:[03h]      ; размер участка в параграфах
    mov bx, 10h
    mul bx
    lea si, MCB
    add si, 25
    call WRD_TO_DEC

    lea di, MCB
    add di, 32
    mov cx, 8
    mov bx, 0
 MCBname:
    mov al, es:[08h + bx] ;sc- в участке системный код, sd-в нем
системные данные
    mov [di + bx], al
    inc bx
 loop MCBname

 mov al, es:[00h] ; тип MCB
```

```
 lea di, MCB
 add di, 43
 call BYTE_TO_HEX
 mov [di], al
 inc di
 mov [di], ah
 lea dx, MCB
 call Print
 mov ax, es
 add ax, es:[03h]
 inc ax
 mov BL, es:[00h]
 mov es, ax   ; адрес следующего MCB в списке

 cmp BL, 4Dh ; 4Dh - не последний в списке
 je loopMCB   ; 5Ah - последний в списке

 pop di
 pop si
 pop es
 pop dx
 pop cx
 pop bx
 pop ax
 ret
MCB_Data ENDP

BEGIN:
 call Available_Memory
 call Extended_Memory
    mov ax, offset programm_end_byte      ; get program size, bytes
 mov bx, ax
 and bx, 0Fh ; приведет bx к 00xx
 cmp bx, 0h
 je size_multiple_of_10h
 add ax, 0fh  ; чтобы округлить
size_multiple_of_10h:
 mov bl, 10h
 div bl
 mov bl, al ; желаемый размер блока в 16-байтовых параграфах
 mov ah, 4ah ;Изменяет размер существующего блока памяти
 int 21h

 call MCB_Data

 xor AL,AL  ;|
 mov AH,4Ch ;| exit to dos
 int 21H    ;|
    programm_end_byte db 0
PCMemory      ENDS
                              END START
```

lr3-3.asm

```
PCMemory      SEGMENT
   ASSUME CS:PCMemory, DS:PCMemory, es:NOTHING, SS:NOTHING
   ORG 100H
START: JMP BEGIN

AvailableMemory db "Available memory:        bytes", 0dh, 0ah, '$'
FreeMemory db "Free memory:        bytes", 0dh, 0ah, '$'
endline db 0dh, 0ah, '$'
ExtendedMemory db "Extended memory:         kbytes", 0dh, 0ah, 0dh, 0ah,
'$'
MCBchain db '                 MCB chain ', 0dh, 0ah, '$'
header db 'Adress     Owner      Size      MCBName   Type', 0dh, 0ah,
'$'
MCB 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 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
;-------------------------------------------------
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
    push AX
 push CX
 push DX
 xor AH,AH
 xor DX,DX
 mov CX,10
loopfld: div CX
 or DL,30h
 mov [SI],DL
 dec SI
 xor DX,DX
 cmp AX,10
 jae loopfld
 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
;--------------------------------
WRD_TO_DEC PROC near
 push cx
 push dx

 mov cx,10
fl: div cx
 or DL, 30h
 mov [SI],DL
 dec SI
 xor dx,dx
 cmp ax,10
 jae fl
 cmp AL,00h
 je endl
 or AL,30h
 mov [SI],AL
endl:
 pop dx
 pop cx
 ret
WRD_TO_DEC ENDP
;--------------------------------
```

```
Print PROC near
 mov AH,09h
 int 21h
 ret
Print ENDP

Available_Memory PROC NEAR
 push AX
 push BX
 push DX
 push si

 sub ax, ax
 mov ah, 04Ah
 mov bx, 0FFFFh
 int 21h
 mov ax, 10h
 mul bx
 mov si, offset AvailableMemory
 add si, 017h
 call WRD_TO_DEC
 mov dx, offset AvailableMemory

 call Print

 pop si
 pop DX
 pop BX
 pop AX
 ret
Available_Memory ENDP

Extended_Memory PROC NEAR
 push ax
 push si
 push dx

 mov al, 31h ;размер расширенной памяти нах. в ячейках 30h, 31h CMOS
 out 70h, al
 in al, 71h ; чтение старшего байта размера расширенной памяти
 mov ah, al ;

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

 lea si, ExtendedMemory
 add si, 23
 xor dx, dx
 call WRD_TO_DEC
```

```
 lea dx, ExtendedMemory
 call Print

 pop dx
 pop si
 pop ax
 ret
Extended_Memory ENDP

MCB_Data PROC near
 push ax
 push bx
 push cx
 push dx
 push es
 push si
 push di

 lea dx, MCBchain
 call Print
 lea dx,  header
 call Print

 mov AH, 52h ; Get List of Lists
 int 21h
 mov es, es:[bx-2]       ; слово по адресу es:[bx-2] - адрес самого
первого MCB

 loopMCB:
     mov ax, es     ; текущий адрес MCB
     lea di, MCB
   add di, 4
   call WRD_TO_HEX

   mov ax, es:[01h] ; сегментный адрес PSP владельца участка памяти
   lea di, MCB
   add di, 14
   call WRD_TO_HEX

   mov ax, es:[03h]      ; размер участка в параграфах
   mov bx, 10h
   mul bx
   lea si, MCB
   add si, 25
   call WRD_TO_DEC

   lea di, MCB
   add di, 32
   mov cx, 8
   mov bx, 0
 MCBname:
```

```
   mov al, es:[08h + bx] ;sc- в участке системный код, sd-в нем
системные данные
   mov [di + bx], al
   inc bx
 loop MCBname

 mov al, es:[00h] ; тип MCB
 lea di, MCB
 add di, 43
 call BYTE_TO_HEX
 mov [di], al
 inc di
 mov [di], ah

 lea dx, MCB
 call Print

 mov ax, es
 add ax, es:[03h]
 inc ax
 mov BL, es:[00h]
 mov es, ax   ; адрес следующего MCB в списке

 cmp BL, 4Dh ; 4Dh - не последний в списке
 je loopMCB   ; 5Ah - последний в списке

 pop di
 pop si
 pop es
 pop dx
 pop cx
 pop bx
 pop ax
 ret
MCB_Data ENDP

BEGIN:
 call Available_Memory
 call Extended_Memory
    mov ah, 4ah
 mov bx, offset programm_end_byte
 int 21h

 mov ah, 48h ;распределить память (дать размер памяти)
 mov bx, 1000h ;запрошенное количество памяти в 16-байтовых параграфах
 int 21h

 call MCB_Data

 xor AL,AL  ;|
 mov AH,4Ch ;| exit to dos
 int 21H    ;|
```

```
    programm_end_byte db 0
PCMemory        ENDS
                                END START
```

lr3-4.asm

```
PCMemory      SEGMENT
   ASSUME CS:PCMemory, DS:PCMemory, es:NOTHING, SS:NOTHING
   ORG 100H
START: JMP BEGIN

AvailableMemory db "Available memory:        bytes", 0dh, 0ah, '$'
FreeMemory db "Free memory:        bytes", 0dh, 0ah, '$'
endline db 0dh, 0ah, '$'
ExtendedMemory db "Extended memory:         kbytes", 0dh, 0ah, 0dh, 0ah,
'$'
MCBchain db '                 MCB chain ', 0dh, 0ah, '$'
header db 'Adress     Owner      Size      MCBName   Type', 0dh, 0ah,
'$'
MCB db '                                                ', 0dh, 0ah, '$'
SuccessMsg    db 'DOS function 48H performed successfully.', 0dh, 0ah,
'$'
FailMsg  db 'Error: DOS function 48H failed to perform.', 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 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
;--------------------------------------------------
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
    push AX
 push CX
 push DX
 xor AH,AH
 xor DX,DX
 mov CX,10
loopfld: div CX
 or DL,30h
 mov [SI],DL
 dec SI
 xor DX,DX
 cmp AX,10
 jae loopfld
 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
;-------------------------------
WRD_TO_DEC PROC near
 push cx
 push dx

 mov cx,10
fl: div cx
 or DL, 30h
 mov [SI],DL
 dec SI
 xor dx,dx
 cmp ax,10
 jae fl
 cmp AL,00h
 je endl
 or AL,30h
 mov [SI],AL
endl:
 pop dx
 pop cx
```

```
 ret
WRD_TO_DEC ENDP
;--------------------------------

Print PROC near
 mov AH,09h
 int 21h
 ret
Print ENDP

Available_Memory PROC NEAR
 push AX
 push BX
 push DX
 push si
 sub ax, ax
 mov ah, 04Ah
 mov bx, 0FFFFh
 int 21h
 mov ax, 10h
 mul bx
 mov si, offset AvailableMemory
 add si, 017h
 call WRD_TO_DEC
 mov dx, offset AvailableMemory
 call Print

 pop si
 pop DX
 pop BX
 pop AX
 ret
Available_Memory ENDP

Extended_Memory PROC NEAR
 push ax
 push si
 push dx
 mov al, 31h ;размер расширенной памяти нах. в ячейках 30h, 31h CMOS
 out 70h, al
 in al, 71h ; чтение старшего байта размера расширенной памяти
 mov ah, al ;
 mov al, 30h ; запись адреса ячейки CMOS
 out 70h, al
 in al, 71h ; чтение младшего байта
 mov bl, al ; размера расширенной памяти

 lea si, ExtendedMemory
 add si, 23
 xor dx, dx
 call WRD_TO_DEC
```

```
 lea dx, ExtendedMemory
 call Print

 pop dx
 pop si
 pop ax
 ret
Extended_Memory ENDP

MCB_Data PROC near
 push ax
 push bx
 push cx
 push dx
 push es
 push si
 push di

 lea dx, MCBchain
 call Print
 lea dx,  header
 call Print

 mov AH, 52h ; Get List of Lists
 int 21h
 mov es, es:[bx-2]       ; слово по адресу es:[bx-2] - адрес самого
первого MCB

 loopMCB:
     mov ax, es     ; текущий адрес MCB
     lea di, MCB
    add di, 4
    call WRD_TO_HEX
    mov ax, es:[01h] ; сегментный адрес PSP владельца участка памяти
    lea di, MCB
    add di, 14
    call WRD_TO_HEX
    mov ax, es:[03h]      ; размер участка в параграфах
    mov bx, 10h
    mul bx
    lea si, MCB
    add si, 25
    call WRD_TO_DEC

    lea di, MCB
    add di, 32
    mov cx, 8
    mov bx, 0
 MCBname:
    mov al, es:[08h + bx] ;sc- в участке системный код, sd-в нем
системные данные
    mov [di + bx], al
```

```
   inc bx
 loop MCBname

 mov al, es:[00h] ; тип MCB
 lea di, MCB
 add di, 43
 call BYTE_TO_HEX
 mov [di], al
 inc di
 mov [di], ah
 lea dx, MCB
 call Print
 mov ax, es
 add ax, es:[03h]
 inc ax
 mov BL, es:[00h]
 mov es, ax   ; адрес следующего MCB в списке

 cmp BL, 4Dh ; 4Dh - не последний в списке
 je loopMCB   ; 5Ah - последний в списке

 pop di
 pop si
 pop es
 pop dx
 pop cx
 pop bx
 pop ax
 ret
MCB_Data ENDP

BEGIN:
 call Available_Memory
 call Extended_Memory
    mov ah, 48h
 mov bx, 1000h
 int 21h
 ; обработка завершений функций ядра, проверка флага CF
 jc error
 lea dx, SuccessMsg
 jmp no_error
error:
 lea dx, FailMsg
no_error:
 call Print
 mov ax, offset programm_end_byte        ; размер программы в байтах
 mov bx, ax
 and bx, 0Fh ; приведет bx к виду 00xx
 cmp bx, 0h
 je size_multiple_of_10h
 add ax, 0fh ; чтобы округлить
size_multiple_of_10h:
```

```
 mov bl, 10h
 div bl
 mov bl, al ; желаемый размер блока в 16-байтовых параграфах
 mov ah, 4ah ;Изменяет размер существующего блока памяти
 int 21h
 call MCB_Data
 xor AL,AL  ;|
 mov AH,4Ch ;| exit to dos
 int 21H    ;|
    programm_end_byte db 0
PCMemory      ENDS
   END START
```