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. 7383 _____ Ханова Ю.А.

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**
Avail_mem	Печатает количество доступной памяти
Extended_mem	Печатает размер расширенной памяти
Chain_MCB	Выводит цепочку блоков управления памятью
Write	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 десятичную с/с и записывает получившееся число по адресу SI, начиная с младшей цифры

Таблица 3 - Описание структур данных.

Название	**Тип**	**Назначение**
Av_mem_	db	Доступная память
Ex_mem_	db	Расширенная память
Ch_MCB_	db	Цепочка блоков управления памятью
Ch_MCB_STR	db	Верхняя строка таблицы
Mem_Error	db	Сообщение об ошибке
Error	db	Сообщение об ошибке
sz	db	0
ENDL	db	Перенос строки
SPC	db	Отступ в начале 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.

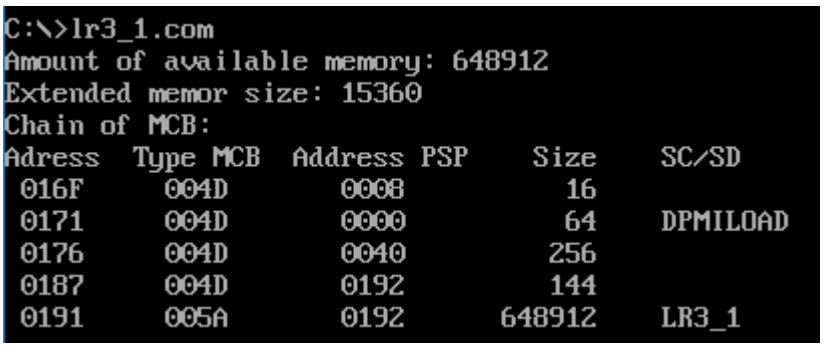

Рисунок 1 – Результат выполнения программы lr3_1.com (без модификаций)

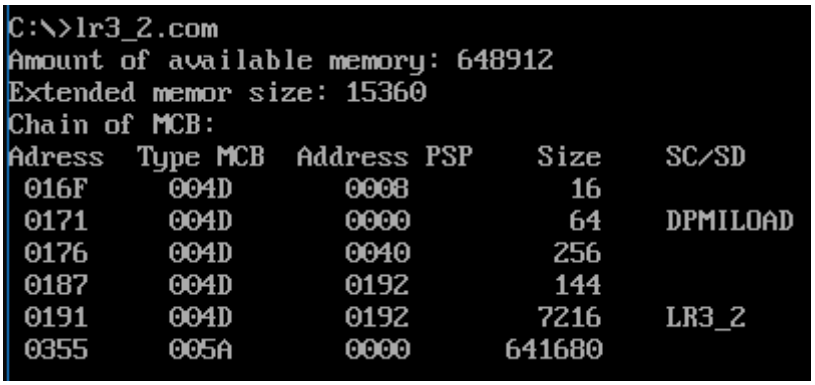

Рисунок 2 – Результат выполнения программы lr3_2.com (с модификацией 1)

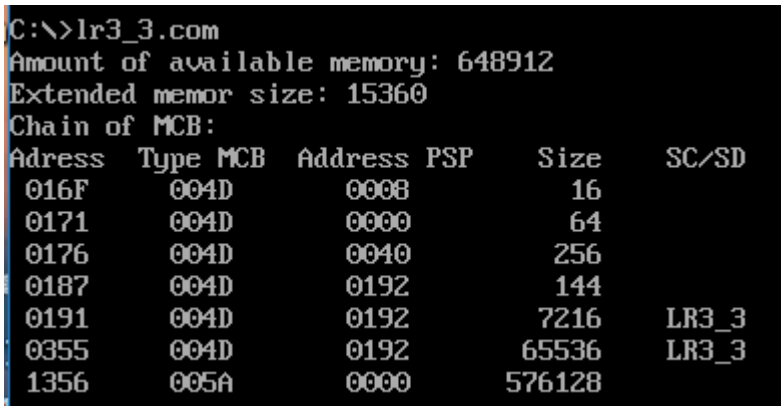

Рисунок 3 – Результат выполнения программы lr3_3.com (с модификацией 2)

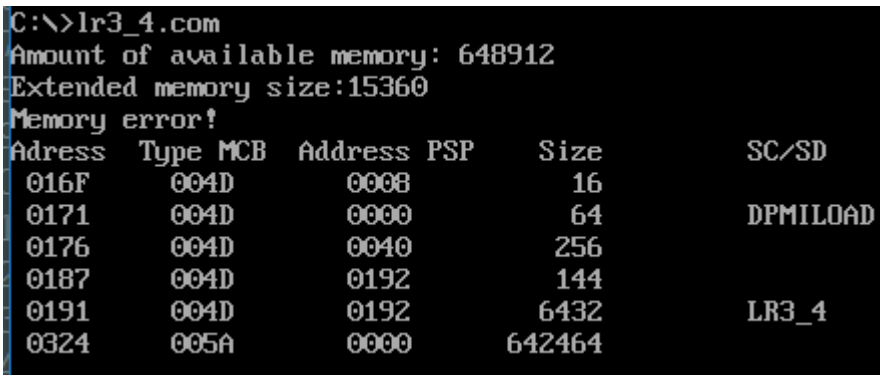

Рисунок 4 – Результат выполнения программы lr3_4.com (с модификацией 3)

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 МСВ блок Вашей программы в списке?

Блок первой программы расположен в конце списка (см. рис. 1).

Блок второй программы есть предпоследняя строка списка (см. рис. 2). В последней строке расположен блок освобожденной памяти.

Блок третьей программы расположен в пятой строке, после него идут блоки выделенной по запросу памяти и свободной памяти соответственно (см. рис. 3). Блок четвертой программы есть предпоследняя строка списка (см. рис. 4). Последнюю строку списка занимает блок, обозначенный, как пустой участок.

3. Какой размер памяти занимает программа в каждом случае?

В первом случае программа занимает всю выделенную память (lr3_1.com): 64 8912 байт. Во втором случае программа занимает свой объем (lr3_2.com): 648912 – 641680 - 16=7216 байт. В третьем случае программа занимает свой размер и объем выделенной памяти (lr3_3.com): 648912-576128-65536-2*16=7216 байт. В четвертом случае (lr3_4.com): 648912 - 642464 - 16=6432 байт.

ПРИЛОЖЕНИЕ А

lr3_1.asm

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

Av_mem_          db 'Amount of available memory:               ',0DH,0AH,'$'
Ex_mem_      db 'Extended memory size:           ',0DH,0AH,'$'
Ch_MCB_      db 'Chain of MCB:    ',0DH,0AH,'$'
Ch_MCB_STR   db 'Adress  Type MCB  Address PSP    Size    SC/SD',0DH,0AH,'$'
ENDL         db 0DH,0AH,'$'
SPC         db '                                        $'

Write PROC near
       mov ah,09h
       int 21h
       ret
Write ENDP

Avail_mem PROC
       mov ax,0
       mov ah,4Ah
       mov bx,0FFFFh
       int 21h
       mov ax,10h
       mul bx
       mov si,offset Av_mem_+33
       call BYTE_TO_DEC
       mov dx, offset Av_mem_
       call Write
       ret
Avail_mem ENDP

Extended_mem PROC near
       mov  AL,30h
    out 70h,AL
    in AL,71h
    mov BL,AL
    mov AL,31h
    out 70h,AL
    in AL,71h
      mov bh,al

      mov ax,bx
      mov dx,0
```

```
        mov si,offset Ex_mem_+25
        call BYTE_TO_DEC
        mov dx,offset Ex_mem_
        call Write

        ret
Extended_mem ENDP

Chain_MCB PROC near
        mov dx,offset Ch_MCB_
        call Write
        mov dx,offset Ch_MCB_STR
        call Write
              push es
        mov ah,52h
        int 21h
        mov bx,es:[bx-2]
        mov es,bx
        Str_Chain:
              mov ax,es
              mov di,offset SPC+4
              call WRD_TO_HEX
              xor ah,ah
              mov al,es:[00h]
              mov di,offset SPC+13
              call WRD_TO_HEX
              mov ax,es:[01h]
              mov di,offset SPC+24
              call WRD_TO_HEX
              mov ax,es:[03h]
              mov si,offset SPC+36
              mov dx, 0
              mov bx, 10h
              mul bx
              call BYTE_TO_DEC
              mov dx,offset SPC
              call Write
              mov cx,8
              mov bx,8
              mov ah,02h
              Str_Chain2:
                    mov dl,es:[bx]
                    add bx,1
                    int 21h
              loop Str_Chain2
              mov dx,offset ENDL
              call Write
```

```
            mov ax,es
            add ax,1
            add ax,es:[03h]
            mov bl,es:[00h]
            mov es,ax

            push bx
            mov bx,offset SPC+42
            mov [bx+19],ax
            mov [bx+21],ax
            mov [bx+23],ax
            pop bx

            cmp bl,4Dh
            je Str_Chain
      pop es
      ret
Chain_MCB ENDP

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
      pop CX
      ret
BYTE_TO_HEX ENDP

; перевод в 16с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
WRD_TO_HEX PROC near
      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

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

BEGIN:
        call Avail_mem
        call Extended_mem
        call Chain_MCB
        xor AL,AL
        mov AH,4Ch
        int 21H
TESTPC ENDS
 END START
```

lr3_2.asm

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

Av_mem_          db 'Amount of available memory:               ',0DH,0AH,'$'
Ex_mem_      db 'Extended memory size:           ',0DH,0AH,'$'
Ch_MCB_      db 'Chain of MCB:    ',0DH,0AH,'$'
Ch_MCB_STR   db 'Adress  Type MCB  Address PSP    Size    SC/SD',0DH,0AH,'$'
ENDL         db 0DH,0AH,'$'
SPC          db '                                        $'
sz           db 0


Write PROC near
       mov ah,09h
       int 21h
       ret
Write ENDP

Avail_mem PROC
       mov ax,0
       mov ah,4Ah
       mov bx,0FFFFh
       int 21h
       mov ax,10h
       mul bx
       mov si,offset Av_mem_+33
       call BYTE_TO_DEC
       mov dx, offset Av_mem_
       call Write
              mov ah,4ah
              mov bx,offset sz
              int 21h
       ret
Avail_mem ENDP

Extended_mem PROC near
       mov  AL,30h
    out 70h,AL
    in AL,71h
    mov BL,AL
    mov AL,31h
    out 70h,AL
    in AL,71h
      mov bh,al
```

```
        mov ax,bx
        mov dx,0
        mov si,offset Ex_mem_+25
        call BYTE_TO_DEC
        mov dx,offset Ex_mem_
        call Write

        ret
Extended_mem ENDP

Chain_MCB PROC near
        mov dx,offset Ch_MCB_
        call Write
        mov dx,offset Ch_MCB_STR
        call Write
              push es
        mov ah,52h
        int 21h
        mov bx,es:[bx-2]
        mov es,bx
        Str_Chain:
              mov ax,es
              mov di,offset SPC+4
              call WRD_TO_HEX
              xor ah,ah
              mov al,es:[00h]
              mov di,offset SPC+13
              call WRD_TO_HEX
              mov ax,es:[01h]
              mov di,offset SPC+24
              call WRD_TO_HEX
              mov ax,es:[03h]
              mov si,offset SPC+36
              mov dx, 0
              mov bx, 10h
              mul bx
              call BYTE_TO_DEC
              mov dx,offset SPC
              call Write
              mov cx,8
              mov bx,8
              mov ah,02h
              Str_Chain2:
                    mov dl,es:[bx]
                    add bx,1
                    int 21h
```

```
            loop Str_Chain2
            mov dx,offset ENDL
            call Write
            mov ax,es
            add ax,1
            add ax,es:[03h]
            mov bl,es:[00h]
            mov es,ax

            push bx
            mov bx,offset SPC+42
            mov [bx+19],ax
            mov [bx+21],ax
            mov [bx+23],ax
            pop bx

            cmp bl,4Dh
            je Str_Chain
      pop es
      ret
Chain_MCB ENDP

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
      pop CX
      ret
BYTE_TO_HEX ENDP

; перевод в 16с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
```

```
WRD_TO_HEX PROC near
       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

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

BEGIN:
       call Avail_mem
       call Extended_mem
       call Chain_MCB
       xor AL,AL
       mov AH,4Ch
       int 21H
```

```
TESTPC ENDS
 END START
```

lr3_3.asm

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

Av_mem_          db 'Amount of available memory:               ',0DH,0AH,'$'
Ex_mem_      db 'Extended memory size:           ',0DH,0AH,'$'
Ch_MCB_      db 'Chain of MCB:    ',0DH,0AH,'$'
Ch_MCB_STR   db 'Adress  Type MCB  Address PSP    Size    SC/SD',0DH,0AH,'$'
ENDL         db 0DH,0AH,'$'
SPC         db '                                        $'
sz          db 0


Write PROC near
       mov ah,09h
       int 21h
       ret
Write ENDP

Avail_mem PROC
       mov ax,0
       mov ah,4Ah
       mov bx,0FFFFh
       int 21h
       mov ax,10h
       mul bx
       mov si,offset Av_mem_+33
       call BYTE_TO_DEC
       mov dx, offset Av_mem_
       call Write
             mov ah,4ah
             mov bx,offset sz
             int 21h

             mov ah,48h
             mov bx,1000h
             int 21h
       ret
Avail_mem ENDP

Extended_mem PROC near
       mov  AL,30h
```

```
    out 70h,AL
    in AL,71h
    mov BL,AL
    mov AL,31h
    out 70h,AL
    in AL,71h
      mov bh,al

      mov ax,bx
      mov dx,0
      mov si,offset Ex_mem_+25
      call BYTE_TO_DEC
      mov dx,offset Ex_mem_
      call Write

      ret
Extended_mem ENDP

Chain_MCB PROC near
      mov dx,offset Ch_MCB_
      call Write
      mov dx,offset Ch_MCB_STR
      call Write
            push es
      mov ah,52h
      int 21h
      mov bx,es:[bx-2]
      mov es,bx
      Str_Chain:
            mov ax,es
            mov di,offset SPC+4
            call WRD_TO_HEX
            xor ah,ah
            mov al,es:[00h]
            mov di,offset SPC+13
            call WRD_TO_HEX
            mov ax,es:[01h]
            mov di,offset SPC+24
            call WRD_TO_HEX
            mov ax,es:[03h]
            mov si,offset SPC+36
            mov dx, 0
            mov bx, 10h
            mul bx
            call BYTE_TO_DEC
            mov dx,offset SPC
            call Write
```

```
            mov cx,8
            mov bx,8
            mov ah,02h
            Str_Chain2:
                  mov dl,es:[bx]
                  add bx,1
                  int 21h
            loop Str_Chain2
            mov dx,offset ENDL
            call Write
            mov ax,es
            add ax,1
            add ax,es:[03h]
            mov bl,es:[00h]
            mov es,ax

            push bx
            mov bx,offset SPC+42
            mov [bx+19],ax
            mov [bx+21],ax
            mov [bx+23],ax
            pop bx

            cmp bl,4Dh
            je Str_Chain
      pop es
      ret
Chain_MCB ENDP

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
        pop CX
        ret
BYTE_TO_HEX ENDP

; перевод в 16с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
WRD_TO_HEX PROC near
        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

; перевод в 10с/с, SI - адрес поля младшей цифры
BYTE_TO_DEC PROC near
        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
```

```
BEGIN:
       call Avail_mem
       call Extended_mem
       call Chain_MCB
       xor AL,AL
       mov AH,4Ch
       int 21H
TESTPC ENDS
 END START
```

lr3_4.asm

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

Av_mem_          db 'Amount of available memory:               ',0DH,0AH,'$'
Ex_mem_      db 'Extended memory size:           ',0DH,0AH,'$'
Ch_MCB_      db 'Chain of MCB:    ',0DH,0AH,'$'
Ch_MCB_STR   db 'Adress  Type MCB  Address PSP    Size         SC/SD',0DH,0AH,'$'
ENDL         db 0DH,0AH,'$'
SPC         db '                                              ','$'
Mem_Error   db 'Memory error!',0Dh,0Ah,'$'
Error       db 'Error!',0Dh,0Ah,'$'

Write PROC near
       mov ah,09h
       int 21h
       ret
Write ENDP

Avail_mem PROC
       ;mov ax,0
       mov ah,4Ah
       mov bx,0FFFFh
       int 21h
       mov ax,10h
       mul bx
       mov si,offset Av_mem_+33
       call BYTE_TO_DEC
       mov dx, offset Av_mem_
       call Write
       ret
Avail_mem ENDP

Extended_mem PROC near
       mov  AL,30h
```

```
    out 70h,AL
    in AL,71h
    mov BL,AL
    mov AL,31h
    out 70h,AL
    in AL,71h
      mov bh,al

      mov ax,bx
      mov dx,0
      mov si,offset Ex_mem_+25
      call BYTE_TO_DEC
      mov dx,offset Ex_mem_
      call Write

      ret
Extended_mem ENDP

Chain_MCB PROC near
      mov bx,1000h
      mov ah,48h
      int 21h
      jnc Mem_Error_
            mov dx,offset Mem_Error
            call Write
      Mem_Error_:

      mov bx,0A000h
      mov ax,offset ENDL
      mov bl,10h
      div bl
      xor ah,ah
      add ax,1

      mov bx,cs
      add ax,bx
      mov bx,es
      sub ax,bx
      mov al,0
      mov ah,4Ah
      int 21h
      jnc Error_
            mov dx,offset Error
            call Write
      Error_:

      mov dx,offset Ch_MCB_STR
```

```
call Write
push es
mov ah,52h
int 21h
mov bx,es:[bx-2]
mov es,bx
Str_Chain:
	mov ax,es
	mov di,offset SPC+4
	call WRD_TO_HEX
	xor ah,ah
	mov al,es:[00h]
	mov di,offset SPC+13
	call WRD_TO_HEX
	mov ax,es:[01h]
	mov di,offset SPC+24
	call WRD_TO_HEX
	mov ax,es:[03h]
	mov si,offset SPC+36
	mov dx, 0
	mov bx, 10h
	mul bx
	call BYTE_TO_DEC
	mov dx,offset SPC
	call Write
	mov cx,8
	mov bx,8
	mov ah,02h
	Str_Chain2:
		mov dl,es:[bx]
		add bx,1
		int 21h
	loop Str_Chain2
	mov dx,offset ENDL
	call Write
	mov ax,es
	add ax,1
	add ax,es:[03h]
	mov bl,es:[00h]
	mov es,ax

	push bx
	mov bx,offset SPC+44
	mov [bx+19],ax
	mov [bx+21],ax
	mov [bx+23],ax
	pop bx
```

```
            cmp bl,4Dh
            je Str_Chain
      pop es
      ret
Chain_MCB ENDP

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
      pop CX
      ret
BYTE_TO_HEX ENDP

; перевод в 16с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
WRD_TO_HEX PROC near
      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

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

BEGIN:
       call Avail_mem
       call Extended_mem
       call Chain_MCB
       xor AL,AL
       mov AH,4Ch
       int 21H
TESTPC ENDS
 END START
```