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 _____ Прокопенко Н.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Цель работы: 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

В лабораторной работе исследуются структуры данных и работа функций управления памятью ядра операционной системы.

Постановка задачи:

Шаг 1. Необходимо написать и отладить программный модуль типа .COM, выбирает и распечатывает следующую информацию:

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

Адреса при выводе представляются шестнадцатеричными числами. Объем памяти функциями управления памятью выводится в параграфах. Необходимо преобразовать его в байты и выводить в виде десятичных чисел. Последние восемь байт MCB выводятся как символы, не следует преобразовывать их в шестнадцатеричные числа.

Шаг 2. Далее необходимо изменить программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah прерывания 21h (пример в разделе «Использование функции 4AH»).

Шаг 3. Затем необходимо изменить программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H.

Шаг 4. Далее нужно изменить первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти.

Шаг 5. Оформить отчёт и ответить на контрольные вопросы.

Необходимые сведения для составления программы:

Учёт занятой и свободной памяти ведется при помощи списка блоков управления памятью MCB (Memory Control Block). MCB занимает 16 байт (параграф) и располагается всегда с адреса кратного 16 (адрес сегмента ОП) и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет.

MCB имеет следующую структуру:

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

		"SD" - если участок принадлежит MS DOS, то в нем системные данные

По сегментному адресу и размеру участка памяти, контролируемого этим MCB можно определить местоположение следующего MCB в списке.

Адрес первого MCB хранится во внутренней структуре MS DOS, называемой "List of Lists" (список списков). Доступ к указателю на эту структуру можно получить, используя функцию f52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX-2] и есть адрес самого первого MCB.

Размер расширенной памяти находится в ячейках 30h, 31h CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта. Размер расширенной памяти в Кбайтах можно определить обращаясь к ячейкам CMOS следующим образом:

```
mov  AL,30h ; запись адреса ячейки CMOS
out 70h,AL
in AL,71h   ; чтение младшего байта
mov BL,AL   ; размера расширенной памяти
mov AL,31h  ; запись адреса ячейки CMOS
out 70h,AL
in AL,71h   ; чтение старшего байта размера расширенной памяти
```

Описание функций и структур данных представлены в табл. 1 и табл.2 соответсвенно.

Таблица 1 – описание функций

Название функции	**Назначение**
AVL_Mem	распечатывает количество доступной памяти
Ext_Mem	распечатывает размер расширенной памяти

Окончание таблицы 1

Write	вызывает функцию печати строки
Chain_of_MBC	выводит цепочку блоков управления памятью
BYTE_TO_HEX	переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah
TETR_TO_HEX	вспомогательная функция для работы функции BYTE_TO_HEX
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 2 – описание структур данных

Название	**Тип**	**Назначение**
avail_mem	db	Строка, информирующая о том, что дальше выведется размер доступной памяти
bu_mem_	db	Строка, хранящая названия столбцов таблицы, в которую будут выводиться данные о MCB
exten_mem	db	Строка, информирующая о том, что дальше выведется размер расширенной памяти

Окончание таблицы 2

MBC_	db	Строка, информирующая об выводе MBC
ERROR_STR	db	Строка, информирующая об ошибке при выделении памяти
ENDL	db	Строка, переводящая курсор на начало новой строки
result	db	Строка для хранения данных о MCB

Примеры работы программы по шагам представлены на рисунке 1, рисунке 2, рисунке 3, рисунке 4.

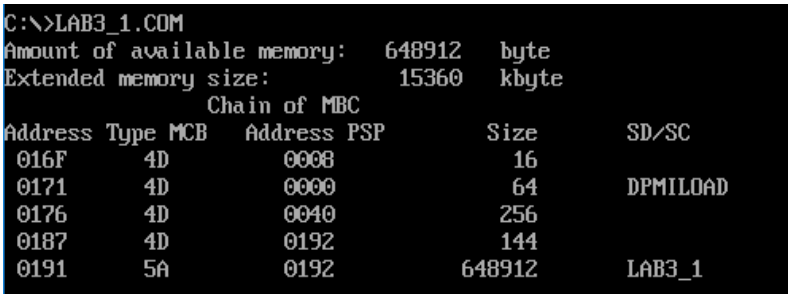

Рисунок 1 – результат работы программы lab3_1.com

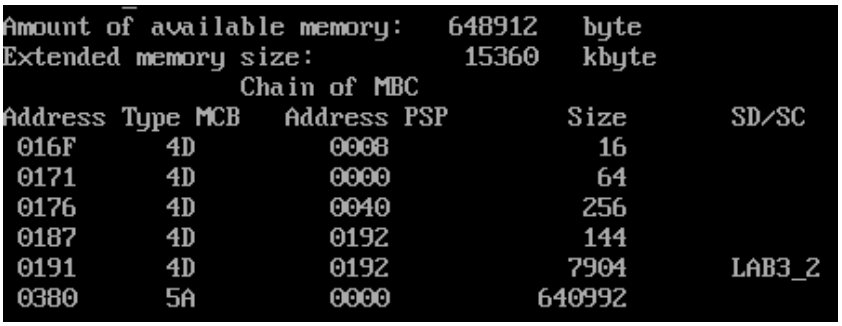

Рисунок 2 – результат работы программы lab3_2.com

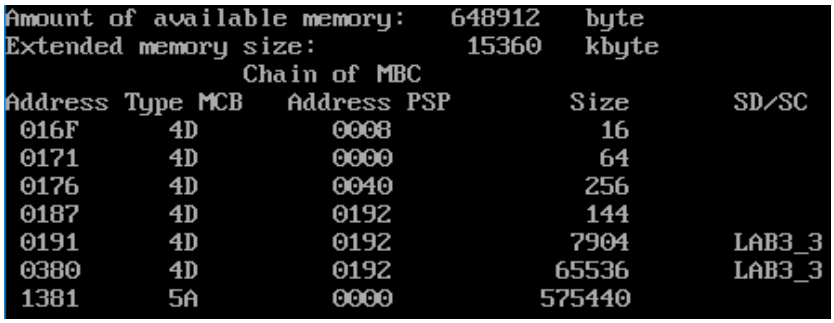

Рисунок 3 – результат работы программы lab3_3.com

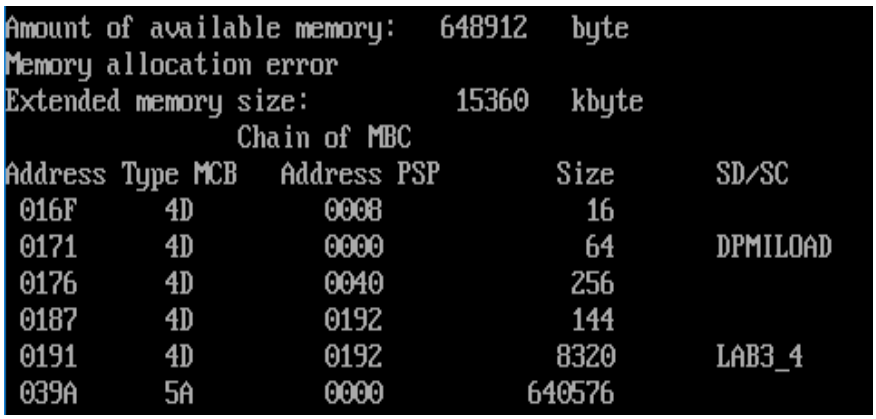

Рисунок 4 – результат работы программы lab3_4.com

Выводы.

В процессе выполнения данной лабораторной работы были исследованы структуры данных и работа функций управления памятью ядра операционной системы. Код программы lab3_1.COM представлен в приложении А, lab3_2.COM в приложении Б, lab3_3.COM в приложении В, lab3_4.COM в приложении Г.

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?

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 (lab3_1.COM): 64 8912 байт. Во втором случае программа занимает свой объем (lab3_2.COM): 648912 – 640992 - 16=7904 байт. В третьем случае программа занимает свой размер и объем выделенной памяти (lab3_3.COM): 648912-575440-65536-2*16=7904 байт. В четвертом случае (lab3_4.COM): 648912 - 640576 - 16=8320 байт.

ПРИЛОЖЕНИЕ А

LAB3_1.ASM

```
      TESTPC SEGMENT
       ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
       ORG 100H
       START: JMP BEGIN
      ; ДАННЫЕ
      avail_mem     db 'Amount of available memory:           byte',0DH,0AH,'$'
      exten_mem     db 'Extended memory size:                 kbyte',0DH,0AH,'$'
      MBC_          db '         Chain of MBC',0DH,0AH,'$'
      bu_mem_       db 'Address  Type MCB   Address PSP         Size    SD/SC
',0DH,0AH,'$'
      result      db '
',0Dh,0Ah,'$'
      ENDL          db 0DH,0AH,'$'

      Write PROC near
       push ax
       mov ah,09h
       int 21h
       pop ax
       ret
      Write ENDP
      AVL_Mem PROC near
       mov ah,4ah
       mov bx,0FFFFh
       int 21h
       mov ax,10h
       mul bx
       mov si,offset avail_mem
       add si,35
       call BYTE_TO_DEC
       mov dx,offset avail_mem
       call Write
       ret
      AVL_Mem ENDP
      Ext_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 exten_mem
 add si,35
 call BYTE_TO_DEC
 mov dx,offset exten_mem
 call Write
 ret
Ext_Mem ENDP
Chain_of_MBC PROC near
        mov di,offset result
        mov ax,es
        add di,4h
        call WRD_TO_HEX
        mov di,offset result
        ;смещение 0h
        add di,0Ch
        xor ah,ah
        mov al,es:[0]
        call WRD_TO_HEX
        mov al,20h
        mov [di],al
        inc di
        mov [di],al
        mov di,offset result
        ;смещение 1h
        mov ax,es:[1]
        add di,19h
        call WRD_TO_HEX
        mov di,offset result
        ;смещение 3h
        mov ax,es:[3]
        mov bx,10h
        mul bx
        add di,29h
        push si
        mov si,di
        call BYTE_TO_DEC
        pop si
        mov di,offset result
        add di,31h
        mov bx,0h
PRINTS:
    mov dl,es:[8+bx]
        mov [di],dl
        inc di
```

```
        inc bx
        cmp bx,8h
        jne PRINTS
        mov ax,es:[3]
        mov bl,es:[0]
        ret
Chain_of_MBC 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
 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
 mov CX,10
loop_bd2: div CX
 or DL,30h
 mov [SI],DL
 dec SI
 xor DX,DX
 cmp AX,10
 jae loop_bd2
 cmp AL,00h
 je end_l2
 or AL,30h
 mov [SI],AL
end_l2: pop DX
 pop CX
 ret
BYTE_TO_DEC ENDP
BEGIN:
 call AVL_Mem
 call Ext_Mem
;MBC
 mov dx, offset MBC_
 call Write
 mov dx,offset bu_mem_
 call Write
 mov ah,52h
 int 21h
 sub bx,2h
 mov es,es:[bx]
MCB_OUT:
        xor ax,ax
        xor bx,bx
        xor cx,cx
        xor dx,dx
        xor di,di
        call Chain_of_MBC
        mov dx,offset result
        call Write
        mov cx,es
        add ax,cx
        inc ax
        mov es,ax
        cmp bl,4Dh
```

```
              je MCB_OUT
       xor AL,AL
       mov AH,4Ch
       int 21H
     TESTPC ENDS
END START
```

ПРИЛОЖЕНИЕ Б
LAB3_2.ASM

```
       TESTPC SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG 100H
        START: JMP BEGIN
       ; ДАННЫЕ
       avail_mem     db 'Amount of available memory:           byte',0DH,0AH,'$'
       exten_mem     db 'Extended memory size:                 kbyte',0DH,0AH,'$'
       MBC_          db '         Chain of MBC',0DH,0AH,'$'
       bu_mem_       db 'Address  Type MCB   Address PSP         Size    SD/SC
',0DH,0AH,'$'
       result      db '
',0Dh,0Ah,'$'
       ENDL          db 0DH,0AH,'$'
       sizepr        db 0

       Write PROC near
        push ax
        mov ah,09h
        int 21h
        pop ax
        ret
       Write ENDP
       AVL_Mem PROC near
        mov ah,4ah
        mov bx,0FFFFh
        int 21h
        mov ax,10h
        mul bx
        mov si,offset avail_mem
        add si,35
        call BYTE_TO_DEC
        mov dx,offset avail_mem
        call Write
        mov ah,4ah
        mov bx,offset sizepr
        int 21h
        ret
       AVL_Mem ENDP
       Ext_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 exten_mem
 add si,35
 call BYTE_TO_DEC
 mov dx,offset exten_mem
 call Write
 ret
Ext_Mem ENDP
Chain_of_MBC PROC near
        mov di,offset result
        mov ax,es
        add di,4h
        call WRD_TO_HEX
        mov di,offset result
        ;смещение 0h
        add di,0Ch
        xor ah,ah
        mov al,es:[00h]
        call WRD_TO_HEX
        mov al,20h
        mov [di],al
        inc di
        mov [di],al
        mov di,offset result
        ;смещение 1h
        mov ax,es:[01h]
        add di,19h
        call WRD_TO_HEX
        mov di,offset result
        ;смещение 3h
        mov ax,es:[03h]
        mov bx,10h
        mul bx
        add di,29h
        mov si,di
        call BYTE_TO_DEC
        mov di,offset result
        add di,31h
        mov bx,0h
PRINTS:
    mov dl,es:[8+bx]
        mov [di],dl
        inc di
        inc bx
        cmp bx,8h
        jne PRINTS
        mov ax,es:[03h]
        mov bl,es:[00h]
        ret
Chain_of_MBC 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
 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
 mov CX,10
loop_bd2: div CX
 or DL,30h
 mov [SI],DL
 dec SI
 xor DX,DX
 cmp AX,10
 jae loop_bd2
 cmp AL,00h
 je end_l2
 or AL,30h
```

```
        mov [SI],AL
      end_l2: pop DX
        pop CX
        ret
      BYTE_TO_DEC ENDP
      BEGIN:
        call AVL_Mem
        call Ext_Mem
      ;MBC
        mov dx, offset MBC_
        call Write
        mov dx,offset bu_mem_
        call Write
        mov ah,52h
        int 21h
        sub bx,2h
        mov es,es:[bx]
      MCB_OUT:
              call Chain_of_MBC
              mov dx,offset result
              call Write
              mov cx,es
              add ax,cx
              inc ax
              mov es,ax
              cmp bl,4Dh
              je MCB_OUT
        xor AL,AL
        mov AH,4Ch
        int 21H
      TESTPC ENDS
END START
```

ПРИЛОЖЕНИЕ В

LAB3_3.ASM

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
 START: JMP BEGIN
; ДАННЫЕ
avail_mem     db 'Amount of available memory:           byte',0DH,0AH,'$'
exten_mem     db 'Extended memory size:                 kbyte',0DH,0AH,'$'
MBC_          db '         Chain of MBC',0DH,0AH,'$'
bu_mem_       db 'Address  Type MCB   Address PSP         Size    SD/SC
',0DH,0AH,'$'
result      db '
',0Dh,0Ah,'$'
ENDL          db 0DH,0AH,'$'
sizepr        db 0

Write PROC near
 push ax
 mov ah,09h
 int 21h
 pop ax
 ret
Write ENDP
AVL_Mem PROC near
 mov ah,4ah
 mov bx,0FFFFh
 int 21h
 mov ax,10h
 mul bx
 mov si,offset avail_mem
 add si,35
 call BYTE_TO_DEC
 mov dx,offset avail_mem
 call Write
 mov ah,4ah
 mov bx,offset sizepr
 int 21h
 mov ah,48h
 mov bx,1000h
 int 21h

 ret
AVL_Mem ENDP
Ext_Mem PROC near
 xor dx,dx
 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 exten_mem
 add si,35
 call BYTE_TO_DEC
 mov dx,offset exten_mem
 call Write
 ret
Ext_Mem ENDP
Chain_of_MBC PROC near
        mov di,offset result
        mov ax,es
        add di,4h
        call WRD_TO_HEX
        mov di,offset result
        ;смещение 0h
        add di,0Ch
        xor ah,ah
        mov al,es:[00h]
        call WRD_TO_HEX
        mov al,20h
        mov [di],al
        inc di
        mov [di],al
        mov di,offset result
        ;смещение 1h
        mov ax,es:[01h]
        add di,19h
        call WRD_TO_HEX
        mov di,offset result
        ;смещение 3h
        mov ax,es:[03h]
        mov bx,10h
        mul bx
        add di,29h
        mov si,di
        call BYTE_TO_DEC
        mov di,offset result
        add di,31h
        mov bx,0h
PRINTS:
    mov dl,es:[8+bx]
        mov [di],dl
        inc di
        inc bx
        cmp bx,8h
        jne PRINTS
        mov ax,es:[03h]
        mov bl,es:[00h]
```

```
        ret
Chain_of_MBC 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
 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
 mov CX,10
loop_bd2: div CX
 or DL,30h
 mov [SI],DL
 dec SI
 xor DX,DX
```

```
        cmp AX,10
        jae loop_bd2
        cmp AL,00h
        je end_l2
        or AL,30h
        mov [SI],AL
      end_l2: pop DX
        pop CX
        ret
      BYTE_TO_DEC ENDP
      BEGIN:
        call AVL_Mem
        call Ext_Mem
      ;MBC

        mov dx, offset MBC_
        call Write
        mov dx,offset bu_mem_
        call Write
        mov ah,52h
        int 21h
        sub bx,2h
        mov es,es:[bx]
      MCB_OUT:
              call Chain_of_MBC
              mov dx,offset result
              call Write
              mov cx,es
              add ax,cx
              inc ax
              mov es,ax
              cmp bl,4Dh
              je MCB_OUT
        xor AL,AL
        mov AH,4Ch
        int 21H
      TESTPC ENDS
END START
```

ПРИЛОЖЕНИЕ Г

LAB3_4.ASM

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
 START: JMP BEGIN
; ДАННЫЕ
avail_mem     db 'Amount of available memory:           byte',0DH,0AH,'$'
exten_mem     db 'Extended memory size:                 kbyte',0DH,0AH,'$'
MBC_          db '         Chain of MBC',0DH,0AH,'$'
bu_mem_       db 'Address  Type MCB   Address PSP          Size    SD/SC
',0DH,0AH,'$'
result                                                          db          '
',0Dh,0Ah,'$'
ENDL          db 0DH,0AH,'$'
ERROR_STR     db 'Memory allocation error',0DH,0AH,'$'
sizepr        db 0


Write PROC near
 push ax
 mov ah,09h
 int 21h
 pop ax
 ret
Write ENDP
AVL_Mem PROC near
 mov ah,4ah
 mov bx,0FFFFh
 int 21h
 mov ax,10h
 mul bx
 mov si,offset avail_mem
 add si,35
 call BYTE_TO_DEC
 mov dx,offset avail_mem
 call Write
 mov ah,48h
 mov bx,1000h
 int 21h
 jnc ERROR
       mov dx,offset ERROR_STR
       call Write
 ERROR:
 mov ah,4ah
 mov bx,offset sizepr
 int 21h

 ret
AVL_Mem ENDP
```

```
Ext_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 exten_mem
 add si,35
 call BYTE_TO_DEC
 mov dx,offset exten_mem
 call Write
 ret
Ext_Mem ENDP
Chain_of_MBC PROC near
        mov di,offset result
        mov ax,es
        add di,4h
        call WRD_TO_HEX
        mov di,offset result
        ;смещение 0h
        add di,0Ch
        xor ah,ah
        mov al,es:[00h]
        call WRD_TO_HEX
        mov al,20h
        mov [di],al
        inc di
        mov [di],al
        mov di,offset result
        ;смещение 1h
        mov ax,es:[01h]
        add di,19h
        call WRD_TO_HEX
        mov di,offset result
        ;смещение 3h
        mov ax,es:[03h]
        mov bx,10h
        mul bx
        add di,29h
        mov si,di
        call BYTE_TO_DEC
        mov di,offset result
        add di,31h
        mov bx,0h
PRINTS:
    mov dl,es:[8+bx]
        mov [di],dl
```

```
        inc di
        inc bx
        cmp bx,8h
        jne PRINTS
        mov ax,es:[03h]
        mov bl,es:[00h]
        ret
Chain_of_MBC 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
 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
```

```
 mov CX,10
loop_bd2: div CX
 or DL,30h
 mov [SI],DL
 dec SI
 xor DX,DX
 cmp AX,10
 jae loop_bd2
 cmp AL,00h
 je end_l2
 or AL,30h
 mov [SI],AL
end_l2: pop DX
 pop CX
 ret
BYTE_TO_DEC ENDP
BEGIN:

 call AVL_Mem
 call Ext_Mem
;MBC

 mov dx, offset MBC_
 call Write
 mov dx,offset bu_mem_
 call Write
 mov ah,52h
 int 21h
 sub bx,2h
 mov es,es:[bx]
MCB_OUT:
        call Chain_of_MBC
        mov dx,offset result
        call Write
        mov cx,es
        add ax,cx
        inc ax
        mov es,ax
        cmp bl,4Dh
        je MCB_OUT
 xor AL,AL
 mov AH,4Ch
 int 21H
TESTPC ENDS
 END START
```