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. 9382 _____ Балаева М.О.

Преподаватель _____ Ефремов М.А

Санкт-Петербург

2021

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.

Описание функций и структур данных.

Название функции	Назначение
DOST	распечатывает количество доступной памяти
RAS	распечатывает размер расширенной памяти
PRINT	вызывает функцию печати строки
MCB	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

Название	Тип	Назначение
DOST_MEM_STR	db	Строка, информирующая о том, что дальше выведется размер доступной памяти

DOST_MEM_STR_	db	Строка для хранения размера доступной памяти
RASS_MEM_STR	db	Строка, информирующая о том, что дальше выведется размер расширенной памяти
RASS_MEM_STR_	db	Строка для хранения размера расширенной памяти
BU_MEM_STR	db	Строка, хранящая названия столбцов таблицы, в которую будут выводиться данные о MCB
ERROR_STR	db	Строка, информирующая об ошибке
ERROR_MEM	db	Строка, информирующая об ошибке при выделении памяти
STRENDL	db	Строка, переводящая курсор на начало новой строки
BU_MEM	db	Строка для хранения данных о MCB

Примеры работы программ:

Рисунок 1 – результат работы программы lab3_1.com

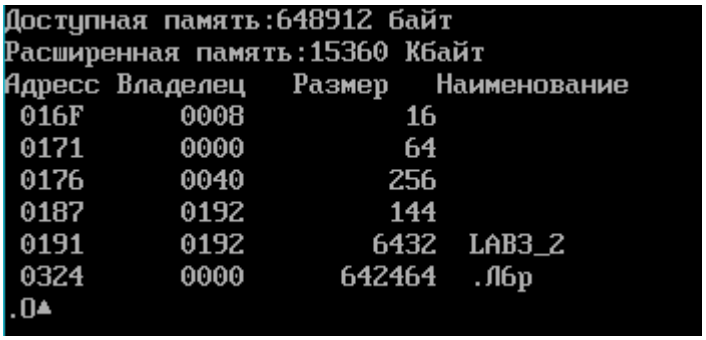

Рисунок 2 – результат работы программы lab3_2.com

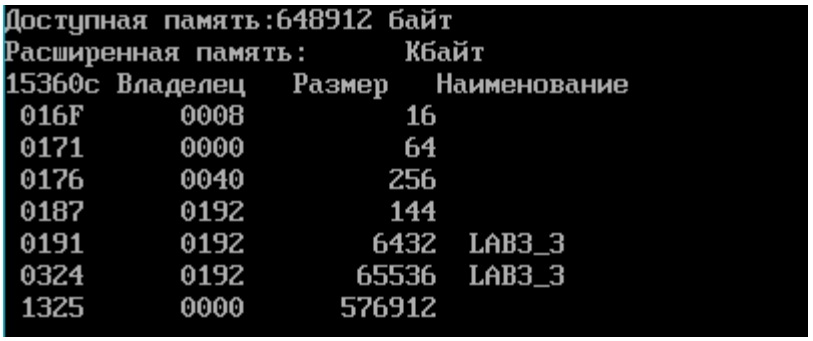

Рисунок 3 – результат работы программы lab3_3.com

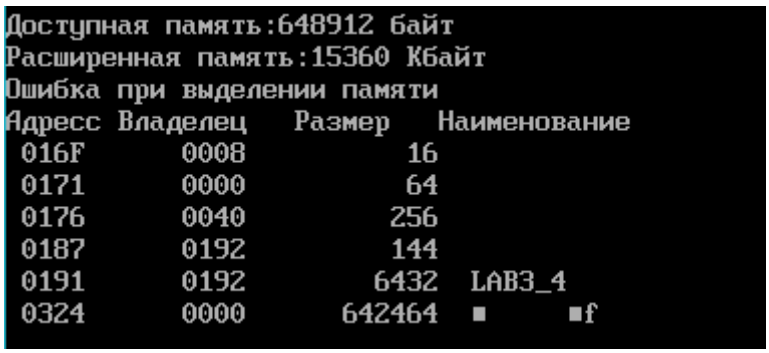

Рисунок 4 – результат работы программы lab3_4.com

Выводы.

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

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)

- 3. Какой размер памяти занимает программа в каждом случае?

В первом случае программа занимает всю выделенную память: 64 8912 б. Во втором случае программа занимает свой объем: 6 432 б. В третьем случае программа занимает свой размер и объем выделенной памяти: 6 432 + 65 536 = 71 968 б. В четвертом случае: 6 432 б

ПРИЛОЖЕНИЕ А

LAB3_1.ASM

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

DOST_MEM_STR db 'Доступная память:'
DOST_MEM_STR_ db '       байт',0DH,0AH,'$'
RASS_MEM_STR db 'Расширенная память:'
RASS_MEM_STR_ db '      Кбайт',0DH,0AH,'$'
BU_MEM_STR   db 'Адресс Владелец   Размер   Наименование',0DH,0AH,'$'
BU_MEM       db '                              $'
STRENDL db 0DH,0AH,'$'


;--------------------------------------
PRINT PROC
 push ax
 mov ah,09h
 int 21h
 pop ax
 ret
PRINT ENDP
;--------------------------------------
DOST PROC
 mov ax,0
 mov ah,4Ah
 mov bx,0FFFFh
 int 21h
 mov ax,bx
 mov bx,16
 mul bx
 mov si,offset DOST_MEM_STR_+5
 call TO_DEC
 mov dx,offset DOST_MEM_STR
 call PRINT
 ret
DOST ENDP
;--------------------------------------
RAS PROC

 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 RASS_MEM_STR_+4
 call TO_DEC
 mov dx,offset RASS_MEM_STR
 call PRINT
```

```
 ret
RAS ENDP
;--------------------------------------
MCB PROC
 mov dx,offset BU_MEM_STR
 call PRINT
 push es
 mov ah,52h
 int 21h
 mov bx,es:[bx-2]
 mov es,bx
 CYCLE:
       mov ax,es
       mov di,offset BU_MEM+4
       call WRD_TO_HEX
       mov ax,es:[01h]
       mov di,offset BU_MEM+14
       call WRD_TO_HEX
       mov ax,es:[03h]
       mov si,offset BU_MEM+26
       mov dx, 0
       mov bx, 10h
       mul bx
       call TO_DEC
       mov dx,offset BU_MEM
       call PRINT
       mov cx,8
       mov bx,8
       mov ah,02h
       CYCLE2:
             mov dl,es:[bx]
             add bx,1
             int 21h
       loop CYCLE2
       mov dx,offset STRENDL
       call PRINT
       mov ax,es
       add ax,1
       add ax,es:[03h]
       mov bl,es:[00h]
       mov es,ax

       push bx
       mov ax,'  '
       mov bx,offset BU_MEM
       mov [bx+19],ax
       mov [bx+21],ax
       mov [bx+23],ax
       pop bx

       cmp bl,4Dh
       je CYCLE
 pop es
 ret
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
 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
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

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

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
TO_DEC ENDP

BEGIN:
 call DOST
 call RAS
 call MCB
 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

DOST_MEM_STR db 'Доступная память:'
DOST_MEM_STR_ db '       байт',0DH,0AH,'$'
RASS_MEM_STR db 'Расширенная память:'
RASS_MEM_STR_ db '      Кбайт',0DH,0AH,'$'
BU_MEM_STR   db 'Адресс Владелец   Размер   Наименование',0DH,0AH,'$'
BU_MEM       db '                             $'
STRENDL db 0DH,0AH,'$'
ERROR_STR    db 'ОШИБКА',0DH,0AH,'$'


;--------------------------------------
PRINT PROC
 push ax
 mov ah,09h
 int 21h
 pop ax
 ret
PRINT ENDP
;--------------------------------------
DOST PROC
 mov ax,0
 mov ah,4Ah
 mov bx,0FFFFh
 int 21h
 mov ax,bx
 mov bx,16
 mul bx
 mov si,offset DOST_MEM_STR_+5
 call TO_DEC
 mov dx,offset DOST_MEM_STR
 call PRINT
 ret
DOST ENDP
;--------------------------------------
RAS PROC

 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 RASS_MEM_STR_+4
 call TO_DEC
 mov dx,offset RASS_MEM_STR
```

```
 call PRINT

 ret
RAS ENDP
;--------------------------------------
MCB PROC
 mov bx,0A000h
 mov ax,offset ENDD ;
 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_STR
       call PRINT
 ERROR:
 mov dx,offset BU_MEM_STR
 call PRINT
 push es
 mov ah,52h
 int 21h
 mov bx,es:[bx-2]
 mov es,bx

 CYCLE:

       mov ax,es
       mov di,offset BU_MEM+4
       call WRD_TO_HEX
       mov ax,es:[01h]
       mov di,offset BU_MEM+14
       call WRD_TO_HEX
       mov ax,es:[03h]
       mov si,offset BU_MEM+26
       mov dx, 0
       mov bx, 10h
       mul bx
       call TO_DEC
       mov dx,offset BU_MEM
       call PRINT
       mov cx,8
       mov bx,8
       mov ah,02h
       CYCLE2:
             mov dl,es:[bx]
             add bx,1
             int 21h
       loop CYCLE2
       mov dx,offset STRENDL
       call PRINT
       mov ax,es
       add ax,1
       add ax,es:[03h]
```

```
        mov bl,es:[00h]
        mov es,ax

        push bx
        mov ax,'  '
        mov bx,offset BU_MEM
        mov [bx+19],ax
        mov [bx+21],ax
        mov [bx+23],ax
        pop bx

        cmp bl,4Dh
        je CYCLE
  pop es
  ret
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
  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
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

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

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
TO_DEC ENDP

BEGIN:
 call DOST
 call RAS
 call MCB
 xor AL,AL
 mov AH,4Ch
 int 21H
ENDD:
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

DOST_MEM_STR db 'Доступная память:'
DOST_MEM_STR_ db '       байт',0DH,0AH,'$'
RASS_MEM_STR db 'Расширенная память:'
RASS_MEM_STR_ db '      Кбайт',0DH,0AH,'$'
BU_MEM_STR   db 'Адресс Владелец   Размер   Наименование',0DH,0AH,'$'
BU_MEM       db '                             $'
STRENDL db 0DH,0AH,'$'
ERROR_STR    db 'ОШИБКА',0DH,0AH,'$'


;--------------------------------------
PRINT PROC
 push ax
 mov ah,09h
 int 21h
 pop ax
 ret
PRINT ENDP
;--------------------------------------
DOST PROC
 mov ax,0
 mov ah,4Ah
 mov bx,0FFFFh
 int 21h
 mov ax,bx
 mov bx,16
 mul bx
 mov si,offset DOST_MEM_STR_+5
 call TO_DEC
 mov dx,offset DOST_MEM_STR
 call PRINT
 ret
DOST ENDP
;--------------------------------------
RAS PROC

 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 RASS_MEM_STR_+4
 call TO_DEC
 mov dx,offset RASS_MEM_STR
 call PRINT
```

```
 ret
RAS ENDP
;-------------------------------------
MCB PROC
 mov bx,0A000h
 mov ax,offset ENDD
 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_STR
       call PRINT
 ERROR:

 mov bx,1000h
 mov ah,48h
 int 21h

 mov dx,offset BU_MEM_STR
 call PRINT
 push es
 mov ah,52h
 int 21h
 mov bx,es:[bx-2]
 mov es,bx
 CYCLE:
       mov ax,es
       mov di,offset BU_MEM+4
       call WRD_TO_HEX
       mov ax,es:[01h]
       mov di,offset BU_MEM+14
       call WRD_TO_HEX
       mov ax,es:[03h]
       mov si,offset BU_MEM+26
       mov dx, 0
       mov bx, 10h
       mul bx
       call TO_DEC
       mov dx,offset BU_MEM
       call PRINT
       mov cx,8
       mov bx,8
       mov ah,02h
       CYCLE2:
             mov dl,es:[bx]
             add bx,1
             int 21h
       loop CYCLE2
       mov dx,offset STRENDL
       call PRINT
       mov ax,es
```

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

        push bx
        mov ax,'  '
        mov bx,offset BU_MEM
        mov [bx+19],ax
        mov [bx+21],ax
        mov [bx+23],ax
        pop bx

        cmp bl,4Dh
        je CYCLE
 pop es
 ret
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
 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
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

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

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
TO_DEC ENDP

BEGIN:
 call DOST
 call RAS
 call MCB
 xor AL,AL
 mov AH,4Ch
 int 21H
ENDD:
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

DOST_MEM_STR db 'Доступная память:'
DOST_MEM_STR_ db '       байт',0DH,0AH,'$'
RASS_MEM_STR db 'Расширенная память:'
RASS_MEM_STR_ db '      Кбайт',0DH,0AH,'$'
BU_MEM_STR   db 'Адресс Владелец   Размер   Наименование',0DH,0AH,'$'
ERROR_STR    db 'ОШИБКА',0DH,0AH,'$'
ERROR_MEM db 'Ошибка при выделении памяти',0DH,0AH,'$'
BU_MEM       db '                                $'
STRENDL db 0DH,0AH,'$'


;--------------------------------------
PRINT PROC
 push ax
 mov ah,09h
 int 21h
 pop ax
 ret
PRINT ENDP
;--------------------------------------
DOST PROC
 mov ax,0
 mov ah,4Ah
 mov bx,0FFFFh
 int 21h
 mov ax,bx
 mov bx,16
 mul bx
 mov si,offset DOST_MEM_STR_+5
 call TO_DEC
 mov dx,offset DOST_MEM_STR
 call PRINT
 ret
DOST ENDP
;--------------------------------------
RAS PROC

 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 RASS_MEM_STR_+4
 call TO_DEC
 mov dx,offset RASS_MEM_STR
```

```
 call PRINT

 ret
RAS ENDP
;--------------------------------------
MCB PROC
 mov bx,1000h
 mov ah,48h
 int 21h
 jnc ERROR
       mov dx,offset ERROR_MEM
       call PRINT
 ERROR:

 mov bx,0A000h
 mov ax,offset ENDD
 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 ERROR2
       mov dx,offset ERROR_STR
       call PRINT
 ERROR2:

 mov dx,offset BU_MEM_STR
 call PRINT
 push es
 mov ah,52h
 int 21h
 mov bx,es:[bx-2]
 mov es,bx
 CYCLE:
       mov ax,es
       mov di,offset BU_MEM+4
       call WRD_TO_HEX
       mov ax,es:[01h]
       mov di,offset BU_MEM+14
       call WRD_TO_HEX
       mov ax,es:[03h]
       mov si,offset BU_MEM+26
       mov dx, 0
       mov bx, 10h
       mul bx
       call TO_DEC
       mov dx,offset BU_MEM
       call PRINT
       mov cx,8
       mov bx,8
       mov ah,02h
       CYCLE2:
             mov dl,es:[bx]
             add bx,1
```

```
            int 21h
        loop CYCLE2
        mov dx,offset STRENDL
        call PRINT
        mov ax,es
        add ax,1
        add ax,es:[03h]
        mov bl,es:[00h]
        mov es,ax

        push bx
        mov ax,'  '
        mov bx,offset BU_MEM
        mov [bx+19],ax
        mov [bx+21],ax
        mov [bx+23],ax
        pop bx

        cmp bl,4Dh
        je CYCLE
  pop es
  ret
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
  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
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

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

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
TO_DEC ENDP

BEGIN:
 call DOST
 call RAS
 call MCB
 xor AL,AL
 mov AH,4Ch
 int 21H
ENDD:
TESTPC ENDS
 END START
```