МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Операционные системы»

ТЕМА: ИССЛЕДОВАНИЕ ОРГАНИЗАЦИИ УПРАВЛЕНИЯ ОСНОВНОЙ ПАМЯТЬЮ.

Студент гр. 9383 _____ Рыбников Р.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

ПОСТАНОВКА ЗАДАЧИ

Цель работы.

Исследование организации управления основной памятью, изучение нестраничной памяти, исследование структур данных и работы функций управления памятью ядра операционной системы.

Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

1. Количество доступной памяти.
2. Размер расширенной памяти.
3. 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. Запустите программу и внимательно оцените результаты. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 2. Измените программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah прерывания 21h (пример в разделе «Использование функции 4AH»). Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущем шаге. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 3. Измените программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H. Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущих шагах. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 4. Измените первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти. Обязательно обрабатывайте завершение функций ядра, проверяя флаг CF. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 5. Оцените результаты, полученные на предыдущих шагах. Ответьте на контрольные вопросы и оформите отчет.

РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ ПРОБЛЕМ

Шаг 1.

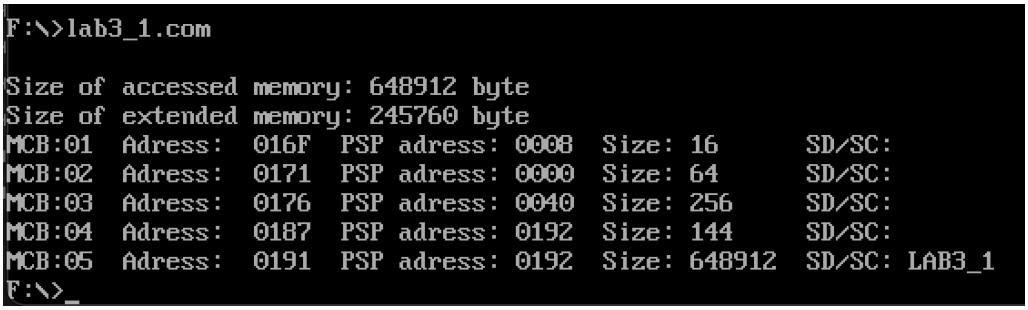


Рисунок 1 -- Работа .COM модуля первого файла.

Шаг 2.

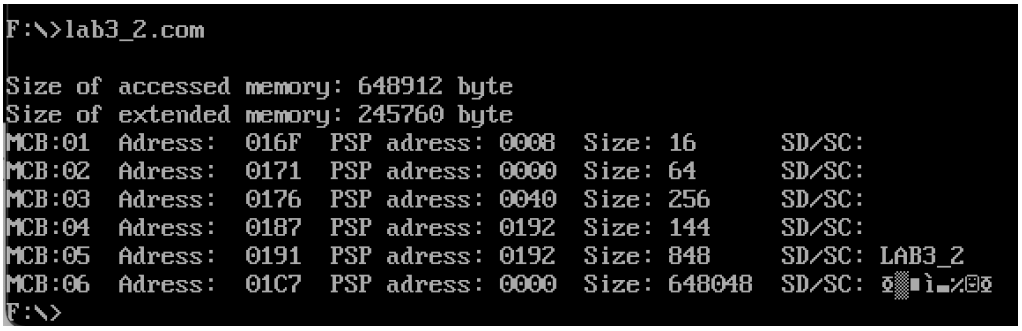
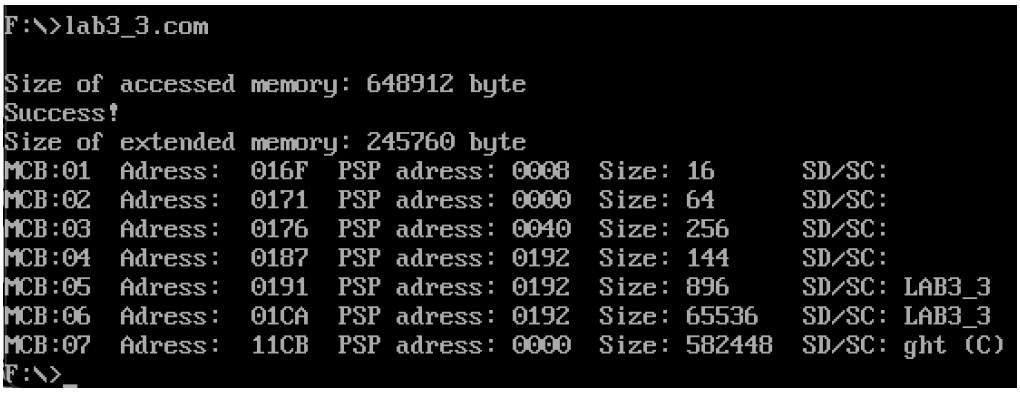


Рисунок 2 -- Работа .COM модуля второго файла.

Шаг 3.



Работа 3 -- Работа .COM модуля третьего файла.

Шаг 4.

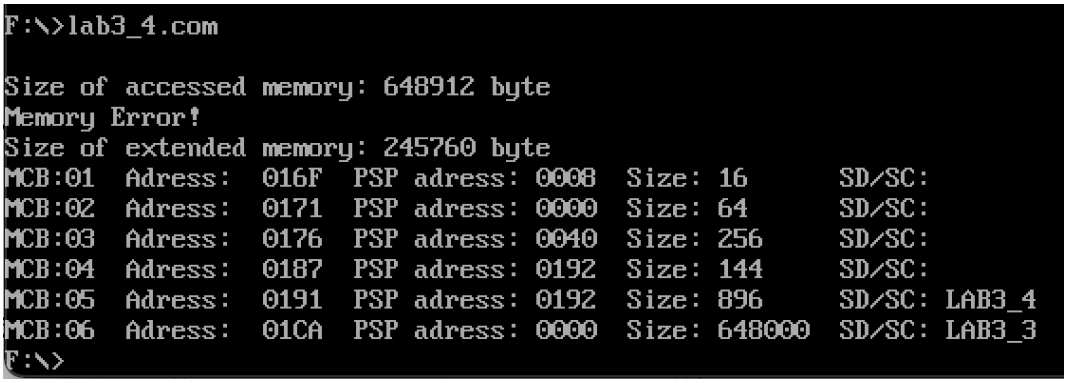


Рисунок 4 -- Работа .COM модуля четвёртого файла.

Шаг 5. Была произведена оценка результатов и были отвечены контрольные вопросы:

1. ***Что означает "доступный объем памяти"?***

Доступный объем памяти – это часть оперативной памяти, которая занимает и использует программа.

2. ***Где MCB блок Вашей программы в списке?***

MCB блок в первом файле находится внизу списка.

MCB блок во втором файле находится внизу списка, но на предпоследнем месте.

MCB блок в третьем файле стоит на третьем снизу по счету месте.

MCB блок в четвертом файле находится опять предпоследним.

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

В первом файле программа занимает всю память, которая ей доступна.

Во втором файле программа занимает столько памяти, сколько ей нужно.(832)

В третьем файле программа занимает нужный ей объем памяти + дополнительно выделенную память.(896+65536)

В четвертом файле только необходимый программе объем (896)

Выводы.

Были исследованы структуры данных и работы функций управления памятью ядра операционной системы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл lab3_2.asm

```
TESTPC SEGMENT
   ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
   ORG 100H
START: JMP BEGIN
ACCESSED_MEMORY db 13,10,'Size of accessed memory:       $'
EXTENDED_MEMORY db 13,10,'Size of extended memory:       $'
STR_BYTE db ' byte $'
STR_MCB db 13,10,'MCB:0   $'
ADRESS db 'Adress:        $'
ADRESS_PSP db 'PSP adress:       $'
STR_SIZE db 'Size:        $'
MCB_SD_SC db ' SD/SC: $'
STR_ERROR db 13,10,'Memory Error!$'
STR_SUCCECC db 13,10,'Success!$'

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 ;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 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
;--------------------------------
WRITE_STRING PROC near
   push AX
   mov AH,09h
   int 21h
   pop AX
   ret
WRITE_STRING ENDP
;--------------------------------

WRITE_SIZE PROC
   push ax
   push bx
   push cx
   push dx
   push si

     mov bx,10h
     mul bx
```

```
     mov bx,0ah
     xor cx,cx
delenie:
     div bx
     push dx
     inc cx
     xor dx,dx
     cmp ax,0h
     jnz delenie

write_symbol:
     pop dx
     or dl,30h
     mov [si], dl
     inc si
     loop write_symbol

   pop si
   pop dx
   pop cx
   pop bx
   pop ax
     ret
WRITE_SIZE ENDP
;-------------------------------
PRINT_MCB PROC
   push ax
   push bx
   push cx
   push dx
   push si

   mov ah,52h
   int 21h
   mov ax,es:[bx-2]
   mov es,ax
   xor cx,cx

      inc cx
pargraph_MCB:
     lea si, STR_MCB
     add si, 7
     mov al,cl
     push cx
     call BYTE_TO_DEC
     lea dx, STR_MCB
     call WRITE_STRING

     mov ax,es
     lea di,ADRESS
     add di,12
     call WRD_TO_HEX
     lea dx,ADRESS
```

```
      call WRITE_STRING

      xor ah,ah
      mov al,es:[0]
      push ax
      mov ax,es:[1]
      lea di, ADRESS_PSP
      add di, 15
      call WRD_TO_HEX
      lea dx, ADRESS_PSP
      call WRITE_STRING
      mov ax,es:[3]
      lea si,STR_SIZE
      add si, 6
      call WRITE_SIZE
      lea dx,STR_SIZE
      call WRITE_STRING
      xor dx, dx
      lea dx , MCB_SD_SC
      call WRITE_STRING
      mov cx,8
      xor di,di

write_char:
      mov dl,es:[di+8]
      mov ah,02h
      int 21h
      inc di
      loop write_char

      mov ax,es:[3]
      mov bx,es
      add bx,ax
      inc bx
      mov es,bx
      pop ax
      pop cx
      inc cx
      cmp al,5Ah
      je exit
      cmp al,4Dh
      jne exit
      jmp pargraph_MCB

      exit:
    pop si
    pop dx
    pop cx
    pop bx
    pop ax
      ret
PRINT_MCB ENDP
```

```
BEGIN:
   mov ah,4ah
      mov bx,0ffffh
      int 21h
   mov ax,bx
   lea si, ACCESSED_MEMORY
   add si, 27
   call WRITE_SIZE
   lea dx, ACCESSED_MEMORY
   call WRITE_STRING
   lea dx,STR_BYTE
   call WRITE_STRING

   ; расширенная память
   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
      lea si,EXTENDED_MEMORY
      add si, 27
      call WRITE_SIZE
      lea dx,EXTENDED_MEMORY
      call WRITE_STRING
      lea dx,STR_BYTE
   call WRITE_STRING

   ;MCB
   call PRINT_MCB

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

Файл lab3_2.asm

```
TESTPC SEGMENT
   ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
   ORG 100H
START: JMP BEGIN
ACCESSED_MEMORY db 13,10,'Size of accessed memory:       $'
EXTENDED_MEMORY db 13,10,'Size of extended memory:       $'
STR_BYTE db ' byte $'
STR_MCB db 13,10,'MCB:0   $'
```

```
ADRESS db 'Adress:        $'
ADRESS_PSP db 'PSP adress:       $'
STR_SIZE db 'Size:        $'
MCB_SD_SC db ' SD/SC: $'
STR_ERROR db 13,10,'Memory Error!$'
STR_SUCCECC db 13,10,'Success!$'

;------------------------------
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
;-------------------------------
WRITE_STRING PROC near
   push AX
   mov AH,09h
   int 21h
   pop AX
   ret
WRITE_STRING ENDP
;-------------------------------

WRITE_SIZE PROC
  push ax
  push bx
  push cx
  push dx
  push si

  mov bx,10h
     mul bx
     mov bx,0ah
     xor cx,cx
delenie:
     div bx
     push dx
     inc cx
     xor dx,dx
     cmp ax,0h
     jnz delenie

write_symbol:
     pop dx
     or dl,30h
     mov [si], dl
     inc si
     loop write_symbol
```

```
  pop si
  pop dx
  pop cx
  pop bx
  pop ax
     ret
WRITE_SIZE ENDP
;-------------------------------
PRINT_MCB PROC
   push ax
   push bx
   push cx
   push dx
   push si

   mov ah,52h
   int 21h
   mov ax,es:[bx-2]
   mov es,ax
   xor cx,cx

     inc cx
pargraph_MCB:
     lea si, STR_MCB
     add si, 7
     mov al,cl
     push cx
     call BYTE_TO_DEC
     lea dx, STR_MCB
     call WRITE_STRING

     mov ax,es
     lea di,ADRESS
     add di,12
     call WRD_TO_HEX
     lea dx,ADRESS
     call WRITE_STRING

     xor ah,ah
     mov al,es:[0]
     push ax
     mov ax,es:[1]
     lea di, ADRESS_PSP
     add di, 15
     call WRD_TO_HEX
     lea dx, ADRESS_PSP
     call WRITE_STRING
     mov ax,es:[3]
     lea si,STR_SIZE
     add si, 6
     call WRITE_SIZE
     lea dx,STR_SIZE
     call WRITE_STRING
```

```
      xor dx, dx
      lea dx , MCB_SD_SC
      call WRITE_STRING
      mov cx,8
      xor di,di

write_char:
      mov dl,es:[di+8]
      mov ah,02h
      int 21h
      inc di
      loop write_char

      mov ax,es:[3]
      mov bx,es
      add bx,ax
      inc bx
      mov es,bx
      pop ax
      pop cx
      inc cx
      cmp al,5Ah
      je exit
      cmp al,4Dh
      jne exit
      jmp pargraph_MCB

      exit:
  pop si
  pop dx
  pop cx
  pop bx
  pop ax
      ret
PRINT_MCB ENDP

FREE_UNUSED_MEMORY PROC
  push ax
  push bx
  push cx
  push dx

  lea ax, end_point
  mov bx,10h
  xor dx,dx
  div bx
  inc ax
  mov bx,ax
  mov al,0
  mov ah,4Ah
  int 21h

  pop dx
```

```
  pop cx
  pop bx
  pop ax
  ret
FREE_UNUSED_MEMORY ENDP


BEGIN:

     mov ah,4ah
     mov bx,0ffffh
     int 21h
  mov ax,bx
  lea si, ACCESSED_MEMORY
  add si, 27
  call WRITE_SIZE
  lea dx, ACCESSED_MEMORY
  call WRITE_STRING
  lea dx,STR_BYTE
  call WRITE_STRING

  call FREE_UNUSED_MEMORY

  ; расширенная память
  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
     lea si,EXTENDED_MEMORY
     add si, 27
     call WRITE_SIZE
     lea dx,EXTENDED_MEMORY
     call WRITE_STRING
     lea dx,STR_BYTE
  call WRITE_STRING


  call PRINT_MCB

  xor AL,AL
  mov AH,4Ch
  int 21H
end_point:
TESTPC ENDS
END START
```

Файл lab3_2.asm

```
TESTPC SEGMENT
   ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
   ORG 100H
START: JMP BEGIN
ACCESSED_MEMORY db 13,10,'Size of accessed memory:       $'
EXTENDED_MEMORY db 13,10,'Size of extended memory:       $'
STR_BYTE db ' byte $'
STR_MCB db 13,10,'MCB:0   $'
ADRESS db 'Adress:        $'
ADRESS_PSP db 'PSP adress:       $'
STR_SIZE db 'Size:        $'
MCB_SD_SC db ' SD/SC: $'
STR_ERROR db 13,10,'Memory Error!$'
STR_SUCCECC db 13,10,'Success!$'


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
;--------------------------------
WRITE_STRING PROC near
   push AX
   mov AH,09h
   int 21h
   pop AX
   ret
WRITE_STRING ENDP
;--------------------------------

WRITE_SIZE PROC near
  push ax
  push bx
  push cx
  push dx
  push si

     mov bx,10h
     mul bx
     mov bx,0ah
     xor cx,cx
delenie:
     div bx
     push dx
     inc cx
     xor dx,dx
```

```
     cmp ax,0h
     jnz delenie

write_symbol:
     pop dx
     or dl,30h
     mov [si], dl
     inc si
     loop write_symbol

  pop si
  pop dx
  pop cx
  pop bx
  pop ax
     ret
WRITE_SIZE ENDP
;-------------------------------
PRINT_MCB PROC near
  push ax
  push bx
  push cx
  push dx
  push si

  mov ah,52h
  int 21h
  mov ax,es:[bx-2]
  mov es,ax
  xor cx,cx

     inc cx
pargraph_MCB:
     lea si, STR_MCB
     add si, 7
     mov al,cl
     push cx
     call BYTE_TO_DEC
     lea dx, STR_MCB
     call WRITE_STRING

     mov ax,es
     lea di,ADRESS
     add di,12
     call WRD_TO_HEX
     lea dx,ADRESS
     call WRITE_STRING

     xor ah,ah
     mov al,es:[0]
     push ax
     mov ax,es:[1]
     lea di, ADRESS_PSP
```

```
     add di, 15
     call WRD_TO_HEX
     lea dx, ADRESS_PSP
     call WRITE_STRING
     mov ax,es:[3]
     lea si,STR_SIZE
     add si, 6
     call WRITE_SIZE
     lea dx,STR_SIZE
     call WRITE_STRING
     xor dx, dx
     lea dx , MCB_SD_SC
     call WRITE_STRING
     mov cx,8
     xor di,di

write_char:
     mov dl,es:[di+8]
     mov ah,02h
     int 21h
     inc di
     loop write_char

     mov ax,es:[3]
     mov bx,es
     add bx,ax
     inc bx
     mov es,bx
     pop ax
     pop cx
     inc cx
     cmp al,5Ah
     je exit
     cmp al,4Dh
     jne exit
     jmp pargraph_MCB

exit:
  pop si
  pop dx
  pop cx
  pop bx
  pop ax
     ret
PRINT_MCB ENDP

FREE_UNUSED_MEMORY PROC near
  push ax
  push bx
  push cx
  push dx

  lea ax, end_point
```

```
  mov bx,10h
  xor dx,dx
  div bx
  inc ax
  mov bx,ax
  mov al,0
  mov ah,4Ah
  int 21h

  pop dx
  pop cx
  pop bx
  pop ax
  ret
FREE_UNUSED_MEMORY ENDP

GET_EXTRA_MEMORY PROC near
  push ax
  push bx
  push dx

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

  jc ERROR1
  jmp SUCCECC

ERROR1:
  lea dx,STR_ERROR
  call WRITE_STRING
  JMP END1
SUCCECC:
   lea dx,STR_SUCCECC
   call WRITE_STRING
END1:
   pop dx
   pop bx
   pop ax
   ret
GET_EXTRA_MEMORY ENDP

BEGIN:

     mov ah,4ah
     mov bx,0ffffh
     int 21h
  mov ax,bx
  lea si, ACCESSED_MEMORY
  add si, 27
  call WRITE_SIZE
  lea dx, ACCESSED_MEMORY
  call WRITE_STRING
```

```
  lea dx,STR_BYTE
  call WRITE_STRING

  call FREE_UNUSED_MEMORY
  call GET_EXTRA_MEMORY

   ; расширенная память
   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
     lea si,EXTENDED_MEMORY
     add si, 27
     call WRITE_SIZE
     lea dx,EXTENDED_MEMORY
     call WRITE_STRING
     lea dx,STR_BYTE
  call WRITE_STRING

   ;MCB
   call PRINT_MCB

   xor AL,AL
   mov AH,4Ch
   int 21H
end_point:
TESTPC ENDS
END START
```

Файл lab3_4.asm

```
TESTPC SEGMENT
   ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
   ORG 100H
START: JMP BEGIN
ACCESSED_MEMORY db 13,10,'Size of accessed memory:       $'
EXTENDED_MEMORY db 13,10,'Size of extended memory:       $'
STR_BYTE db ' byte $'
STR_MCB db 13,10,'MCB:0   $'
ADRESS db 'Adress:        $'
ADRESS_PSP db 'PSP adress:       $'
STR_SIZE db 'Size:        $'
MCB_SD_SC db ' SD/SC: $'
STR_ERROR db 13,10,'Memory Error!$'
STR_SUCCECC db 13,10,'Success!$'
```

```
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
;-------------------------------
WRITE_STRING PROC near
   push AX
   mov AH,09h
   int 21h
   pop AX
   ret
WRITE_STRING ENDP
;-------------------------------

WRITE_SIZE PROC near
  push ax
  push bx
  push cx
  push dx
  push si

     mov bx,10h
     mul bx
     mov bx,0ah
     xor cx,cx
delenie:
     div bx
     push dx
     inc cx
     xor dx,dx
     cmp ax,0h
     jnz delenie

write_symbol:
     pop dx
     or dl,30h
     mov [si], dl
     inc si
     loop write_symbol

  pop si
  pop dx
  pop cx
  pop bx
  pop ax
     ret
WRITE_SIZE ENDP
```

```
;------------------------------
PRINT_MCB PROC near
  push ax
  push bx
  push cx
  push dx
  push si

  mov ah,52h
  int 21h
  mov ax,es:[bx-2]
  mov es,ax
  xor cx,cx

     inc cx
pargraph_MCB:
     lea si, STR_MCB
     add si, 7
     mov al,cl
     push cx
     call BYTE_TO_DEC
     lea dx, STR_MCB
     call WRITE_STRING

     mov ax,es
     lea di,ADRESS
     add di,12
     call WRD_TO_HEX
     lea dx,ADRESS
     call WRITE_STRING

     xor ah,ah
     mov al,es:[0]
     push ax
     mov ax,es:[1]
     lea di, ADRESS_PSP
     add di, 15
     call WRD_TO_HEX
     lea dx, ADRESS_PSP
     call WRITE_STRING
     mov ax,es:[3]
     lea si,STR_SIZE
     add si, 6
     call WRITE_SIZE
     lea dx,STR_SIZE
     call WRITE_STRING
     xor dx, dx
     lea dx , MCB_SD_SC
     call WRITE_STRING
     mov cx,8
     xor di,di

write_char:
```

```
      mov dl,es:[di+8]
      mov ah,02h
      int 21h
      inc di
      loop write_char

      mov ax,es:[3]
      mov bx,es
      add bx,ax
      inc bx
      mov es,bx
      pop ax
      pop cx
      inc cx
      cmp al,5Ah
      je exit
      cmp al,4Dh
      jne exit
      jmp pargraph_MCB

exit:
  pop si
  pop dx
  pop cx
  pop bx
  pop ax
     ret
PRINT_MCB ENDP

FREE_UNUSED_MEMORY PROC near
  push ax
  push bx
  push cx
  push dx

  lea ax, end_point
  mov bx,10h
  xor dx,dx
  div bx
  inc ax
  mov bx,ax
  mov al,0
  mov ah,4Ah
  int 21h

  pop dx
  pop cx
  pop bx
  pop ax
  ret
FREE_UNUSED_MEMORY ENDP

GET_EXTRA_MEMORY PROC near
```

```
  push ax
  push bx
  push dx

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

  jc ERROR1
  jmp SUCCECC

ERROR1:
  lea dx,STR_ERROR
  call WRITE_STRING
  JMP END1
SUCCECC:
  lea dx,STR_SUCCECC
  call WRITE_STRING
END1:
  pop dx
  pop bx
  pop ax
  ret
GET_EXTRA_MEMORY ENDP


BEGIN:
     mov ah,4ah
     mov bx,0ffffh
     int 21h
  mov ax,bx
  lea si, ACCESSED_MEMORY
  add si, 27
  call WRITE_SIZE
  lea dx, ACCESSED_MEMORY
  call WRITE_STRING
  lea dx,STR_BYTE
  call WRITE_STRING

  call GET_EXTRA_MEMORY
  call FREE_UNUSED_MEMORY


   ; расширенная память
   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
   lea si,EXTENDED_MEMORY
      add si, 27
      call WRITE_SIZE
      lea dx,EXTENDED_MEMORY
      call WRITE_STRING
      lea dx,STR_BYTE
   call WRITE_STRING

   ;MCB
   call PRINT_MCB

   xor AL,AL
   mov AH,4Ch
   int 21H
end_point:
TESTPC ENDS
END START
```