ОТЧЕТ

по лабораторной работе №3

по дисциплине «Операционные системы»

Тема: Исследование организации управления основной памятью

Студент гр. 8383	_____	Дейнега В.Е.
Преподаватель	_____	Ефремов М.А,

Санкт-Петербург

2020

Цель работы.

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

Выполнение.

Написан код .COM модуля, который выбирает и выводит информацию о количестве доступной памяти, размер расширенной памяти, цепочку блоков управления памятью. Результат работы представлен на рис. 1, исходный код представлен в приложении А.
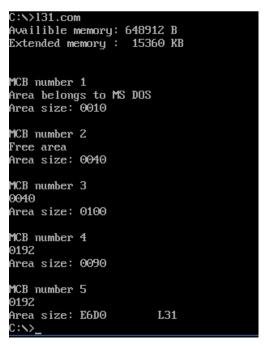

Рисунок 1 – Выполнение первой программы

Далее программа была модифицирована, теперь она освобождает всю неиспользуемую ей память. Результат работы представлен на рис. 2, исходный код представлен в приложении А.

Рисунок 2 – Выполнение второй программы

Далее программа была изменена таким образом, чтобы после освобождения памяти, программа запрашивала дополнительно 64 Кб памяти. Результат работы представлен на рис. 3, исходный код – приложение А.

Рисунок 3 – Выполнение третьей программы

Далее изначальная программа модифицируется таким образом, чтобы запрашивать память перед ее очищением. Результат работы представлен в приложении на рисунке 4, исходный код – приложение А.

Рисунок 4 – Выполнение четвертой программы

Контрольные вопросы.

1. Что означает «доступный объем памяти»?

Максимальный объем памяти, который может быть доступен для запуска и выполнения программ.

2. Где MCB блок вашей программы в списке?

В 1, 2 и 4-й программе MCB блок программы – пятый, в 3-й еще и шестой блок, для дополнительно выделенной памяти.

3. Какой размер памяти занимает программа в каждом случае?

1-я программа занимает 648912 + 144 байт

2-я программа занимает 3ED0 + 144 байт

3-я программа занимает 3F40 + 144 байт, а после выделения занимает дополнительно 64кБ

4-я программа занимает 3FD0 + 144 байт.

Выводы.

В ходе выполнения лабораторной работы был изучен список блоков управления памятью, а также методы выделения и освобождения памяти для программы.

ПРИЛОЖЕНИЕ А

L31.ASM

```
TESTPC SEGMENT
        ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        org 100H

START:  JMP BEGIN

A_MEMORY db 'Availible memory:       B           ',0Dh,0Ah,'$'
E_MEMORY db 'Extended memory :       KB          ',0Dh,0Ah,'$'

STR_NUM db 13, 10, "MCB number      $"
ENDL db 13, 10, "$"
STR_386CB db 13, 10, "Area is busy by 386MAX UMB$"
STR_FREE db 13, 10, "Free area$"
STR_XMS db 13, 10, "Area belongs to OS XMS UMB$"
STR_TM db 13, 10, "Area is excluded driver's top memory$"
STR_DOS db 13, 10, "Area belongs to MS DOS$"
STR_386B db 13, 10, "Area is blocked by 386MAX$"
STR_386 db 13, 10, "Area belongs to the 386MAX UMB$"
STR_PSP db 13, 10, "      $"
STR_SIZE db 13, 10, "Area size:            $"
;------------------------------------------
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

                WRD_TO_DEC PROC near
                   push CX
                   push DX
                   mov CX,10
                metka1: div CX
                   or DL,30h
                   mov [SI],DL
                   dec SI
                   xor DX,DX
                   cmp AX,10
                   jae metka1
                   cmp AL,00h
                   je end_pr
                   or AL,30h
                   mov [SI],AL
                end_pr:  pop DX
                   pop CX
                   ret
                WRD_TO_DEC 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

                ;------------------------------------------------------------
                A_MEM     PROC near
                   mov ah, 4ah
                   mov bx, 0ffffh
                   int 21h
                   mov ax, 10h
                   mul bx
                   mov si, offset A_MEMORY
                   add si, 23
                   call WRD_TO_DEC
                     ret
                A_MEM     ENDP

                E_MEM     PROC near
                   xor dx,dx
                   mov al,30h
```

```asm
        out 70h,al
        in al,71h
        mov bl,AL
        mov al,31h
        out 70h,al
        in al,71h

    mov ah, al
    mov al, bl

    mov si, offset E_MEMORY
    add si, 23
    call WRD_TO_DEC
    ret
E_MEM   ENDP

MCB PROC near
    push ax
    push bx
    push cx
    push di
    push si

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

    next_mcb:
        inc cx
        mov dx, offset ENDL
        call WRITE_STR
        mov si, offset STR_NUM
        add si, 13
        mov ax, cx
        push cx
        call BYTE_TO_DEC
        mov dx, offset STR_NUM
        call WRITE_STR

        xor ax, ax
        mov al, es:[0h]
        push ax
        mov ax, es:[1h]

        cmp ax, 0h
        je if_free_area
        cmp ax, 6h
        je if_driver
        cmp ax, 7h
        je if_upper_memory
        cmp ax, 8h
        je if_msdos
        cmp ax, 0FFFAh
        je if_386max_umb
        cmp ax, 0FFFDh
        je if_block_386max
        cmp ax, 0FFFEh
        je if_belongs_386max
```

```
        xor dx, dx
        mov di, offset STR_PSP
        add di, 5
        call WRD_TO_HEX
        mov dx, offset  STR_PSP
        jmp end_of_01h
        pop ax

  if_free_area:
        mov dx, offset STR_FREE
        jmp end_of_01h

  if_driver:
        mov dx, offset STR_XMS
        jmp end_of_01h

  if_upper_memory:
        mov dx, offset STR_TM
        jmp end_of_01h

  if_msdos:
        mov dx, offset STR_DOS
        jmp end_of_01h

  if_386max_umb:
        mov dx, offset STR_386CB
        jmp end_of_01h

  if_block_386max:
        mov dx, offset STR_386B
        jmp end_of_01h

  if_belongs_386max:
        mov dx, offset STR_386
        jmp end_of_01h

  end_of_01h:
        call WRITE_STR
        mov di, offset STR_SIZE
        add di, 16
        mov ax, es:[3h]
        mov bx, 10h
        mul bx
        call WRD_TO_HEX
        mov dx, offset STR_SIZE
        call WRITE_STR
        mov cx, 8
        xor si, si

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

        mov ax, es:[3h]
        mov bx, es
        add bx, ax
```

```asm
        inc bx
        mov es, bx
        pop ax
        pop cx
        cmp al, 5Ah
        je end_of_proc
        jmp next_mcb

    end_of_proc:
        pop si
        pop di
        pop cx
        pop bx
        pop ax
    ret
MCB ENDP

WRITE_STR PROC near
    push ax
    mov ah,09h
    int  21h
    pop ax
    ret
WRITE_STR ENDP

BEGIN:
    call A_MEM
    mov dx, offset A_MEMORY
    call WRITE_STR

    call E_MEM
    mov dx, offset E_MEMORY
    call WRITE_STR
    call MCB
    xor AL, AL
    mov AH, 4Ch
    int 21h
TESTPC   ENDS
    END START
```

L32.ASM

```asm
TESTPC SEGMENT
        ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        org 100H

START:  JMP BEGIN

A_MEMORY db 'Availible memory:       B           ',0Dh,0Ah,'$'
E_MEMORY db 'Extended memory :       KB          ',0Dh,0Ah,'$'

STR_NUM db 13, 10, "MCB number      $"
;ENDL db 13, 10, "$"
STR_386CB db 13, 10, "Area is busy by 386MAX UMB$"
STR_FREE db 13, 10, "Free area$"
STR_XMS db 13, 10, "Area belongs to OS XMS UMB$"
STR_TM db 13, 10, "Area is excluded driver's top memory$"
STR_DOS db 13, 10, "Area belongs to MS DOS$"
STR_386B db 13, 10, "Area is blocked by 386MAX$"
```

```
STR_386 db 13, 10, "Area belongs to the 386MAX UMB$"
STR_PSP db 13, 10, "      $"
STR_SIZE db 13, 10, "Area size:          $"
;------------------------------------------
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

WRD_TO_DEC PROC near
    push CX
    push DX
    mov CX,10
metka1: div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp AX,10
    jae metka1
    cmp AL,00h
    je end_pr
    or AL,30h
    mov [SI],AL
end_pr:  pop DX
    pop CX
    ret
```

```
WRD_TO_DEC 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

;----------------------------------------------------------
A_MEM     PROC near
   mov ah, 4ah
   mov bx, 0ffffh
   int 21h
   mov ax, 10h
   mul bx
   mov si, offset A_MEMORY
   add si, 23
   call WRD_TO_DEC
     ret
A_MEM     ENDP

E_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 ah, al
   mov al, bl

   mov si, offset E_MEMORY
   add si, 23
   call WRD_TO_DEC
   ret
E_MEM     ENDP

MCB PROC near
   push ax
   push bx
   push cx
```

12

```asm
        push di
        push si

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

next_mcb:
        inc cx
;       mov dx, offset ENDL
;       call WRITE_STR
        mov si, offset STR_NUM
        add si, 13
        mov ax, cx
        push cx
        call BYTE_TO_DEC
        mov dx, offset STR_NUM
        call WRITE_STR

        xor ax, ax
        mov al, es:[0h]
        push ax
        mov ax, es:[1h]

        cmp ax, 0h
        je if_free_area
        cmp ax, 6h
        je if_driver
        cmp ax, 7h
        je if_upper_memory
        cmp ax, 8h
        je if_msdos
        cmp ax, 0FFFAh
        je if_386max_umb
        cmp ax, 0FFFDh
        je if_block_386max
        cmp ax, 0FFFEh
        je if_belongs_386max

        xor dx, dx
        mov di, offset STR_PSP
        add di, 5
        call WRD_TO_HEX
        mov dx, offset  STR_PSP
        jmp end_of_01h
        pop ax

if_free_area:
        mov dx, offset STR_FREE
        jmp end_of_01h

if_driver:
        mov dx, offset STR_XMS
        jmp end_of_01h

if_upper_memory:
        mov dx, offset STR_TM
        jmp end_of_01h
```

```
    if_msdos:
        mov dx, offset STR_DOS
        jmp end_of_01h

    if_386max_umb:
        mov dx, offset STR_386CB
        jmp end_of_01h

    if_block_386max:
        mov dx, offset STR_386B
        jmp end_of_01h

    if_belongs_386max:
        mov dx, offset STR_386
        jmp end_of_01h

    end_of_01h:
        call WRITE_STR
        mov di, offset STR_SIZE
        add di, 16
        mov ax, es:[3h]
        mov bx, 10h
        mul bx
        call WRD_TO_HEX
        mov dx, offset STR_SIZE
        call WRITE_STR
        mov cx, 8
        xor si, si

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

        mov ax, es:[3h]
        mov bx, es
        add bx, ax
        inc bx
        mov es, bx
        pop ax
        pop cx
        cmp al, 5Ah
        je end_of_proc
        jmp next_mcb

    end_of_proc:
        pop si
        pop di
        pop cx
        pop bx
        pop ax
    ret
MCB ENDP

WRITE_STR PROC near
    push ax
    mov ah,09h
```

```
    int  21h
    pop ax
    ret
WRITE_STR ENDP

BEGIN:
    call A_MEM
    mov dx, offset A_MEMORY
    call WRITE_STR

    mov ah, 4ah
    mov bx, offset end_size
    int 21h

    call E_MEM
    mov dx, offset E_MEMORY
    call WRITE_STR
    call MCB
    xor AL, AL
    mov AH, 4Ch
    int 21h
    end_size db 0
TESTPC   ENDS
     END START
```

L33.ASM

```
TESTPC SEGMENT
        ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        org 100H

START:  JMP BEGIN

A_MEMORY db 'Availible memory:      B          ',0Dh,0Ah,'$'
E_MEMORY db 'Extended memory :      KB          ',0Dh,0Ah,'$'

STR_NUM db 13, 10, "MCB number     $"
;ENDL db 13, 10, "$"
STR_386CB db 13, 10, "Area is busy by 386MAX UMB$"
STR_FREE db 13, 10, "Free area$"
STR_XMS db 13, 10, "Area belongs to OS XMS UMB$"
STR_TM db 13, 10, "Area is excluded driver's top memory$"
STR_DOS db 13, 10, "Area belongs to MS DOS$"
STR_386B db 13, 10, "Area is blocked by 386MAX$"
STR_386 db 13, 10, "Area belongs to the 386MAX UMB$"
STR_PSP db 13, 10, "     $"
STR_SIZE db 13, 10, "Area size:            $"
;-----------------------------------------
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

WRD_TO_DEC PROC near
      push CX
      push DX
      mov CX,10
metka1: div CX
      or DL,30h
      mov [SI],DL
      dec SI
      xor DX,DX
      cmp AX,10
      jae metka1
      cmp AL,00h
      je end_pr
      or AL,30h
      mov [SI],AL
end_pr:   pop DX
      pop CX
      ret
WRD_TO_DEC 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
```

```asm
    cmp AL,00h
    je end_l
    or AL,30h
    mov [SI],AL
end_l: pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

;--------------------------------------------------------------
A_MEM    PROC near
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h
    mov ax, 10h
    mul bx
    mov si, offset A_MEMORY
    add si, 23
    call WRD_TO_DEC
       ret
A_MEM    ENDP

E_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 ah, al
    mov al, bl

    mov si, offset E_MEMORY
    add si, 23
    call WRD_TO_DEC
    ret
E_MEM    ENDP

MCB PROC near
    push ax
    push bx
    push cx
    push di
    push si

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

    next_mcb:
      inc cx
    ; mov dx, offset ENDL
    ; call WRITE_STR
      mov si, offset STR_NUM
      add si, 13
```

```
    mov ax, cx
    push cx
    call BYTE_TO_DEC
    mov dx, offset STR_NUM
    call WRITE_STR

    xor ax, ax
    mov al, es:[0h]
    push ax
    mov ax, es:[1h]

    cmp ax, 0h
    je if_free_area
    cmp ax, 6h
    je if_driver
    cmp ax, 7h
    je if_upper_memory
    cmp ax, 8h
    je if_msdos
    cmp ax, 0FFFAh
    je if_386max_umb
    cmp ax, 0FFFDh
    je if_block_386max
    cmp ax, 0FFFEh
    je if_belongs_386max

    xor dx, dx
    mov di, offset STR_PSP
    add di, 5
    call WRD_TO_HEX
    mov dx, offset  STR_PSP
    jmp end_of_01h
    pop ax

if_free_area:
    mov dx, offset STR_FREE
    jmp end_of_01h

if_driver:
    mov dx, offset STR_XMS
    jmp end_of_01h

if_upper_memory:
    mov dx, offset STR_TM
    jmp end_of_01h

if_msdos:
    mov dx, offset STR_DOS
    jmp end_of_01h

if_386max_umb:
    mov dx, offset STR_386CB
    jmp end_of_01h

if_block_386max:
    mov dx, offset STR_386B
    jmp end_of_01h

if_belongs_386max:
    mov dx, offset STR_386
```

```
          jmp end_of_01h

      end_of_01h:
        call WRITE_STR
        mov di, offset STR_SIZE
        add di, 16
        mov ax, es:[3h]
        mov bx, 10h
        mul bx
        call WRD_TO_HEX
        mov dx, offset STR_SIZE
        call WRITE_STR
        mov cx, 8
        xor si, si

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

        mov ax, es:[3h]
        mov bx, es
        add bx, ax
        inc bx
        mov es, bx
        pop ax
        pop cx
        cmp al, 5Ah
        je end_of_proc
        jmp next_mcb

      end_of_proc:
        pop si
        pop di
        pop cx
        pop bx
        pop ax
      ret
MCB ENDP

WRITE_STR PROC near
    push ax
    mov ah,09h
    int  21h
    pop ax
    ret
WRITE_STR ENDP

BEGIN:
    call A_MEM
    mov dx, offset A_MEMORY
    call WRITE_STR

    mov ah, 4ah
    mov bx, offset end_size
    int 21h

    mov ah, 48h
```

```
  mov bx, 1000h
  int 21h


  call E_MEM
  mov dx, offset E_MEMORY
  call WRITE_STR
  call MCB
  xor AL, AL
  mov AH, 4Ch
  int 21h
  end_size db 0
TESTPC   ENDS
    END START
```

L34.ASM

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

A_MEMORY db 'Availible memory:       B          ',0Dh,0Ah,'$'
E_MEMORY db 'Extended memory :       KB          ',0Dh,0Ah,'$'

STR_NUM db 13, 10, "MCB number      $"
;ENDL db 13, 10, "$"
STR_386CB db 13, 10, "Area is busy by 386MAX UMB$"
STR_FREE db 13, 10, "Free area$"
STR_XMS db 13, 10, "Area belongs to OS XMS UMB$"
STR_TM db 13, 10, "Area is excluded driver's top memory$"
STR_DOS db 13, 10, "Area belongs to MS DOS$"
STR_ERR db 13, 10, "ERROR!$"
STR_386B db 13, 10, "Area is blocked by 386MAX$"
STR_386 db 13, 10, "Area belongs to the 386MAX UMB$"
STR_PSP db 13, 10, "     $"
STR_SIZE db 13, 10, "Area size:            $"
;-----------------------------------------
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
```

```asm
    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

WRD_TO_DEC PROC near
    push CX
    push DX
    mov CX,10
metka1: div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp AX,10
    jae metka1
    cmp AL,00h
    je end_pr
    or AL,30h
    mov [SI],AL
end_pr:  pop DX
    pop CX
    ret
WRD_TO_DEC 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

;-------------------------------------------------------------

A_MEM    PROC near
```

```asm
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h
    mov ax, 10h
    mul bx
    mov si, offset A_MEMORY
    add si, 23
    call WRD_TO_DEC
      ret
A_MEM    ENDP


E_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 ah, al
  mov al, bl

  mov si, offset E_MEMORY
  add si, 23
  call WRD_TO_DEC
  ret
E_MEM    ENDP

MCB PROC near
  push ax
  push bx
  push cx
  push di
  push si

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

  next_mcb:
    inc cx
    ;mov dx, offset ENDL
  ; call WRITE_STR
    mov si, offset STR_NUM
    add si, 13
    mov ax, cx
    push cx
    call BYTE_TO_DEC
    mov dx, offset STR_NUM
    call WRITE_STR

    xor ax, ax
    mov al, es:[0h]
    push ax
    mov ax, es:[1h]
```

```
    cmp ax, 0h
    je if_free_area
    cmp ax, 6h
    je if_driver
    cmp ax, 7h
    je if_upper_memory
    cmp ax, 8h
    je if_msdos
    cmp ax, 0FFFAh
    je if_386max_umb
    cmp ax, 0FFFDh
    je if_block_386max
    cmp ax, 0FFFEh
    je if_belongs_386max

    xor dx, dx
    mov di, offset STR_PSP
    add di, 5
    call WRD_TO_HEX
    mov dx, offset  STR_PSP
    jmp end_of_01h
    pop ax

if_free_area:
    mov dx, offset STR_FREE
    jmp end_of_01h

if_driver:
    mov dx, offset STR_XMS
    jmp end_of_01h

if_upper_memory:
    mov dx, offset STR_TM
    jmp end_of_01h

if_msdos:
    mov dx, offset STR_DOS
    jmp end_of_01h

if_386max_umb:
    mov dx, offset STR_386CB
    jmp end_of_01h

if_block_386max:
    mov dx, offset STR_386B
    jmp end_of_01h

if_belongs_386max:
    mov dx, offset STR_386
    jmp end_of_01h

end_of_01h:
    call WRITE_STR
    mov di, offset STR_SIZE
    add di, 16
    mov ax, es:[3h]
    mov bx, 10h
    mul bx
    call WRD_TO_HEX
    mov dx, offset STR_SIZE
```

```
      call WRITE_STR
      mov cx, 8
      xor si, si

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

      mov ax, es:[3h]
      mov bx, es
      add bx, ax
      inc bx
      mov es, bx
      pop ax
      pop cx
      cmp al, 5Ah
      je end_of_proc
      jmp next_mcb

   end_of_proc:
      pop si
      pop di
      pop cx
      pop bx
      pop ax
    ret
MCB ENDP

WRITE_STR PROC near
   push ax
   mov ah,09h
   int  21h
   pop ax
   ret
WRITE_STR ENDP

BEGIN:
   call A_MEM
   mov dx, offset A_MEMORY
   call WRITE_STR

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

   mov ah, 4ah
   mov bx, offset end_size
   int 21h

   call E_MEM
   mov dx, offset E_MEMORY
   call WRITE_STR
   call MCB
   xor AL, AL
   mov AH, 4Ch
   int 21h
   end_size db 0
```

```
    prog_end:
TESTPC    ENDS
      END START
```