

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

ЛАБОРАТОРНАЯ РАБОТА № 3
по дисциплине «Операционные системы»
ТЕМА: Исследование организации управления основной памятью.

Студентка гр. 9381

Москаленко Е.М.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Для исследования организации управления памятью. Необходимо ориентироваться на тип основной памяти, реализованный 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 этот список.

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

Функции:

Название	Описание
PRINT_SYMB	Вывод символа на экран (используя функцию DOS 02h, прерывания 21h).
TETR_TO_HEX	Перевод четырех младших битов регистре AL в 16-ричную цифру.
BYTE_TO_HEX	Перевод байта из AL в число 16-ной с.с. Символы записываются в регистры AL и AH.
WRD_TO_HEX	Перевод слова из AH в число в 16-ной с.с. Записывается в виде 4 символов по адресу из DI.
BYTE_TO_DEC	Перевод байта из AL в 16-ной с.с в число в 10-ной с.с.. Записывается по адресу, на который указывает SI (младшая цифра).
PRINT	Вывод строки на экран при помощи функции 9h прерывания 21h.

Выполнение работы.

1) Был написан и отлажен программный модуль .COM, который выбирает и распечатывает следующую информацию:

- Количество доступной памяти

- Размер расширенной памяти
- Выводит цепочку битов управления памятью

```
F:\> lab3_1.com
Available memory: 648912 b
Extended memory: 15360 Kb
List of MCB:
MCB type: 4Dh   PSP address: 0008h   Size:      16 b
MCB type: 4Dh   PSP address: 0000h   Size:      64 b
MCB type: 4Dh   PSP address: 0040h   Size:     256 b
MCB type: 4Dh   PSP address: 0192h   Size:     144 b
MCB type: 5Ah   PSP address: 0192h   Size:    648912 b      LAB3_1
```

2) Далее программа была изменена таким образом, что она освобождает память, которую она не занимает. Для этого использована функция 4Ah прерывания 21H. Результат программы представлен ниже.

```
F:\> lab3_2.com
Available memory: 648912 b
Extended memory: 15360 Kb
List of MCB:
MCB type: 4Dh   PSP address: 0008h   Size:      16 b
MCB type: 4Dh   PSP address: 0000h   Size:      64 b
MCB type: 4Dh   PSP address: 0040h   Size:     256 b
MCB type: 4Dh   PSP address: 0192h   Size:     144 b
MCB type: 4Dh   PSP address: 0192h   Size:      800 b      LAB3_2
MCB type: 5Ah   PSP address: 0000h   Size:    648096 b      Q& i0Y=
```

3) Затем программа была изменена еще раз таким способом, что после освобождения памяти она запрашивает 64Кб памяти функцией 48H прерывания 21H. Результат программы представлен ниже.

```
F:\>lab3_3.com
Available memory: 648912 b
Memory request succeeded
Extended memory: 15360 Kb
List of MCB:
MCB type: 4Dh   PSP address: 0008h   Size:      16 b
MCB type: 4Dh   PSP address: 0000h   Size:      64 b
MCB type: 4Dh   PSP address: 0040h   Size:     256 b
MCB type: 4Dh   PSP address: 0192h   Size:     144 b
MCB type: 4Dh   PSP address: 0192h   Size:      896 b      LAB3_3
MCB type: 4Dh   PSP address: 0192h   Size:    65536 b      LAB3_3
MCB type: 5Ah   PSP address: 0000h   Size:    582448 b      ght (C)
```

4) Программа была изменена таким образом, что запрашивает 64Кб памяти функцией 48H прерывания 21H до освобождения памяти. Результат программы представлен ниже.

```
F:\> lab3_4.com
Available memory: 648912 b
Memory request failed
Extended memory: 15360 Kb
List of MCB:
MCB type: 4Dh PSP address: 0008h Size: 16 b
MCB type: 4Dh PSP address: 0000h Size: 64 b
MCB type: 4Dh PSP address: 0040h Size: 256 b
MCB type: 4Dh PSP address: 0192h Size: 144 b
MCB type: 4Dh PSP address: 0192h Size: 896 b
MCB type: 5Ah PSP address: 0000h Size: 648000 b
```

Выводы:

В ходе выполнения лабораторной работы были освоены и применены функции управления памятью ядра операционной системы и изучены способы управления динамическими разделами памяти.

Ответы на контрольные вопросы:

1. Что означает “доступный объем памяти”?

Доступный объем памяти – это область основной памяти, выделенная программе.

2. Где MCB блок вашей программы в списке?

MCB блок моей программы PSP адрес (сегментный адрес владельца участка памяти) = 0192h. Такой же сегментный адрес имеет среда, принадлежащая программе.

3. Какой размер памяти занимает программа в каждом случае?

В 1-ой версии программы она занимает всю доступную память: 648912 б + 144 б = 649056 б.

Во 2-ой – только объем, занимаемый самой программой: $800 \text{ б} + 144 \text{ б} =$
944 б.

В 3-ей – $65536 + 896 \text{ б} + 144 \text{ б} =$ **66576 б**, 4кб выделили

В 4-ой – выделить 64кб невозможно: $896 \text{ б} + 144 \text{ б} =$ **1040 б.**

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

lab3_1.asm

```
TESTPC    segment
            ASSUME CS: TESTPC, DS: TESTPC, ES:nothing, SS:nothing
            ORG 100h

Start:
    jmp Begin

    Available_memory      db 'Available memory:          b$'
    Extended_memory db 'Extended memory:                Kb$'
    MCB                   db 'List of MCB:$'
    MCBtype db 'MCB type: 00h$'
    PSP_address           db 'PSP adress: 0000h$'
    size_s                db 'Size:                      b$'
    Endl                  db 13, 10, '$'
    Tab                   db 9, '$'

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_1
    or AL,30h
    mov [SI],AL

end_1:
    pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

WRD_TO_DEC PROC near
    push cx
    push dx
    mov cx, 10
wloop_bd:
    div cx
    or dl, 30h
    mov [si], dl
    dec si
    xor dx, dx
    cmp ax, 10
    jae wloop_bd
    cmp al, 00h
    je wend_1
    or al, 30h
    mov [si], al
wend_1:
    pop dx
    pop cx

```



```

        ret
WRD_TO_DEC endp

```

;ВЫВОД строки

```

PRINT proc near
    push    ax
    mov     ah, 09h
    int     21h
    pop     ax
    ret
PRINT endp

```

;print a symbol

```

Print_symb proc near
    push    ax
    mov     ah, 02h
    int     21h
    pop     ax
    ret
Print_symb endp

```

Begin:

```

    mov     ah, 4Ah      ;amount of available memory
    mov     bx, 0FFFFh   ; max amount
    int     21h

    mov     dx, 0
    mov     ax, bx
    mov     cx, 10h
    mul     cx

    lea     si, Available_memory
    add     si, 24
    call    wrd_to_dec

```

```

    lea    dx, Available_memory
    call   print
    lea    dx, Endl
    call   print

    mov    al, 30h          ; amount of extended memory
    out    70h, al
    in     al, 71h
    mov    bl, al
    mov    al, 31h
    out    70h, al
    in     al, 71h
    mov    ah, al
    mov    al, bl

    lea    si, Extended_memory
add si, 24
    mov    dx, 0
    call   wrd_to_dec

    lea    dx, Extended_memory
    call   print
    lea    dx, Endl
    call   print

    lea    dx, MCB          ; MCB blocks
    call   print
    lea    dx, Endl
    call   print

    mov    ah, 52h          ;getting access to MCB
    int    21h
    mov    ax, es:[bx-2]
    mov    es, ax

```

```

        ;type of MCB
tag1:
        mov     al, es:[0000h]
        call    BYTE_TO_HEX
        lea     di, MCBtype
        add di, 10
        mov     [di], ax

        lea     dx, MCBtype
        call    print
        lea     dx, Tab
        call    print

        mov     ax, es:[1h]      ; segment address PSP owner of piece of memory
        lea     di, PSP_address
        add     di, 15
        call    WRD_TO_HEX

        lea     dx, PSP_address
        call    print
        lea     dx, Tab
        call    print

        mov     ax, es:[0003h] ;
        mov     cx, 10h
        mul     cx

        lea     si, size_s
        add     si, 13
        call    WRD_TO_DEC
        lea     dx, Size_s
        call    Print
        lea     dx, Tab

```

```

call    Print

;последние 8 байт
push    ds
push    es
pop     ds

mov     dx, 8h
mov     di, dx
mov     cx, 8h
tag2:
        cmp     cx, 0
        je      tag3
mov     dl, byte PTR [di]
call    Print_symb
dec     cx
inc     di
jmp     tag2
tag3:
        pop     ds
        lea     dx, Endl
call    Print

cmp     byte ptr es:[0000h], 5Ah ; check is the last block or not
je      Quit

mov     ax, es      ; address of next block
add     ax, es:[3h]
inc     ax
mov     es, ax
jmp     tag1

Quit:
mov     ax, 0
mov     ah, 4ch ;finish

```

```

        int    21h
TESTPC      ENDS
            END    Start

```

lab3_2.asm

```

TESTPC      segment
ASSUME CS: TESTPC, DS: TESTPC, ES:nothing, SS:nothing
ORG    100h

```

```

Start:
jmp Begin

```

```

Available_memory      db 'Available memory:          b$'
Extended_memory db 'Extended memory:                Kb$'
MCB                   db 'List of MCB:$'
MCBtype db 'MCB type: 00h$'
PSP_address          db 'PSP adress: 0000h$'
size_s              db 'Size:                b$'
Endl db 13, 10, '$'
Tab      db 9, '$'

```

```

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_1
or AL,30h
mov [SI],AL

end_1:
pop DX
pop CX
ret
BYTE_TO_DEC ENDP

WRD_TO_DEC PROC near
push cx
push dx
mov cx, 10
wloop_bd:
div cx
or dl, 30h
mov [si], dl
dec si
xor dx, dx
cmp ax, 10
jae wloop_bd
cmp al, 00h
je wend_1
or al, 30h
mov [si], al
wend_1:
pop dx
pop cx

```

```
ret
WRD_TO_DEC endp
```

```
;ВЫВОД строки
PRINT proc near
push ax
mov ah, 09h
int 21h
pop ax
ret
PRINT endp
```

```
;print a symbol
Print_symb proc near
push ax
mov ah, 02h
int 21h
pop ax
ret
Print_symb endp
```

```
Begin:
mov ah, 4Ah ;amount of available memory
mov bx, 0FFFFh ; max amount
int 21h

mov dx, 0
mov ax, bx
mov cx, 10h
mul cx

lea si, Available_memory
add si, 24
call wrd_to_dec
```



```

lea    dx, Available_memory
call   print
lea    dx, Endl
call   print

; clear memory
lea    ax, ProgEnds
mov     bx, 10h
mov     dx, 0
div     bx
inc     ax
mov     bx, ax
mov     al, 0
mov     ah, 4Ah
int     21h

mov     al, 30h      ; amount of extended memory
out     70h, al
in      al, 71h
mov     bl, al
mov     al, 31h
out     70h, al
in      al, 71h
mov     ah, al
mov     al, bl

lea     si, Extended_memory
add     si, 24
mov     dx, 0
call    wrd_to_dec

lea     dx, Extended_memory
call    print
lea     dx, Endl

```

```

call print

lea      dx, MCB      ;   MCB blocks
call print
lea      dx, End1
call print

mov      ah, 52h      ;getting access to MCB
int      21h
mov      ax, es:[bx-2]
mov      es, ax

;type of MCB
tag1:
mov      al, es:[0000h]
call     BYTE_TO_HEX
lea      di, MCBtype
add      di, 10
mov      [di], ax

lea      dx, MCBtype
call     print
lea      dx, Tab
call     print

mov      ax, es:[1h]      ; segment address PSP owner of piece of memory
lea      di, PSP_address
add      di, 15
call     WRD_TO_HEX

lea      dx, PSP_address
call     print
lea      dx, Tab
call     print

```

```

mov  ax, es:[0003h] ;
mov  cx, 10h
mul  cx

```

```

lea  si, size_s
add  si, 13
call WRD_TO_DEC
lea  dx, Size_s
call Print
lea  dx, Tab
call Print

```

```

;последние 8 байт
push ds
push es
pop  ds

```

```

mov  dx, 8h
mov  di, dx
mov  cx, 8h
tag2:
cmp  cx, 0
je   tag3
mov  dl, byte PTR [di]
call Print_symb
dec  cx
inc  di
jmp  tag2
tag3:
pop  ds
lea  dx, Endl
call Print

```

```

cmp  byte ptr es:[0000h], 5Ah ; check is the last block or not

```

```

je          Quit

mov  ax, es      ; address of next block
add  ax, es:[3h]
inc  ax
mov  es, ax
jmp  tag1

```

```

Quit:
mov ax, 0
mov ah, 4ch ;finish
int  21h
ProgEnds:
TESTPC      ENDS
END  Start

```

lab3_3.asm

```

TESTPC      segment
                ASSUME CS: TESTPC, DS: TESTPC, ES:nothing, SS:nothing
                ORG 100h

```

```

Start:

```

```

    jmp Begin

```

```

    Available_memory      db 'Available memory:          b$'
    Extended_memory db 'Extended memory:                Kb$'
    MCB                   db 'List of MCB:$'
    MCBtype db 'MCB type: 00h$'
    PSP_address           db 'PSP adress: 0000h$'
    size_s                db 'Size:                      b$'
    Endl                  db 13, 10, '$'
    Tab                   db 9, '$'
    Fail db 'Memory request failed$'
    Success db 'Memory request succeeded$'

```

```

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

WRD_TO_DEC PROC near
    push    cx
    push    dx
    mov     cx, 10
wloop_bd:
    div     cx

```

```

        or     dl, 30h
        mov    [si], dl
        dec    si
        xor    dx, dx
        cmp    ax, 10
        jae    wloop_bd
        cmp    al, 00h
        je     wend_1
        or     al, 30h
        mov    [si], al
wend_1:
        pop    dx
        pop    cx
        ret
WRD_TO_DEC endp

```

```

PRINT proc near

```

```

        push   ax
        mov    ah, 09h
        int    21h
        pop    ax
        ret

```

```

PRINT endp

```

```

;print a symbol

```

```

Print_symb proc near

```

```

        push   ax
        mov     ah, 2h
        int     21h
        pop     ax
        ret

```

```

Print_symb endp

```

```

Begin:

```

```

    mov  ah, 4Ah      ;amount of available memory
    mov  bx, 0FFFFh   ; max amount
    int  21h

    mov  dx, 0
    mov  ax, bx
    mov  cx, 10h
    mul  cx

    lea  si,  Available_memory
add si, 24
    call wrd_to_dec

    lea  dx, Available_memory
    call print
    lea  dx, Endl
    call print

; clear memory
    lea  ax, ProgEnds
    mov  bx, 10h
    mov  dx, 0
    div  bx
    inc  ax
    mov  bx, ax
    mov  al, 0
    mov  ah, 4Ah
    int  21h

;request of memory
    mov  ax, 0
    mov  bx, 1000h ; 64kb
    mov  ah, 48h
    int  21h

```



```

        jnc      Success_Ex
    lea     dx, Fail
    call    Print
        lea     dx, Endl
    call    Print

```

Success_Ex:

```

    lea     dx, Success
    call    Print
    lea     dx, Endl
    call    Print

    mov     al, 30h      ; amount of extended memory
    out     70h, al
    in      al, 71h
    mov     bl, al
    mov     al, 31h
    out     70h, al
    in      al, 71h
    mov     ah, al
    mov     al, bl

    lea     si, Extended_memory
add si, 24
    mov     dx, 0
    call    wrd_to_dec

    lea     dx, Extended_memory
    call    print
    lea     dx, Endl
    call    print

    lea     dx, MCB      ; MCB blocks
    call    print
        lea     dx, Endl
    call    print

```

```

mov     ah, 52h    ;getting access to MCB
int     21h
mov     ax, es:[bx-2]
mov     es, ax

        ;type of MCB
tag1:
        mov     al, es:[0000h]
        call    BYTE_TO_HEX
        lea     di, MCBtype
        add di, 10
        mov     [di], ax

        lea     dx, MCBtype
        call    print
        lea     dx, Tab
        call    print

        mov     ax, es:[1h]    ; segment address PSP owner of piece of memory
        lea     di, PSP_address
        add     di, 15
        call    WRD_TO_HEX

        lea     dx, PSP_address
        call    print
        lea     dx, Tab
        call    print

        mov     ax, es:[0003h] ;
        mov     cx, 10h
        mul     cx

        lea     si, size_s

```

```

    add    si, 13
    call   WRD_TO_DEC
    lea     dx, Size_s
    call   Print
    lea     dx, Tab
call    Print

    push    ds
    push    es
    pop     ds

    mov     dx, 8h
    mov     di, dx
    mov     cx, 8h
tag2:
    cmp     cx, 0
    je      tag3
    mov     dl, byte PTR [di]
    call    Print_symb
    dec     cx
    inc     di
    jmp     tag2
tag3:
    pop     ds
    lea     dx, Endl
    call    Print

    cmp     byte ptr es:[0000h], 5Ah ; check is the last block or not
    je      Quit

    mov     ax, es ; address of next block
    add     ax, es:[3h]
    inc     ax
    mov     es, ax
    jmp     tag1

```

```

Quit:

    mov ax, 0

    mov ah, 4ch ;finish

    int 21h

ProgEnds:

TESTPC      ENDS

    END      Start

```

lab3_4.asm

```

TESTPC      segment
                ASSUME CS: TESTPC, DS: TESTPC, ES:nothing, SS:nothing
                ORG 100h

Start:
    jmp Begin

    Available_memory      db 'Available memory:          b$'
    Extended_memory db 'Extended memory:                Kb$'
    MCB                   db 'List of MCB:$'
    MCBtype db 'MCB type: 00h$'
    PSP_address           db 'PSP address: 0000h$'
    size_s                db 'Size:                      b$'
    Endl                  db 13, 10, '$'
    Tab                   db 9, '$'
    Fail db 'Memory request failed$'
    Success db 'Memory request succeeded$'

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

WRD_TO_DEC PROC near
    push cx
    push dx
    mov cx, 10
wloop_bd:
    div cx
    or dl, 30h
    mov [si], dl
    dec si
    xor dx, dx
    cmp ax, 10
    jae wloop_bd
    cmp al, 00h
    je wend_l
    or al, 30h
    mov [si], al

```

```

wend_l:
    pop    dx
    pop    cx
    ret
WRD_TO_DEC endp

PRINT proc near
    push    ax
    mov     ah, 09h
    int     21h
    pop     ax
    ret
PRINT endp

;print a symbol
Print_symb proc near
    push    ax
    mov     ah, 2h
    int     21h
    pop     ax
    ret
Print_symb endp

Begin:
    mov     ah, 4Ah      ;amount of available memory
    mov     bx, 0FFFFh   ; max amount
    int     21h

    mov     dx, 0
    mov     ax, bx
    mov     cx, 10h
    mul     cx

    lea     si, Available_memory
    add     si, 24
    call    wrd_to_dec

    lea     dx, Available_memory
    call    print
    lea     dx, Endl
    call    print

;request of memory
    mov     ax, 0
    mov     bx, 1000h ; 64kb
    mov     ah, 48h
    int     21h

    jnc     Success_Ex
    lea     dx, Fail
    call    Print
    lea     dx, Endl
    call    Print
    jmp     Clear

```

```

Success_Ex:
    lea     dx, Success
    call    Print
    lea     dx, Endl
    call    Print

    ; clear memory
Clear:
    lea ax, ProgEnds
    mov     bx, 10h
    mov     dx, 0
    div     bx
    inc     ax
    mov     bx, ax
    mov     al, 0
    mov     ah, 4Ah
    int     21h

    mov     al, 30h        ; amount of extended memory
    out     70h, al
    in      al, 71h
    mov     bl, al
    mov     al, 31h
    out     70h, al
    in      al, 71h
    mov     ah, al
    mov     al, bl

    lea     si, Extended_memory
    add     si, 24
    mov     dx, 0
    call    wrd_to_dec

    lea     dx, Extended_memory
    call    print
    lea     dx, Endl
    call    print

    lea     dx, MCB        ; MCB blocks
    call    print
    lea     dx, Endl
    call    print

    mov     ah, 52h        ;getting access to MCB
    int     21h
    mov     ax, es:[bx-2]
    mov     es, ax

    ;type of MCB
tag1:
    mov     al, es:[0000h]
    call    BYTE_TO_HEX
    lea     di, MCBtype
    add     di, 10
    mov     [di], ax

```

```

        lea     dx, MCBtype
        call    print
        lea     dx, Tab
        call    print

memory    mov     ax, es:[1h]      ; segment address PSP owner of piece of
        lea     di, PSP_address
        add     di, 15
        call    WRD_TO_HEX

        lea     dx, PSP_address
        call    print
        lea     dx, Tab
        call    print

        mov     ax, es:[0003h] ;
        mov     cx, 10h
        mul     cx

        lea     si, size_s
        add     si, 13
        call    WRD_TO_DEC
        lea     dx, Size_s
        call    Print
        lea     dx, Tab
        call    Print

        push    ds
        push    es
        pop     ds

        mov     dx, 8h
        mov     di, dx
        mov     cx, 8h
tag2:
        cmp     cx, 0
        je      tag3
        mov     dl, byte PTR [di]
        call    Print_symb
        dec     cx
        inc     di
        jmp     tag2
tag3:
        pop     ds
        lea     dx, Endl
        call    Print

not        cmp     byte ptr es:[0000h], 5Ah ; check is the last block or
        je      Quit

        mov     ax, es      ; address of next block
        add     ax, es:[3h]
        inc     ax
        mov     es, ax

```



```
        jmp     tag1

Quit:
        mov ax, 0
        mov ah, 4ch ;finish
        int  21h
ProgEnds:
TESTPC      ENDS
        END    Start
```