

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. 0382

Шангичев В. А.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2022

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.

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) 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 МСВ 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 4АН»). 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.

Выполнение работы.

Шаг 1. Для выполнения данного шага был написан исходный файл lab3_1.asm. В нем были реализованы следующие процедуры:

`print_available_memory` - печатает размер доступной памяти в байтах.

`print_extended_memory` - печатает размер расширенной памяти в байтах.

`print_memory_control_blocks` - печатает содержимое блоков контроля памяти.

`print_mcb` - печатает содержимое одного блока управления памятью.

`set_sc` - записывает в строку, выводимую процедурой `print_mcb`, значение поля `sc` блока контроля памяти.

`set_size` - записывает в строку, выводимую процедурой `print_mcb`, размер памяти в байтах, контролируемой MCB.

`set_pcp_owner` - записывает в строку, выводимую процедурой `print_mcb`, сегментный адрес PSP владельца участка памяти, контролируемой MCB.

`set_mcb_address` - записывает в строку, выводимую процедурой `print_mcb`, адрес блока контроля памяти.

`print` - печатает строку по смещению, указанному в регистре `dx`.

`convert_to_decimal` - конвертирует значение размера памяти, указанного в регистре `ax` в параграфах, в десятичное значение, записывая число по адресу `[si]` справа налево.

Также были использованы шаблонные процедуры, описанные в методических указаниях.

Вывод программы:

```
C:\DOS>exe2bin.exe lab3_1.exe lab3_1.com

C:\DOS>lab3_1.com
Available memory size:      648912
Extended memory size:      245760
Address: 016F PCP owner:0008 Size:      16 SC/SD:
Address: 0171 PCP owner:0000 Size:      64 SC/SD:
Address: 0176 PCP owner:0040 Size:     256 SC/SD:
Address: 0187 PCP owner:0192 Size:     144 SC/SD:
Address: 0191 PCP owner:0192 Size:    648912 SC/SD: LAB3_1

C:\DOS>_
```

Шаг 2. Для освобождения памяти была реализована процедура `free_memory`. Вывод измененной программы:

```

C:\DOS>lab3_2.com
Available memory size:      648912
Extended memory size:      245760
Address: 016F PCP owner:0008 Size:      16 SC/SD:
Address: 0171 PCP owner:0000 Size:      64 SC/SD:
Address: 0176 PCP owner:0040 Size:     256 SC/SD:
Address: 0187 PCP owner:0192 Size:     144 SC/SD:
Address: 0191 PCP owner:0192 Size:     768 SC/SD: LAB3_2
Address: 01C2 PCP owner:0000 Size:    648128 SC/SD:

C:\DOS>

```

Как можно видеть, программа теперь занимает только необходимую ей память.

Шаг 3. Для запроса памяти была написана процедура `request_memory`.

Вывод программы:

```

C:\DOS>lab3_3.com
Available memory size:      648912
Extended memory size:      245760
Address: 016F PCP owner:0008 Size:      16 SC/SD:
Address: 0171 PCP owner:0000 Size:      64 SC/SD:
Address: 0176 PCP owner:0040 Size:     256 SC/SD:
Address: 0187 PCP owner:0192 Size:     144 SC/SD:
Address: 0191 PCP owner:0192 Size:     848 SC/SD: LAB3_3
Address: 01C7 PCP owner:0192 Size:    65536 SC/SD: LAB3_3
Address: 11C8 PCP owner:0000 Size:    582496 SC/SD:

C:\DOS>

```

Теперь программе принадлежат два блока: тот, который остался после освобождения, и блок размером в 64Кб, который был запрошен позднее.

Шаг 4. Вывод программы:

```
C:\DOS>lab3_4.com
Available memory size:      648912
Extended memory size:      245760
Error: the size of requested memory is too large.
Address: 016F PCP owner:0008 Size:      16 SC/SD:
Address: 0171 PCP owner:0000 Size:      64 SC/SD:
Address: 0176 PCP owner:0040 Size:     256 SC/SD:
Address: 0187 PCP owner:0192 Size:     144 SC/SD:
Address: 0191 PCP owner:0192 Size:     848 SC/SD: LAB3_4
Address: 01C7 PCP owner:0000 Size:    648048 SC/SD:

C:\DOS>_
```

Как можно догадаться из предупредительного сообщения, выведенного процедурой `request_memory`, выделить 64Кб программе до освобождения памяти не удалось.

Выводы.

Для исследования организации управления памятью была написана программа, реализующая печать ознакомительных сообщений, относящихся к блокам управления памятью. В ходе выполнения лабораторной работы были изучены и применены на практике прерывания для работы с памятью.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл lab3_1.asm

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100h

start:

jmp begin

data:

```
available_memory_msg db "Available memory size:           ",
0dh, 0ah, '$'
extended_memory_msg db "Extended memory size:           ",
0dh, 0ah, '$'
mcb_msg db "Address:                                PCP  owner:                Size:
SC/SD:           ", 0dh, 0ah, '$'
```

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
    push ax
    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 ax
    pop DX
    pop CX
    ret
```

byte_to_dec ENDP

convert_to_decimal proc near

```
    ; ax - paragraph
    ; si - low digit of result
    push bx
    push dx
```

```
    mov bx, 16
    mul bx ; convert to num of bytes
```

```
    mov bx, 10
```

convert:

```
    div bx
    add dl, '0'
    mov [si], dl
    dec si
    xor dx, dx
    cmp ax, 0000h
    jne convert
```

```
    pop dx
    pop bx
    ret
```

convert_to_decimal endp

print proc near

```
    ; dx - offset of message
    push ax
    mov ah, 09h
    int 21h
    pop ax
```



```

        ret
print endp

print_available_memory proc near
    mov ah, 4Ah
    mov bx, 0ffffh
    int 21h ; now bx contains size of available memory

    mov si, offset available_memory_msg
    add si, 33

    mov ax, bx
    call convert_to_decimal

    mov dx, offset available_memory_msg
    call print
    ret
print_available_memory endp

print_extended_memory 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 si, offset extended_memory_msg
    add si, 33

    call convert_to_decimal

    mov dx, offset extended_memory_msg
    call print
    ret
print_extended_memory endp

set_mcb_address proc near
    ; ax - address
    push di
    mov di, offset mcb_msg
    add di, 12
    call wrd_to_hex
    pop di
    ret
set_mcb_address endp

set_pcp_owner proc near
    ; es - address of mcb
    push ax
    push di
    mov ax, es:[1]
    mov di, offset mcb_msg

```

```

        add di, 27
        call wrd_to_hex
        pop di
        pop ax
        ret
set_pcp_owner endp

set_size proc near
    ; es - address of mcb
    push si
    push ax

    mov si, offset mcb_msg
    add si, 45
    mov ax, es:[3]
    call convert_to_decimal

    pop ax
    pop si
    ret
set_size endp

set_sc proc near
    push di
    push ax
    push bx

    mov di, offset mcb_msg
    add di, 54
    mov si, 8
sc_write:
    mov bx, es:[si]
    mov [di], bx
    add si, 2
    add di, 2
    cmp si, 16
    jb sc_write

    pop bx
    pop ax
    pop di
    ret
set_sc endp

print_mcb proc near
    ; es - address of mcb
    push ax
    push dx

    mov ax, es
    call set_mcb_address
    call set_pcp_owner
    call set_size
    call set_sc
    mov dx, offset mcb_msg
    call print

    pop dx

```

```

        pop ax
        ret
print_mcb endp

print_memory_control_blocks proc near
    ; get address of first block
    mov ah, 52h
    int 21h
    mov es, es:[bx-2]

    print_msbs:
        call print_mcb
        mov ah, es:[0]
        cmp ah, 5Ah
        je end_
        mov ax, es
        add ax, es:[3]
        inc ax
        mov es, ax
        jmp print_msbs

    end_:

    ret
print_memory_control_blocks endp

begin:
    call print_available_memory
    call print_extended_memory
    call print_memory_control_blocks

    xor al, al
    mov ah, 4Ch
    int 21h

finish_program:

TESTPC ENDS
END start

```

Файл lab3_2.asm

```

TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100h

start:
    jmp begin

data:
    available_memory_msg db "Available memory size:      ",
0dh, 0ah, '$'
    extended_memory_msg db "Extended memory size:      ",
0dh, 0ah, '$'

```

SC/SD: mcb_msg db "Address:
 ", 0dh, 0ah, '\$'

PCP owner:

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
```

```
byte_to_dec PROC near
    push CX
    push DX
    push ax
    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 ax
        pop DX
        pop CX
        ret
byte_to_dec ENDP

```

```

convert_to_decimal proc near
    ; ax - paragraph
    ; si - low digit of result
    push bx
    push dx

    mov bx, 16
    mul bx ; convert to num of bytes

    mov bx, 10
convert:
    div bx
    add dl, '0'
    mov [si], dl
    dec si
    xor dx, dx
    cmp ax, 0000h
    jne convert

    pop dx
    pop bx
    ret
convert_to_decimal endp

```

```

print proc near
    ; dx - offset of message
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
print endp

```

```

print_available_memory proc near
    mov ah, 4Ah
    mov bx, 0ffffh
    int 21h ; now bx contains size of available memory

    mov si, offset available_memory_msg
    add si, 33

    mov ax, bx
    call convert_to_decimal

    mov dx, offset available_memory_msg
    call print

```

```

        ret
print_available_memory endp

print_extended_memory 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 si, offset extended_memory_msg
    add si, 33

    call convert_to_decimal

    mov dx, offset extended_memory_msg
    call print
    ret
print_extended_memory endp

set_mcb_address proc near
    ; ax - address
    push di
    mov di, offset mcb_msg
    add di, 12
    call wrd_to_hex
    pop di
    ret
set_mcb_address endp

set_pcp_owner proc near
    ; es - address of mcb
    push ax
    push di
    mov ax, es:[1]
    mov di, offset mcb_msg
    add di, 27
    call wrd_to_hex
    pop di
    pop ax
    ret
set_pcp_owner endp

set_size proc near
    ; es - address of mcb
    push si
    push ax

    mov si, offset mcb_msg
    add si, 45
    mov ax, es:[3]
    call convert_to_decimal

```

```

        pop ax
        pop si
        ret
set_size endp

set_sc proc near
    push di
    push ax
    push bx

    mov di, offset mcb_msg
    add di, 54
    mov si, 8
sc_write:
    mov bx, es:[si]
    mov [di], bx
    add si, 2
    add di, 2
    cmp si, 16
    jb sc_write

    pop bx
    pop ax
    pop di
    ret
set_sc endp

print_mcb proc near
    ; es - address of mcb
    push ax
    push dx

    mov ax, es
    call set_mcb_address
    call set_pcp_owner
    call set_size
    call set_sc
    mov dx, offset mcb_msg
    call print

    pop dx
    pop ax
    ret
print_mcb endp

print_memory_control_blocks proc near
    ; get address of first block
    mov ah, 52h
    int 21h
    mov es, es:[bx-2]

    print_msbs:
        call print_mcb
        mov ah, es:[0]
        cmp ah, 5Ah
        je end_
        mov ax, es
        add ax, es:[3]

```

```

        inc ax
        mov es, ax
        jmp print_msbs

    end_:

    ret
print_memory_control_blocks endp

free_memory proc near
    push ax
    push bx
    push dx

    lea ax, finish_program
    mov bx, 10h
    xor dx, dx
    div bx
    inc ax

    start_free:
        mov bx, ax
        xor ax, ax
        mov ah, 4ah
        int 21h

    pop dx
    pop bx
    pop ax
    ret
free_memory endp

begin:
    call print_available_memory
    call print_extended_memory
    call free_memory
    call print_memory_control_blocks

    xor al, al
    mov ah, 4Ch
    int 21h

finish_program:

TESTPC ENDS
END start

```

Файл lab3_3.asm

```

TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100h

start:
    jmp begin

```



```

data:
    available_memory_msg db "Available memory size:           ",
0dh, 0ah, '$'
    extended_memory_msg db "Extended memory size:           ",
0dh, 0ah, '$'
    mcb_msg db "Address:                PCP owner:                Size:
SC/SD:      ", 0dh, 0ah, '$'
    error_msg db "Error: the size of requested memory is too
large.", 0dh, 0ah, '$'

```

```

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
    push ax
    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 ax
        pop DX
        pop CX
        ret
byte_to_dec ENDP

```

```

convert_to_decimal proc near
    ; ax - paragraph
    ; si - low digit of result
    push bx
    push dx

    mov bx, 16
    mul bx ; convert to num of bytes

    mov bx, 10
convert:
    div bx
    add dl, '0'
    mov [si], dl
    dec si
    xor dx, dx
    cmp ax, 0000h
    jne convert

    pop dx
    pop bx
    ret
convert_to_decimal endp

```

```

print proc near
    ; dx - offset of message
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
print endp

```

```

print_available_memory proc near
    mov ah, 4Ah
    mov bx, 0ffffh
    int 21h ; now bx contains size of available memory

```

```

    mov si, offset available_memory_msg
    add si, 33

    mov ax, bx
    call convert_to_decimal

    mov dx, offset available_memory_msg
    call print
    ret
print_available_memory endp

print_extended_memory 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 si, offset extended_memory_msg
    add si, 33

    call convert_to_decimal

    mov dx, offset extended_memory_msg
    call print
    ret
print_extended_memory endp

set_mcb_address proc near
    ; ax - address
    push di
    mov di, offset mcb_msg
    add di, 12
    call wrd_to_hex
    pop di
    ret
set_mcb_address endp

set_pcp_owner proc near
    ; es - address of mcb
    push ax
    push di
    mov ax, es:[1]
    mov di, offset mcb_msg
    add di, 27
    call wrd_to_hex
    pop di
    pop ax
    ret
set_pcp_owner endp

set_size proc near
    ; es - address of mcb

```

```

    push si
    push ax

    mov si, offset mcb_msg
    add si, 45
    mov ax, es:[3]
    call convert_to_decimal

    pop ax
    pop si
    ret
set_size endp

set_sc proc near
    push di
    push ax
    push bx

    mov di, offset mcb_msg
    add di, 54
    mov si, 8
sc_write:
    mov bx, es:[si]
    mov [di], bx
    add si, 2
    add di, 2
    cmp si, 16
    jb sc_write

    pop bx
    pop ax
    pop di
    ret
set_sc endp

print_mcb proc near
    ; es - address of mcb
    push ax
    push dx

    mov ax, es
    call set_mcb_address
    call set_pcp_owner
    call set_size
    call set_sc
    mov dx, offset mcb_msg
    call print

    pop dx
    pop ax
    ret
print_mcb endp

print_memory_control_blocks proc near
    ; get address of first block
    mov ah, 52h
    int 21h
    mov es, es:[bx-2]

```

```

print_msbs:
    call print_mcb
    mov ah, es:[0]
    cmp ah, 5Ah
    je end_
    mov ax, es
    add ax, es:[3]
    inc ax
    mov es, ax
    jmp print_msbs

end_:

ret
print_memory_control_blocks endp

free_memory proc near
    push ax
    push bx
    push dx

    lea ax, finish_program
    mov bx, 10h
    xor dx, dx
    div bx
    inc ax

    start_free:
        mov bx, ax
        xor ax, ax
        mov ah, 4ah
        int 21h

    pop dx
    pop bx
    pop ax
    ret
free_memory endp

request_memory proc near
    ; bx - size of requested memory
    push ax

    mov ah, 48h
    int 21h
    jnc end_proc

    handle_error:
        push dx
        mov dx, offset error_msg
        call print
        pop dx

    end_proc:
        pop ax

```

```

        ret
request_memory endp

begin:
    call print_available_memory
    call print_extended_memory
    call free_memory
    mov bx, 1000h
    call request_memory
    call print_memory_control_blocks

    xor al, al
    mov ah, 4Ch
    int 21h

finish_program:

TESTPC ENDS
END start

```

Файл lab3_4.asm

```

TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100h

start:
    jmp begin

data:
    available_memory_msg db "Available memory size:           ",
0dh, 0ah, '$'
    extended_memory_msg db "Extended memory size:           ",
0dh, 0ah, '$'
    mcb_msg db "Address:                                PCP owner:                Size:
SC/SD:      ", 0dh, 0ah, '$'
    error_msg db "Error: the size of requested memory is too
large.", 0dh, 0ah, '$'

    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
    push ax
    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 ax
    pop DX
    pop CX
    ret
byte_to_dec ENDP

convert_to_decimal proc near
    ; ax - paragraph
    ; si - low digit of result
    push bx
    push dx

    mov bx, 16
    mul bx ; convert to num of bytes

```

```

    mov bx, 10
convert:
    div bx
    add dl, '0'
    mov [si], dl
    dec si
    xor dx, dx
    cmp ax, 0000h
    jne convert

    pop dx
    pop bx
    ret
convert_to_decimal endp

print proc near
    ; dx - offset of message
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
print endp

print_available_memory proc near
    mov ah, 4Ah
    mov bx, 0ffffh
    int 21h ; now bx contains size of available memory

    mov si, offset available_memory_msg
    add si, 33

    mov ax, bx
    call convert_to_decimal

    mov dx, offset available_memory_msg
    call print
    ret
print_available_memory endp

print_extended_memory 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 si, offset extended_memory_msg
    add si, 33

    call convert_to_decimal

```



```

        mov dx, offset extended_memory_msg
        call print
        ret
print_extended_memory endp

set_mcb_address proc near
    ; ax - address
    push di
    mov di, offset mcb_msg
    add di, 12
    call wrd_to_hex
    pop di
    ret
set_mcb_address endp

set_pcp_owner proc near
    ; es - address of mcb
    push ax
    push di
    mov ax, es:[1]
    mov di, offset mcb_msg
    add di, 27
    call wrd_to_hex
    pop di
    pop ax
    ret
set_pcp_owner endp

set_size proc near
    ; es - address of mcb
    push si
    push ax

    mov si, offset mcb_msg
    add si, 45
    mov ax, es:[3]
    call convert_to_decimal

    pop ax
    pop si
    ret
set_size endp

set_sc proc near
    push di
    push ax
    push bx

    mov di, offset mcb_msg
    add di, 54
    mov si, 8
    sc_write:
        mov bx, es:[si]
        mov [di], bx
        add si, 2
        add di, 2
        cmp si, 16

```

```

        jb sc_write

    pop bx
    pop ax
    pop di
    ret
set_sc endp

print_mcb proc near
    ; es - address of mcb
    push ax
    push dx

    mov ax, es
    call set_mcb_address
    call set_pcp_owner
    call set_size
    call set_sc
    mov dx, offset mcb_msg
    call print

    pop dx
    pop ax
    ret
print_mcb endp

print_memory_control_blocks proc near
    ; get address of first block
    mov ah, 52h
    int 21h
    mov es, es:[bx-2]

    print_msbs:
        call print_mcb
        mov ah, es:[0]
        cmp ah, 5Ah
        je end_
        mov ax, es
        add ax, es:[3]
        inc ax
        mov es, ax
        jmp print_msbs

    end_:

    ret
print_memory_control_blocks endp

free_memory proc near
    push ax
    push bx
    push dx

    lea ax, finish_program
    mov bx, 10h
    xor dx, dx
    div bx

```

```

    inc ax

    start_free:
        mov bx, ax
        xor ax, ax
        mov ah, 4ah
        int 21h

    pop dx
    pop bx
    pop ax
    ret
free_memory endp

request_memory proc near
    ; bx - size of requested memory
    push ax

    mov ah, 48h
    int 21h
    jnc end_proc

    handle_error:
        push dx
        mov dx, offset error_msg
        call print
        pop dx

    end_proc:
        pop ax

    ret
request_memory endp

begin:
    call print_available_memory
    call print_extended_memory
    mov bx, 1000h
    call request_memory
    call free_memory
    call print_memory_control_blocks

    xor al, al
    mov ah, 4Ch
    int 21h

finish_program:

TESTPC ENDS
END start

```