

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

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Операционные системы»
Тема: Исследование интерфейсов программных модулей

Студент гр. 0382

Охотникова Г.С.

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 MSB 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.

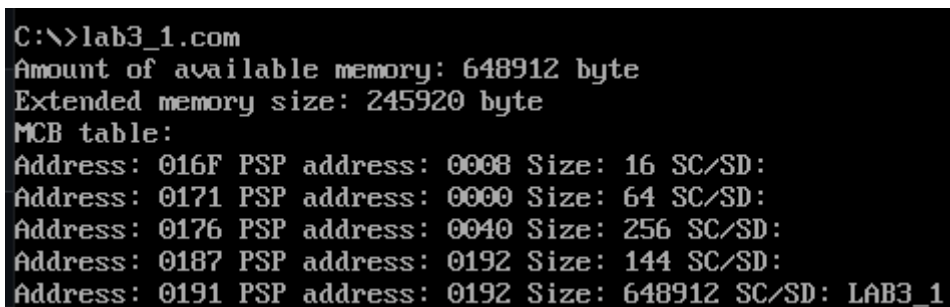
Шаг 4. Измените первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти. Обязательно обрабатывайте завершение функций ядра, проверяя флаг CF. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

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

Выполнение работы.

1. Был написан модуль типа .COM, шаблон которого взят из методических указаний. 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 виде десятичных чисел. Последние 8 байт MCB выводятся как символы.



```
C:\>lab3_1.com
Amount of available memory: 648912 byte
Extended memory size: 245920 byte
MCB table:
Address: 016F PSP address: 0008 Size: 16 SC/SD:
Address: 0171 PSP address: 0000 Size: 64 SC/SD:
Address: 0176 PSP address: 0040 Size: 256 SC/SD:
Address: 0187 PSP address: 0192 Size: 144 SC/SD:
Address: 0191 PSP address: 0192 Size: 648912 SC/SD: LAB3_1
```

Рисунок 1 — Результат запуска модуля LAB3_1.COM

2. Программа была изменена таким образом, чтобы она освобождала память, которую она не занимает.

```

C:\>lab3_2.com
Amount of available memory: 648912 byte
Extended memory size: 245920 byte
MCB table:
Address: 016F PSP address: 0008 Size: 16 SC/SD:
Address: 0171 PSP address: 0000 Size: 64 SC/SD:
Address: 0176 PSP address: 0040 Size: 256 SC/SD:
Address: 0187 PSP address: 0192 Size: 144 SC/SD:
Address: 0191 PSP address: 0192 Size: 6432 SC/SD: LAB3_2
Address: 0324 PSP address: 0000 Size: 642464 SC/SD: .i6p
.Ä

```

Рисунок 2 — Результат запуска модуля LAB3_2.COM

Теперь программа занимает ту область памяти, которая необходима для того, чтобы хранить ее.

3. Чтобы после освобождения памяти программа запрашивала 64Кб памяти функцией 48h прерывания 21h, была написана процедура MEMORY_REQUEST, которая вызывается после освобождения памяти.

```

C:\>lab3_3.com
Amount of available memory: 648912 byte
Extended memory size: 245920 byte
Memory request succeeded
MCB table:
Address: 016F PSP address: 0008 Size: 16 SC/SD:
Address: 0171 PSP address: 0000 Size: 64 SC/SD:
Address: 0176 PSP address: 0040 Size: 256 SC/SD:
Address: 0187 PSP address: 0192 Size: 144 SC/SD:
Address: 0191 PSP address: 0192 Size: 6432 SC/SD: LAB3_3
Address: 0324 PSP address: 0192 Size: 65536 SC/SD: LAB3_3
Address: 1325 PSP address: 0000 Size: 576912 SC/SD:

```

Рисунок 3 — Результат запуска модуля LAB3_3.COM.

4. Для того, чтобы программа запрашивала 64Кб памяти функцией 48h прерывания 21h до освобождения памяти, программа была изменена. Также сообщается о том, что запрос памяти провален, так как осуществляется попытка запросить память до освобождения, а до освобождения программа занимает всю доступную память.

```
C:\>lab3_4.com
Memory request failed
Amount of available memory: 648912 byte
Extended memory size: 245920 byte
MCB table:
Address: 016F PSP address: 0008 Size: 16 SC/SD:
Address: 0171 PSP address: 0000 Size: 64 SC/SD:
Address: 0176 PSP address: 0040 Size: 256 SC/SD:
Address: 0187 PSP address: 0192 Size: 144 SC/SD:
Address: 0191 PSP address: 0192 Size: 6432 SC/SD: LAB3_4
Address: 0324 PSP address: 0000 Size: 642464 SC/SD: .i6p
.ä
```

Рисунок 4 — Результат запуска модуля LAB3_4.COM.

Исходный программный код см. в приложении А.

Контрольные вопросы.

1. Что означает «доступный объём памяти»?

Доступный объём памяти — это количество оперативной памяти, которое открыто для использования программой во время выполнения.

2. Где MCB блок Вашей программы в списке?

На скриншотах с примерами работы программы MCB блок подписан названием исполняемого файла в столбце SC/SD.

3. Какой размер памяти занимает программа в каждом случае?

В первом случае программа занимает весь доступный объём памяти. Во втором — только необходимый объём памяти, то есть 6432 байт. В третьем — необходимый объём памяти и запрошенные 64Кб памяти ($6432 + 65536 = 71968$). В четвертом — только необходимый объём памяти (6432 байт).

Выводы.

При выполнении данной лабораторной работы были исследованы структуры данных и работа функций управления памятью ядра операционной системы.

ПРИЛОЖЕНИЕ А.

Название файла: lab3_1.asm

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

AVAILABLE_MEMORY DB 'Amount of available memory: ', '$'
EXTENDED_MEMORY DB 'Extended memory size: ', '$'
STRING_BYTE DB ' byte ', '$'
MCB_TABLE DB 'MCB table: ', 0DH, 0AH, '$'
ADDRESS DB 'Address:      ', '$'
PSP_ADDRESS DB 'PSP address:      ', '$'
STRING_SIZE DB 'Size: ', '$'
SC_SD DB 'SC/SD: ', '$'
NEW_STRING DB 0DH, 0AH, '$'
SPACE_STRING DB ' ', '$'
```

; Процедуры

;-----

```
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
; Байт в AL переводится в два символа шест. числа в AX
        push CX
        mov AH, AL
        call TETR_TO_HEX
```

```

    xchg AL,AH
    mov CL,4
    shr AL,CL
    call TETR_TO_HEX ; В AL старшая цифра
    pop CX ; В AH младшая цифра
    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 10 с/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_1
        or AL,30h
        mov [SI],AL

end_1:
        pop DX
        pop CX
        ret
BYTE_TO_DEC ENDP

PRINT_STRING PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_STRING endp

PARAGRAPH_TO_BYTE PROC
        mov bx, 0ah
        xor cx, cx

division_loop:
        div bx
        push dx
        inc cx
        sub dx, dx
        cmp ax, 0h
        jne division_loop

print:
        pop dx

```



```

        add dl,30h
        mov ah,02h
        int 21h

        loop print
        ret
PARAGRAPH_TO_BYTE endp

MEMORY_AVAILABLE PROC near
        mov dx, offset AVAILABLE_MEMORY
        call PRINT_STRING
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h
        mov ax, bx
        mov bx, 16
        mul bx
        call PARAGRAPH_TO_BYTE

        mov dx, offset STRING_BYTE
        call PRINT_STRING

        mov dx, offset NEW_STRING
        call PRINT_STRING
        ret
MEMORY_AVAILABLE endp

MEMORY_EXTENDED proc near
        mov al, 30h
        out 70h, al
        in al, 71h
        mov al, 31h
        out 70h, al
        in al, 71h
        mov ah, al
        mov bh, al
        mov ax, bx

```

```

    mov dx, offset EXTENDED_MEMORY
    call PRINT_STRING

    mov bx, 010h
    mul bx

    call PARAGRAPH_TO_BYTE

    mov dx, offset STRING_BYTE
    call PRINT_STRING

    mov dx, offset NEW_STRING
    call PRINT_STRING
    ret
MEMORY_EXTENDED endp

MCB PROC near
    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    mov es, ax
    mov dx, offset MCB_TABLE
    call PRINT_STRING

MCB_loop:
    mov ax, es
    mov di, offset ADDRESS
    add di, 12
    call WRD_TO_HEX
    mov dx, offset ADDRESS
    call PRINT_STRING
    mov dx, offset SPACE_STRING
    call PRINT_STRING

    mov ax, es:[1]    ;PSP
    mov di, offset PSP_ADDRESS
    add di, 16
    call WRD_TO_HEX
    mov dx, offset PSP_ADDRESS

```

```

call PRINT_STRING

mov dx, offset STRING_SIZE
call PRINT_STRING
mov ax, es:[3]
mov di, offset STRING_SIZE
add di, 6
mov bx, 16
mul bx
call PARAGRAPH_TO_BYTE
mov dx, offset SPACE_STRING
call PRINT_STRING

mov bx, 8          ;SC/SD
mov dx, offset SC_SD
call PRINT_STRING
mov cx, 7

SC_SD_loop:
mov dl, es:[bx]
mov ah, 02h
int 21h
inc bx
loop SC_SD_loop

mov dx, offset NEW_STRING
call PRINT_STRING

mov bx, es:[3h]
mov al, es:[0h]
cmp al, 5ah
je MCB_END

mov ax, es
inc ax
add ax, bx
mov es, ax
jmp MCB_loop

```

MCB_END:

ret

MCB endp

BEGIN:

call MEMORY_AVAILABLE

call MEMORY_EXTENDED

call MCB

xor al, al

mov ah, 4ch

int 21h

TESTPC ENDS

END START

Название файла: lab3_2.asm

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

AVAILABLE_MEMORY DB 'Amount of available memory: ', '\$'

EXTENDED_MEMORY DB 'Extended memory size: ', '\$'

STRING_BYTE DB ' byte ', '\$'

MCB_TABLE DB 'MCB table: ', 0DH, 0AH, '\$'

ADDRESS DB 'Address: ', '\$'

PSP_ADDRESS DB 'PSP address: ', '\$'

STRING_SIZE DB 'Size: ', '\$'

SC_SD DB 'SC/SD: ', '\$'

NEW_STRING DB 0DH, 0AH, '\$'

SPACE_STRING DB ' ', '\$'

; Процедуры

;-----

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
; Байт в AL переводится в два символа шест. числа в AX
        push CX
        mov AH,AL
        call TETR_TO_HEX
        xchg AL,AH
        mov CL,4
        shr AL,CL
        call TETR_TO_HEX ; В AL старшая цифра
        pop CX ; В AH младшая цифра
        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 10 с/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_1
    or AL,30h
    mov [SI],AL

end_1:
    pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

PRINT_STRING PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT_STRING endp

PARAGRAPH_TO_BYTE PROC
    mov bx, 0ah

```

```

        xor cx, cx

division_loop:
        div bx
        push dx
        inc cx
        sub dx, dx
        cmp ax, 0h
        jne division_loop

print:
        pop dx
        add dl, 30h
        mov ah, 02h
        int 21h

        loop print
        ret
PARAGRAPH_TO_BYTE endp

MEMORY_AVAILABLE PROC near
        mov dx, offset AVAILABLE_MEMORY
        call PRINT_STRING
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h
        mov ax, bx
        mov bx, 16
        mul bx
        call PARAGRAPH_TO_BYTE

        mov dx, offset STRING_BYTE
        call PRINT_STRING

        mov dx, offset NEW_STRING
        call PRINT_STRING
        ret
MEMORY_AVAILABLE endp

```

```

MEMORY_EXTENDED proc near
    mov al, 30h
    out 70h, al
    in al, 71h
    mov al, 31h
    out 70h, al
    in al, 71h
    mov ah, al
    mov bh, al
    mov ax, bx

    mov dx, offset EXTENDED_MEMORY
    call PRINT_STRING

    mov bx, 010h
    mul bx

    call PARAGRAPH_TO_BYTE

    mov dx, offset STRING_BYTE
    call PRINT_STRING

    mov dx, offset NEW_STRING
    call PRINT_STRING
    ret
MEMORY_EXTENDED endp

```

```

MCB PROC near
    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    mov es, ax
    mov dx, offset MCB_TABLE
    call PRINT_STRING

```

```

MCB_loop:
    mov ax, es
    mov di, offset ADDRESS

```



```

    add di, 12
    call WRD_TO_HEX
    mov dx, offset ADDRESS
    call PRINT_STRING
    mov dx, offset SPACE_STRING
    call PRINT_STRING

    mov ax, es:[1]    ;PSP
    mov di, offset PSP_ADDRESS
    add di, 16
    call WRD_TO_HEX
    mov dx, offset PSP_ADDRESS
    call PRINT_STRING

    mov dx, offset STRING_SIZE
    call PRINT_STRING
    mov ax, es:[3]
    mov di, offset STRING_SIZE
    add di, 6
    mov bx, 16
    mul bx
    call PARAGRAPH_TO_BYTE
    mov dx, offset SPACE_STRING
    call PRINT_STRING

    mov bx, 8          ;SC/SD
    mov dx, offset SC_SD
    call PRINT_STRING
    mov cx, 7

SC_SD_loop:
    mov dl, es:[bx]
    mov ah, 02h
    int 21h
    inc bx
    loop SC_SD_loop

    mov dx, offset NEW_STRING
    call PRINT_STRING

```

```

        mov bx, es:[3h]
        mov al, es:[0h]
        cmp al, 5ah
        je MCB_END

        mov ax, es
        inc ax
        add ax, bx
        mov es, ax
        jmp MCB_loop

MCB_END:
        ret
MCB endp

FREE_MEMORY PROC near
        mov     ax, cs
        mov     es, ax
        mov     bx, offset TESTPC_END
        mov     ax, es
        mov     bx, ax
        mov     ah, 4ah
        int     21h
        ret
FREE_MEMORY endp

BEGIN:
        call MEMORY_AVAILABLE
        call MEMORY_EXTENDED
        call FREE_MEMORY
        call MCB
        xor al, al
        mov ah, 4ch
        int 21h
TESTPC_END:
TESTPC ENDS
END START

```

Название файла: lab3_3.asm

```
TESTPC    SEGMENT

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

    ORG 100H

START:    JMP BEGIN


AVAILABLE_MEMORY DB 'Amount of available memory: ', '$'
EXTENDED_MEMORY DB 'Extended memory size: ', '$'
STRING_BYTE DB ' byte ', '$'
MCB_TABLE DB 'MCB table: ', 0DH, 0AH, '$'
ADDRESS DB 'Address:      ', '$'
PSP_ADDRESS DB 'PSP address:      ', '$'
STRING_SIZE DB 'Size: ', '$'
SC_SD DB 'SC/SD: ', '$'
NEW_STRING DB 0DH,0AH,'$'
SPACE_STRING DB ' ', '$'
MEMORY_REQUEST_FAIL DB 'Memory request failed', 0DH, 0AH, '$'
MEMORY_REQUEST_SUCCESS DB 'Memory request succeeded', 0DH, 0AH, '$'


; Процедуры
;-----
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
; Байт в AL переводится в два символа шест. числа в AX
    push CX
    mov AH,AL
    call TETR_TO_HEX
    xchg AL,AH
```

```

        mov CL,4
        shr AL,CL
        call TETR_TO_HEX ; В AL старшая цифра
        pop CX ; В AH младшая цифра
        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 10 с/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_1
        or AL,30h
        mov [SI],AL

end_1:
        pop DX
        pop CX
        ret
BYTE_TO_DEC ENDP

PRINT_STRING PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_STRING endp

PARAGRAPH_TO_BYTE PROC
        mov bx, 0ah
        xor cx, cx

division_loop:
        div bx
        push dx
        inc cx
        sub dx, dx
        cmp ax, 0h
        jne division_loop

print:
        pop dx
        add dl,30h

```

```

    mov ah, 02h
    int 21h

    loop print
    ret
PARAGRAPH_TO_BYTE endp

MEMORY_AVAILABLE PROC near
    mov dx, offset AVAILABLE_MEMORY
    call PRINT_STRING
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov bx, 16
    mul bx
    call PARAGRAPH_TO_BYTE

    mov dx, offset STRING_BYTE
    call PRINT_STRING

    mov dx, offset NEW_STRING
    call PRINT_STRING
    ret
MEMORY_AVAILABLE endp

MEMORY_EXTENDED proc near
    mov al, 30h
    out 70h, al
    in al, 71h
    mov al, 31h
    out 70h, al
    in al, 71h
    mov ah, al
    mov bh, al
    mov ax, bx

    mov dx, offset EXTENDED_MEMORY

```

```

    call PRINT_STRING

    mov bx, 010h
    mul bx

    call PARAGRAPH_TO_BYTE

    mov dx, offset STRING_BYTE
    call PRINT_STRING

    mov dx, offset NEW_STRING
    call PRINT_STRING
    ret
MEMORY_EXTENDED endp

MCB PROC near
    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    mov es, ax
    mov dx, offset MCB_TABLE
    call PRINT_STRING

MCB_loop:
    mov ax, es
    mov di, offset ADDRESS
    add di, 12
    call WRD_TO_HEX
    mov dx, offset ADDRESS
    call PRINT_STRING
    mov dx, offset SPACE_STRING
    call PRINT_STRING

    mov ax, es:[1]    ;PSP
    mov di, offset PSP_ADDRESS
    add di, 16
    call WRD_TO_HEX
    mov dx, offset PSP_ADDRESS
    call PRINT_STRING

```

```

    mov dx, offset STRING_SIZE
    call PRINT_STRING
    mov ax, es:[3]
    mov di, offset STRING_SIZE
    add di, 6
    mov bx, 16
    mul bx
    call PARAGRAPH_TO_BYTE
    mov dx, offset SPACE_STRING
    call PRINT_STRING

    mov bx, 8          ;SC/SD
    mov dx, offset SC_SD
    call PRINT_STRING
    mov cx, 7

SC_SD_loop:
    mov dl, es:[bx]
    mov ah, 02h
    int 21h
    inc bx
    loop SC_SD_loop

    mov dx, offset NEW_STRING
    call PRINT_STRING

    mov bx, es:[3h]
    mov al, es:[0h]
    cmp al, 5ah
    je MCB_END

    mov ax, es
    inc ax
    add ax, bx
    mov es, ax
    jmp MCB_loop

MCB_END:

```



```

        ret
MCB endp

FREE_MEMORY PROC near
    mov     ax, cs
    mov     es, ax
    mov     bx, offset TESTPC_END
    mov     ax, es
    mov     bx, ax
    mov     ah, 4ah
    int     21h
    ret
FREE_MEMORY endp

MEMORY_REQUEST PROC near
    mov     bx, 1000h ;64KB
    mov     ah, 48h
    int     21h
    jnb     memory_fail
    jmp     memory_success

memory_fail:
    mov     dx, offset MEMORY_REQUEST_FAIL
    call    PRINT_STRING
    jmp     memory_request_end

memory_success:
    mov     dx, offset MEMORY_REQUEST_SUCCESS
    call    PRINT_STRING

memory_request_end:
    ret
MEMORY_REQUEST endp

BEGIN:
    call    MEMORY_AVAILABLE
    call    MEMORY_EXTENDED
    call    FREE_MEMORY

```

```

        call MEMORY_REQUEST
        call MCB
        xor al, al
        mov ah, 4ch
        int 21h
TESTPC_END:
TESTPC ENDS
END START

```

Название файла: lab3_4.asm

```

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

AVAILABLE_MEMORY DB 'Amount of available memory: ', '$'
EXTENDED_MEMORY DB 'Extended memory size: ', '$'
STRING_BYTE DB ' byte ', '$'
MCB_TABLE DB 'MCB table: ', 0DH, 0AH, '$'
ADDRESS DB 'Address:      ', '$'
PSP_ADDRESS DB 'PSP address:      ', '$'
STRING_SIZE DB 'Size: ', '$'
SC_SD DB 'SC/SD: ', '$'
NEW_STRING DB 0DH, 0AH, '$'
SPACE_STRING DB ' ', '$'
MEMORY_REQUEST_FAIL DB 'Memory request failed', 0DH, 0AH, '$'
MEMORY_REQUEST_SUCCESS DB 'Memory request succeeded', 0DH, 0AH, '$'

; Процедуры
;-----
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
; Байт в AL переводится в два символа шест. числа в AX
    push CX
    mov AH,AL
    call TETR_TO_HEX
    xchg AL,AH
    mov CL,4
    shr AL,CL
    call TETR_TO_HEX ; В AL старшая цифра
    pop CX ; В AH младшая цифра
    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 10 с/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_1
        or AL,30h
        mov [SI],AL

end_1:
        pop DX
        pop CX
        ret
BYTE_TO_DEC ENDP

PRINT_STRING PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_STRING endp

PARAGRAPH_TO_BYTE PROC
        mov bx, 0ah
        xor cx, cx

division_loop:
        div bx
        push dx

```

```

        inc cx
        sub dx, dx
        cmp ax, 0h
        jne division_loop

print:
        pop dx
        add dl, 30h
        mov ah, 02h
        int 21h

        loop print
        ret
PARAGRAPH_TO_BYTE endp

MEMORY_AVAILABLE PROC near
        mov dx, offset AVAILABLE_MEMORY
        call PRINT_STRING
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h
        mov ax, bx
        mov bx, 16
        mul bx
        call PARAGRAPH_TO_BYTE

        mov dx, offset STRING_BYTE
        call PRINT_STRING

        mov dx, offset NEW_STRING
        call PRINT_STRING
        ret
MEMORY_AVAILABLE endp

MEMORY_EXTENDED proc near
        mov al, 30h
        out 70h, al
        in al, 71h

```

```

    mov al, 31h
    out 70h, al
    in al, 71h
    mov ah, al
    mov bh, al
    mov ax, bx

    mov dx, offset EXTENDED_MEMORY
    call PRINT_STRING

    mov bx, 010h
    mul bx

    call PARAGRAPH_TO_BYTE

    mov dx, offset STRING_BYTE
    call PRINT_STRING

    mov dx, offset NEW_STRING
    call PRINT_STRING
    ret
MEMORY_EXTENDED endp

MCB PROC near
    mov ah, 52h
    int 21h
    mov ax, es:[bx-2]
    mov es, ax
    mov dx, offset MCB_TABLE
    call PRINT_STRING

MCB_loop:
    mov ax, es
    mov di, offset ADDRESS
    add di, 12
    call WRD_TO_HEX
    mov dx, offset ADDRESS
    call PRINT_STRING
    mov dx, offset SPACE_STRING

```

```

call PRINT_STRING

mov ax, es:[1]    ;PSP
mov di, offset PSP_ADDRESS
add di, 16
call WRD_TO_HEX
mov dx, offset PSP_ADDRESS
call PRINT_STRING

mov dx, offset STRING_SIZE
call PRINT_STRING
mov ax, es:[3]
mov di, offset STRING_SIZE
add di, 6
mov bx, 16
mul bx
call PARAGRAPH_TO_BYTE
mov dx, offset SPACE_STRING
call PRINT_STRING

mov bx, 8          ;SC/SD
mov dx, offset SC_SD
call PRINT_STRING
mov cx, 7

SC_SD_loop:
mov dl, es:[bx]
mov ah, 02h
int 21h
inc bx
loop SC_SD_loop

mov dx, offset NEW_STRING
call PRINT_STRING

mov bx, es:[3h]
mov al, es:[0h]
cmp al, 5ah
je MCB_END

```

```

        mov ax, es
        inc ax
        add ax, bx
        mov es, ax
        jmp MCB_loop

MCB_END:
        ret
MCB endp

FREE_MEMORY PROC near
        mov     ax, cs
        mov     es, ax
        mov     bx, offset TESTPC_END
        mov     ax, es
        mov     bx, ax
        mov     ah, 4ah
        int     21h
        ret
FREE_MEMORY endp

MEMORY_REQUEST PROC near
        mov     bx, 1000h ;64KB
        mov     ah, 48h
        int     21h

        jnb     memory_fail
        jmp     memory_success

memory_fail:
        mov     dx, offset MEMORY_REQUEST_FAIL
        call    PRINT_STRING
        jmp     memory_request_end

memory_success:
        mov     dx, offset MEMORY_REQUEST_SUCCESS
        call    PRINT_STRING

```



```
memory_request_end:
    ret
MEMORY_REQUEST endp

BEGIN:
    call MEMORY_REQUEST
    call MEMORY_AVAILABLE
    call MEMORY_EXTENDED
    call FREE_MEMORY
    call MCB

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

TESTPC_END:
TESTPC ENDS
END START
```