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

Студентка гр. 9383 _____ Лихашва А.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Постановка задачи

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.

Сведения о функциях и структурах данных.

В данной программе используются следующие функции и структуры данных:

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа, который записывается в AL
BYTE_TO_HEX	Перевод значений байта в число 16-ой СС и его представление в виде двух символов
WRD_TO_HEX	Перевод слова в число 16-ой СС и представление его в виде четырех символов
BYTE_TO_DEC	Перевод значения байта в число 10-ой СС и представляет его в виду символов
PRINT_STRING	Вывод строки на экран
PARAGRAPH_TO_BYTE	Перевод и запись числа 16-ой СС из ax в 10-ую СС по адресу, записанному в di
MEMORY_AVAILABLE	Запись количества доступной памяти
MEMORY_EXTENDED	Запись количества расширенной памяти

MCB	Печать цепочки блоков управления памятью
UNUSED_MEMORY_FREE	Освобождение неиспользуемой программой памяти
MEMORY_REQUEST	Запрос 64 Кб (1000h) памяти. Печать результат работы на экран.

Выполнение шагов лабораторной работы:

1 шаг:

Был написан модуль типа .COM, который выбирает и распечатывает следующую информацию: количество доступной памяти, размер расширенной памяти и выводит цепочку блоков управления памятью.

Адреса при выводе представляются в шестнадцатеричными числами. Объем памяти функциями управления памятью выводится в виде десятичных чисел. Последние 8 байт MCB выводятся как символы.

Результаты, полученные программой:

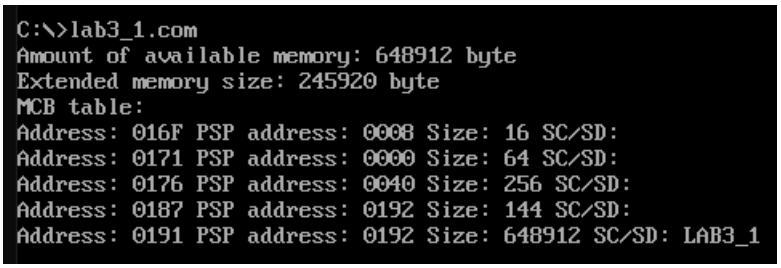

Рисунок 1: Пример работы программы №1

2 шаг:

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

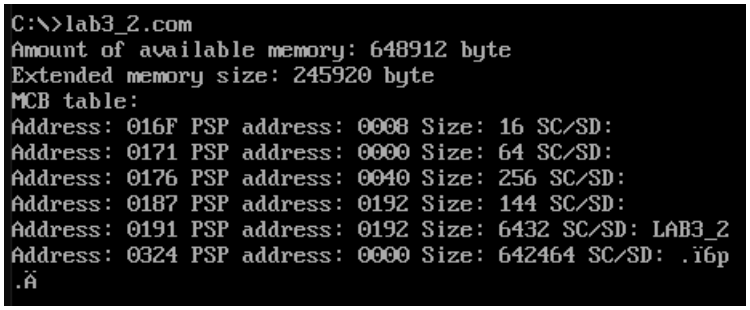

Рисунок 2: Пример работы программы №2

3 шаг:

Программа была изменена таким образом, чтобы после освобождения памяти программа запрашивала 64Кб памяти функцией 48h прерывания 21h

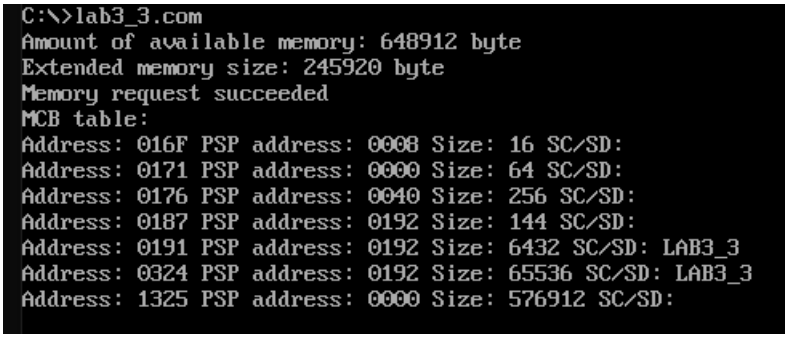

Рисунок 3: Пример работы программы №3

4 шаг:

Первоначальная программа была изменена таким образом, чтобы она запрашивала 64Кб памяти функцией 48h прерывания 21h до освобождения памяти.

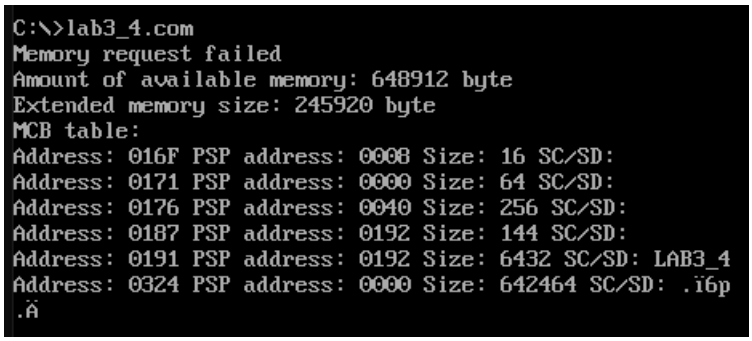

Рисунок 4: Пример работы программы №4

Ответы на контрольные вопросы

1. Что означает «доступный объём памяти»?

Доступный объём памяти — это область оперативной памяти, которая открыта для использования программой .

2. Где MCB блок Вашей программы в списке?

На скриншотах с примерами работы моей 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 случае программа занимает только необходимый объём памяти, то есть 6432 байт.

В третьем случае программа занимает необходимый объём памяти и запрошенные 64Кб памяти, то есть 6432 + 65536 = 71968.

В четвертом случае программа занимает только необходимый объём памяти, то есть 6432 байт, потому что была выделена память в 64Кб, и после этого сразу же освободили неиспользуемую память.

Заключение.

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

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

; Данные
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_l
     or AL,30h
     mov [SI],AL
end_l:  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              ;(dx ax) = ax*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                  ;address
    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 address
    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  ;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
```

Файл: lab3_2.asm

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

; Данные
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_l
     or AL,30h
     mov [SI],AL
end_l:  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              ;(dx ax) = ax*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                ;address
     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 address
     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  ;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
```

```
UNUSED_MEMORY_FREE 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
UNUSED_MEMORY_FREE endp


BEGIN:
     call MEMORY_AVAILABLE
     call MEMORY_EXTENDED
     call UNUSED_MEMORY_FREE
     call MCB

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

TESTPC_END:
TESTPC ENDS

END START
```

Файл: lab3_3.asm

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

; Данные
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_l
     or AL,30h
     mov [SI],AL
end_l:  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              ;(dx ax) = ax*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                  ;address
    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 address
    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  ;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


UNUSED_MEMORY_FREE 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
UNUSED_MEMORY_FREE endp


MEMORY_REQUEST PROC near
```

```
    mov     bx, 1000h ;64kb
    mov     ah, 48h
    int     21h

    jb      memory_fail ;cf = 1
    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 UNUSED_MEMORY_FREE
     call MEMORY_REQUEST
     call MCB

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

TESTPC_END:
TESTPC ENDS
```

```
END START
```

Файл: lab3_4.asm

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

; Данные
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_l
     or AL,30h
     mov [SI],AL
end_l:  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              ;(dx ax) = ax*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                 ;address
     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 address
     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  ;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


UNUSED_MEMORY_FREE 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
UNUSED_MEMORY_FREE endp


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

    jb      memory_fail ;cf = 1
    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 UNUSED_MEMORY_FREE
        call MCB

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

TESTPC_END:
TESTPC ENDS

END START
```