

**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

Тюленев Т.В.

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 не страничная 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, предусматривают и преобразуют этот список.

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

## **Постановка задачи.**

**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 4AH»). 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 48Н прерывания 21Н. 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 48Н прерывания 21Н до освобождения памяти. Обязательно обрабатывайте завершение функций ядра, проверяя флаг CF. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

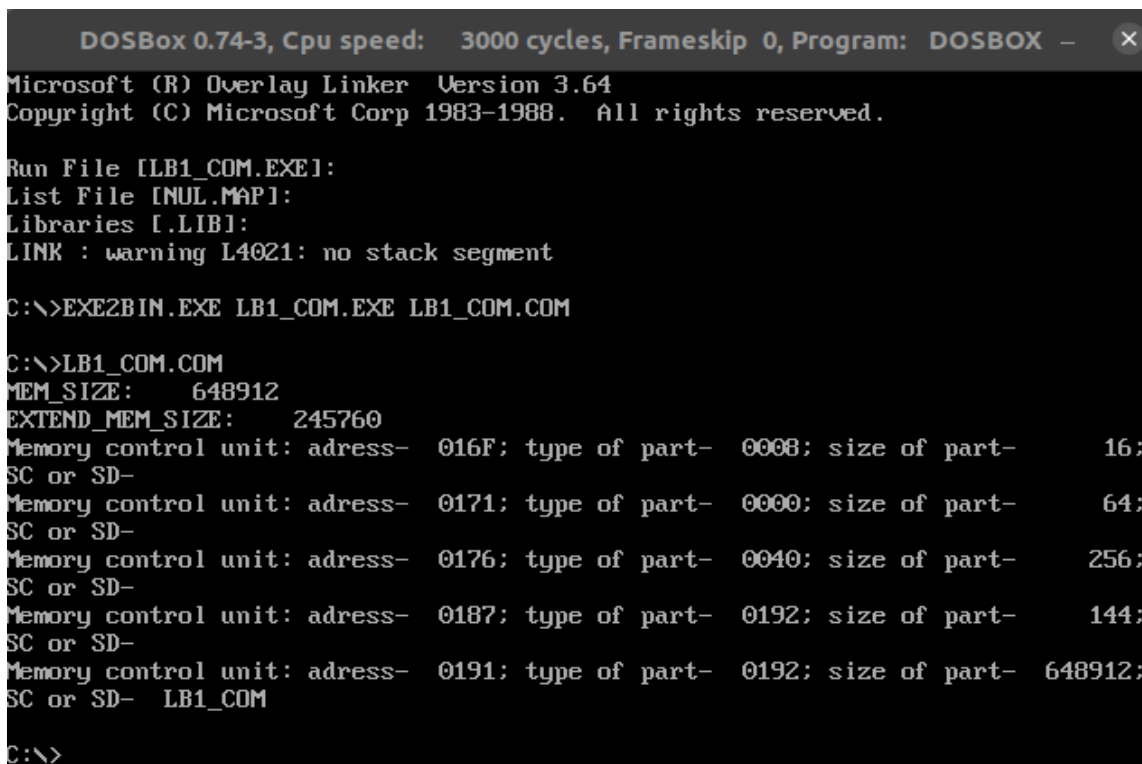
**Шаг 5.** Оцените результаты, полученные на предыдущих шагах. Ответьте на контрольные вопросы и оформите отчет

## Выполнение работы.

Исходные коды модулей представлены в приложении А.

**Шаг 1.** Были написаны все основные функции из задания, а именно PRINT\_MEM\_SIZE(выводит размер занимаемой памяти), PRINT\_EXTEND\_MEM\_SIZE(выводит размер расширенной памяти), PRINT\_MCU(выводит блик управления памятью), PRINT\_CHAIN\_MCU(выводит цепочку блоков управления памятью).

Результат работы программы, написанной на первом шаге представлен на рисунке 1.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Run File [LB1_COM.EXE]:
List File [NUL.MAP]:
Libraries [LIB1]:
LINK : warning L4021: no stack segment

C:\>EXE2BIN.EXE LB1_COM.EXE LB1_COM.COM

C:\>LB1_COM.COM
MEM_SIZE: 648912
EXTEND_MEM_SIZE: 245760
Memory control unit: adress- 016F; type of part- 0008; size of part- 16;
SC or SD-
Memory control unit: adress- 0171; type of part- 0000; size of part- 64;
SC or SD-
Memory control unit: adress- 0176; type of part- 0040; size of part- 256;
SC or SD-
Memory control unit: adress- 0187; type of part- 0192; size of part- 144;
SC or SD-
Memory control unit: adress- 0191; type of part- 0192; size of part- 648912;
SC or SD- LB1_COM

C:\>_
```

Рисунок 1 – Результат запуска модуля lb3.1.com

**Шаг 2.** На втором шаге была написана функция CLEAN\_MEM, которая освобождает память, не занимаемую программой.

Результат работы программы, написанной на втором шаге представлен на рисунке 2.

```

C:\>LB3_U2.COM
MEM_SIZE: 648912
EXTEND_MEM_SIZE: 245760
Memory control unit: adress- 016F; type of part- 0008; size of part- 16;
SC or SD-
Memory control unit: adress- 0171; type of part- 0000; size of part- 64;
SC or SD-
Memory control unit: adress- 0176; type of part- 0040; size of part- 256;
SC or SD-
Memory control unit: adress- 0187; type of part- 0192; size of part- 144;
SC or SD-
Memory control unit: adress- 0191; type of part- 0192; size of part- 816;
SC or SD- LB3_U2
Memory control unit: adress- 01C5; type of part- 0000; size of part- 648080;
SC or SD- ;x°u0i¹
C:\>

```

Рисунок 2 – Результат выполнения модуля lb3.2.com

Заметим, что на первом шаге программа занимала практически всю свободную память, однако на втором шаге свободная память выделилась в отдельный блок, который является последним на рисунке 2. Блок, в который загружена программа стал размеров, необходимых для хранения программы.

**Шаг 3.** Далее была написана функция EXTRA\_MEM, которая дополнительно запрашивает 64 Кб памяти, при помощи функции 48h прерывания int 21.

Результат выполнения программы, написанной на третьем шаге представлен на рисунке 3.

```

C:\>LB3_U3.COM
MEM_SIZE: 648912
EXTEND_MEM_SIZE: 245760
Memory control unit: adress- 016F; type of part- 0008; size of part- 16;
SC or SD-
Memory control unit: adress- 0171; type of part- 0000; size of part- 64;
SC or SD-
Memory control unit: adress- 0176; type of part- 0040; size of part- 256;
SC or SD-
Memory control unit: adress- 0187; type of part- 0192; size of part- 144;
SC or SD-
Memory control unit: adress- 0191; type of part- 0192; size of part- 832;
SC or SD- LB3_U3
Memory control unit: adress- 01C6; type of part- 0192; size of part- 65536;
SC or SD- LB3_U3
Memory control unit: adress- 11C7; type of part- 0000; size of part- 582512;
SC or SD-
↑COM

```

Рисунок 3 – Результат выполнения модуля lb3.3.com

Из изображения видно, что относительно предыдущего случая появился ещё один блок памяти, рассчитанный как на 64Кб, он выделился из свободной памяти.

**Шаг 4.** Отличие от предыдущего выполнения заключается в обратной очерёдности вызовов функций EXTRA\_MEM и CLEAN\_MEM(в данном случае сначала запрашивается 64Кб, а только потом высвобождается память)

Результат выполнения программы, написанной на четвёртом шаге представлен на рисунке 4.

```
C:\>LB3_U3.COM
MEM_SIZE:      648912
EXTEND_MEM_SIZE: 245760

      ??? WARNING !!!  ??? WARNING !!!  ??? WARNING !!!

Memory control unit: adress- 016F; type of part- 0008; size of part-    16;
SC or SD-
Memory control unit: adress- 0171; type of part- 0000; size of part-    64;
SC or SD-
Memory control unit: adress- 0176; type of part- 0040; size of part-   256;
SC or SD-
Memory control unit: adress- 0187; type of part- 0192; size of part-   144;
SC or SD-
Memory control unit: adress- 0191; type of part- 0192; size of part-   912;
SC or SD- LB3_U3
Memory control unit: adress- 01CB; type of part- 0000; size of part- 647984;
SC or SD- â·  ѵЅТ?
```

Рисунок 4 – Результат выполнения модуля lb3.3.com (изменённый)

В данной вариации выводится сообщение об ошибке, т. к. вся память уже принадлежит программе, поэтому дополнительные 64Кб выделены быть не могут.

### Контрольные вопросы.

Сегментный адрес недоступной памяти:

1. Что означает «доступный объём памяти»?

Это размер оперативной памяти в системе, который используется для запуска и выполнения программы.

2. Где MCB блок Вашей программы в списке?

На каждом шаге программе принадлежат МСВ блоки, в графе SD/SC которых написано название файла.

3. Какой размер памяти занимает программа в каждом случае?

Размер памяти, занимаемой программой, это сумма размеров всех блоков, принадлежащих программе.

- 1) Шаг 1 – 648912 байт
- 2) Шаг 2 – 816 байт
- 3) Шаг 3 – 66368 байт
- 4) Шаг 4 – 912 байта

### **Выводы.**

В ходе работы были исследованы структуры данных и работа функций управления памятью ядра операционной системы

## ПРИЛОЖЕНИЕ А.

### Исходный код модулей

#### lb3\_v1.asm:

```
; Шаблон текста программы на ассемблере для модуля типа .COM
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H

START: JMP BEGIN

; ДАННЫЕ
MEM_SIZE db 'MEM_SIZE:                ',0DH,0AH,'$'
EXTEND_MEM_SIZE db 'EXTEND_MEM_SIZE:                ',0DH,0AH,'$'
MCU db 'Memory control unit: adress-            ; type of part-            ;
size of part-            ; SC or SD-            ',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 шестн. 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
    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 10с/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
;-----

WRD_TO_DEC PROC near
; ax - paragraph, si - low digit of result
        push bx
        push dx

        mov bx, 16
        mul bx
        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
WRD_TO_DEC ENDP

print_st PROC near
        mov AH, 09h
        int 21h
        ret

```

```

print_st ENDP

print_symb PROC near
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
print_symb ENDP

PRINT_MEM_SIZE PROC near
    push ax
    push dx
    push cx
    push bx
    push si

    MOV AH,4AH
    MOV BX,0FFFFH ; заведомо большая память
    INT 21H

    mov si, offset MEM_SIZE
    add si, 18
    mov ax, bx
    call WRD_TO_DEC

    mov dx, offset MEM_SIZE
    call print_st

    pop si
    pop bx
    pop cx
    pop dx
    pop ax
    ret
PRINT_MEM_SIZE ENDP

PRINT_EXTEND_MEM_SIZE PROC near
    push ax
    push dx
    push cx
    push bx
    push si

    mov AL,30h ; запись адреса ячейки CMOS
    out 70h,AL
    in AL,71h ; чтение младшего байта
    mov BL,AL

    mov AL,31h ; запись адреса ячейки CMOS
    out 70h,AL
    in AL,71h
    mov BH,AL

    mov si, offset EXTEND_MEM_SIZE
    add si, 25
    mov ax, bx
    call WRD_TO_DEC

    mov dx, offset EXTEND_MEM_SIZE

```

```

    call print_st

    pop si
    pop bx
    pop cx
    pop dx
    pop ax
    ret
PRINT_EXTEND_MEM_SIZE ENDP

PRINT_MCU PROC near
    push ax
    push dx
    push cx
    push bx
    push si

    mov di, offset MCU
    add di, 33
    mov ax, es
    call WRD_TO_HEX

    mov di, offset MCU
    add di, 54
    mov ax, es:[01h]
    call WRD_TO_HEX

    mov si, offset MCU
    add si, 77
    mov ax, es:[03h]
    call WRD_TO_DEC

    mov di, offset MCU
    add di, 91
    mov si, 8

sc_sd:
    mov bx, es:[si]
    mov [di], bx
    add si, 2
    add di, 2
    cmp si, 16
    jb sc_sd

    mov dx, offset MCU
    call print_st

    pop si
    pop bx
    pop cx
    pop dx
    pop ax
    ret
PRINT_MCU ENDP

PRINT_CHAIN_MCU PROC near
    push ax
    push dx
    push cx
    push bx

```

```

    push si
    push es

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

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

end_:
    pop es
    pop si
    pop bx
    pop cx
    pop dx
    pop ax
    ret
PRINT_CHAIN_MCU ENDP

```

```

; КОД
BEGIN:
    call PRINT_MEM_SIZE
    call PRINT_EXTEND_MEM_SIZE
    call PRINT_CHAIN_MCU
; Выход в DOS
    xor AL,AL
    mov AH,4Ch
    int 21H
TESTPC ENDS
    END START ;конец модуля, START - точка входа

```

### lb3\_v2.asm:

```

; Шаблон текста программы на ассемблере для модуля типа .COM
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H

START: JMP BEGIN

; ДАННЫЕ
MEM_SIZE db 'MEM_SIZE:                                ',0DH,0AH,'$'
EXTEND_MEM_SIZE db 'EXTEND_MEM_SIZE:                    ',0DH,0AH,'$'
MCU db 'Memory control unit: adress-                    ; type of part-
size of part-          ; SC or SD-                        ',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 шестн. 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
    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 10с/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
;-----

WRD_TO_DEC PROC near
; ax - paragraph, si - low digit of result
        push bx
        push dx

        mov bx, 16
        mul bx
        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
WRD_TO_DEC ENDP

print_st PROC near
        mov AH, 09h
        int 21h
        ret
print_st ENDP

print_symb PROC near
        push ax
        mov ah, 02h
        int 21h
        pop ax
        ret
print_symb ENDP

PRINT_MEM_SIZE PROC near
        push ax
        push dx
        push cx
        push bx
        push si

        MOV AH,4AH
        MOV BX,0FFFFH ; заведомо большая память
        INT 21H

        mov si, offset MEM_SIZE
        add si, 18

```

```

    mov ax, bx
    call WRD_TO_DEC

    mov dx, offset MEM_SIZE
    call print_st

    pop si
    pop bx
    pop cx
    pop dx
    pop ax
    ret
PRINT_MEM_SIZE ENDP

PRINT_EXTEND_MEM_SIZE PROC near
    push ax
    push dx
    push cx
    push bx
    push si

    mov AL, 30h ; запись адреса ячейки CMOS
    out 70h, AL
    in AL, 71h ; чтение младшего байта
    mov BL, AL

    mov AL, 31h ; запись адреса ячейки CMOS
    out 70h, AL
    in AL, 71h
    mov BH, AL

    mov si, offset EXTEND_MEM_SIZE
    add si, 25
    mov ax, bx
    call WRD_TO_DEC

    mov dx, offset EXTEND_MEM_SIZE
    call print_st

    pop si
    pop bx
    pop cx
    pop dx
    pop ax
    ret
PRINT_EXTEND_MEM_SIZE ENDP

PRINT_MCU PROC near
    push ax
    push dx
    push cx
    push bx
    push si

    mov di, offset MCU
    add di, 33
    mov ax, es
    call WRD_TO_HEX

    mov di, offset MCU

```

```

    add di, 54
    mov ax, es:[01h]
    call WRD_TO_HEX

    mov si, offset MCU
    add si, 77
    mov ax, es:[03h]
    call WRD_TO_DEC

    mov di, offset MCU
    add di, 91
    mov si, 8

sc_sd:
    mov bx, es:[si]
    mov [di], bx
    add si, 2
    add di, 2
    cmp si, 16
    jb sc_sd

    mov dx, offset MCU
    call print_st

    pop si
    pop bx
    pop cx
    pop dx
    pop ax
    ret
PRINT_MCU ENDP

PRINT_CHAIN_MCU PROC near
    push ax
    push dx
    push cx
    push bx
    push si
    push es

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

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

end_:
    pop es
    pop si
    pop bx
    pop cx

```



```

        pop dx
        pop ax
        ret
PRINT_CHAIN_MCU ENDP

CLEAN_MEM PROC near
        push ax
        push bx
        push dx

        mov ax, offset FINAL_LB3
        mov bx, 10h
        xor dx, dx
        div bx
        inc ax
        mov bx, ax
        mov al, 0
        mov ah, 4ah
        int 21h

        pop dx
        pop bx
        pop ax
        ret
CLEAN_MEM ENDP

; КОД
BEGIN:
        call PRINT_MEM_SIZE
        call PRINT_EXTEND_MEM_SIZE
        call CLEAN_MEM
        call PRINT_CHAIN_MCU
; Выход в DOS
        xor AL,AL
        mov AH,4Ch
        int 21H
FINAL_LB3:
TESTPC ENDS
        END START ;конец модуля, START - точка входа

```

### lb3\_v3.asm:

```

; Шаблон текста программы на ассемблере для модуля типа .COM
TESTPC SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG 100H

START: JMP BEGIN

; ДАННЫЕ
MEM_SIZE db 'MEM_SIZE:                                ',0DH,0AH,'$'
EXTEND_MEM_SIZE db 'EXTEND_MEM_SIZE:                                ',0DH,0AH,'$'
MCU db 'Memory control unit: adress-                                ; type of part-                                ;
size of part-                                ; SC or SD-                                ',0DH,0AH,'$'
WARNING db 0DH,0AH, '                                !!! WARNING !!!                                !!!'
WARNING !!!                                !!! WARNING !!!',0DH,0AH,0DH,0AH,'$'

;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 шестн. 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
    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 10с/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
;-----

WRD_TO_DEC PROC near
; ax - paragraph, si - low digit of result
        push bx
        push dx

        mov bx, 16
        mul bx
        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
WRD_TO_DEC ENDP

print_st PROC near
        mov AH, 09h
        int 21h
        ret
print_st ENDP

print_symb PROC near
        push ax
        mov ah, 02h
        int 21h
        pop ax
        ret
print_symb ENDP

PRINT_MEM_SIZE PROC near
        push ax
        push dx
        push cx
        push bx
        push si

        MOV AH,4AH
        MOV BX,0FFFFH ; заведомо большая память
        INT 21H

        mov si, offset MEM_SIZE
        add si, 18
        mov ax, bx

```

```

    call WRD_TO_DEC

    mov dx, offset MEM_SIZE
    call print_st

    pop si
    pop bx
    pop cx
    pop dx
    pop ax
    ret
PRINT_MEM_SIZE ENDP

PRINT_EXTEND_MEM_SIZE PROC near
    push ax
    push dx
    push cx
    push bx
    push si

    mov AL,30h ; запись адреса ячейки CMOS
    out 70h,AL
    in AL,71h ; чтение младшего байта
    mov BL,AL

    mov AL,31h ; запись адреса ячейки CMOS
    out 70h,AL
    in AL,71h
    mov BH,AL

    mov si, offset EXTEND_MEM_SIZE
    add si, 25
    mov ax, bx
    call WRD_TO_DEC

    mov dx, offset EXTEND_MEM_SIZE
    call print_st

    pop si
    pop bx
    pop cx
    pop dx
    pop ax
    ret
PRINT_EXTEND_MEM_SIZE ENDP

PRINT_MCU PROC near
    push ax
    push dx
    push cx
    push bx
    push si

    mov di, offset MCU
    add di, 33
    mov ax, es
    call WRD_TO_HEX

    mov di, offset MCU
    add di, 54

```

```

    mov ax, es:[01h]
    call WRD_TO_HEX

    mov si, offset MCU
    add si, 77
    mov ax, es:[03h]
    call WRD_TO_DEC

    mov di, offset MCU
    add di, 91
    mov si, 8

sc_sd:
    mov bx, es:[si]
    mov [di], bx
    add si, 2
    add di, 2
    cmp si, 16
    jb sc_sd

    mov dx, offset MCU
    call print_st

    pop si
    pop bx
    pop cx
    pop dx
    pop ax
    ret
PRINT_MCU ENDP

PRINT_CHAIN_MCU PROC near
    push ax
    push dx
    push cx
    push bx
    push si
    push es

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

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

end_:
    pop es
    pop si
    pop bx
    pop cx
    pop dx

```

```

        pop ax
        ret
PRINT_CHAIN_MCU ENDP

CLEAN_MEM PROC near
    push ax
    push bx
    push dx

    mov ax, offset FINAL_LB3
    mov bx, 10h
    xor dx, dx
    div bx
    inc ax
    mov bx, ax
    mov al, 0
    mov ah, 4ah
    int 21h

    pop dx
    pop bx
    pop ax
    ret
CLEAN_MEM ENDP

EXTRA_MEM PROC near
    push ax
    push bx
    push dx

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

    jnc end_ask
    mov dx, offset WARNING
    call print_st

end_ask:
    pop dx
    pop bx
    pop cx
    ret
EXTRA_MEM ENDP

; КОД
BEGIN:
    call PRINT_MEM_SIZE
    call PRINT_EXTEND_MEM_SIZE
    call EXTRA_MEM
    call CLEAN_MEM
    call PRINT_CHAIN_MCU
; Выход в DOS
    xor AL, AL
    mov AH, 4Ch
    int 21H
FINAL_LB3:
TESTPC ENDS
    END START ;конец модуля, START - точка входа

```