

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 память

Студент гр. 9382

Рыжих Р.В.

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.

Необходимые сведения для составления программы.

Учет занятой и свободной памяти ведется при помощи списка блоков управления памятью MCB (Memory Control Block). MCB занимает 16 байт (параграф) и располагается всегда с адреса кратного 16 (адрес сегмента ОП) и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет.

По сегментному адресу и размеру участка памяти, контролируемого этим MCB можно определить местоположение следующего MCB в списке.

Адрес первого MCB хранится во внутренней структуре MS DOS, называемой "List of Lists" (список списков). Доступ к указателю на эту структуру можно получить используя функцию f52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX-2] и есть адрес самого первого MCB.

Размер расширенной памяти находится в ячейках 30h, 31h CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ.

Объем памяти составляет 64 байта. Размер расширенной памяти в Кбайтах можно определить обращаясь к ячейкам CMOS следующим образом:

```
mov AL,30h ; запись адреса ячейки CMOS
out 70h,AL
in AL,71h ; чтение младшего байта
mov BL,AL ; размера расширенной памяти
mov AL,31h ; запись адреса ячейки CMOS
out 70h,AL
in AL,71h ; чтение старшего байта
; размера расширенной памяти
```

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 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 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.

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
WRITESTRING	Вывод строки на экран

PRINT_AVB_MEM	Печать на экран количества доступной памяти
PRINT_EXP_MEM	Печать на экран размера расширенной памяти
FREE_MEM	Отчистка свободной памяти
PRINT_BLOCK_CHAIN	Печать на экран цепочки блоков управления памятью
PRINT_MCB	Печать на экран списка блоков управления памятью

Выполнение работы.

1. Был написан COM модуль, который выводит:
 - 1) Количество доступной памяти.
 - 2) Размер расширенной памяти.
 - 3) Выводит цепочку блоков управления памятью

```

C:\>LAB3_1.COM
Available memory (b):648912
Expanded memory (Kb): 15420
MCB:Address:016F   Address PSP:0008   Size:      16   SC/SD:
MCB:Address:0171   Address PSP:0000   Size:       64   SC/SD:
MCB:Address:0176   Address PSP:0040   Size:     256   SC/SD:
MCB:Address:0187   Address PSP:0192   Size:     144   SC/SD:
MCB:Address:0191   Address PSP:0192   Size: 648912   SC/SD:LAB3_1

```

Рис.1 – Работа COM модуля Lab3_1.asm

2. COM модуль был изменён так, что он освобождает память, которую программа не занимает.

```

C:\>LAB3_2.COM
Available memory (b):648912
Expanded memory (Kb): 15420
MCB:Address:016F   Address PSP:0008   Size:      16   SC/SD:
MCB:Address:0171   Address PSP:0000   Size:       64   SC/SD:
MCB:Address:0176   Address PSP:0040   Size:     256   SC/SD:
MCB:Address:0187   Address PSP:0192   Size:     144   SC/SD:
MCB:Address:0191   Address PSP:0192   Size:     7168   SC/SD:LAB3_2
MCB:Address:0352   Address PSP:0000   Size: 641728   SC/SD:

```

Рис.2 – Работа COM модуля Lab3_2.asm

Освобождённую память можно видеть в последней строке, а память, которую занимает сама программа – в предпоследней. В этом случае, программа занимает 7 168 байт.

3. COM модуль был изменён так, что он после освобождения память запрашивает 64Кб памяти.

```
C:\>LAB3_3.COM
Available memory (b):648912
Expanded memory (Kb): 15420
MCB:Address:016F   Address PSP:0008   Size:      16   SC/SD:
MCB:Address:0171   Address PSP:0000   Size:       64   SC/SD:
MCB:Address:0176   Address PSP:0040   Size:     256   SC/SD:
MCB:Address:0187   Address PSP:0192   Size:     144   SC/SD:
MCB:Address:0191   Address PSP:0192   Size:    7168   SC/SD:LAB3_3
MCB:Address:0352   Address PSP:0192   Size:   65536   SC/SD:LAB3_3
MCB:Address:1353   Address PSP:0000   Size:  576176   SC/SD:
```

Рис.3 – Работа COM модуля Lab3_3.asm

4. COM модуль был изменён так, что теперь он запрашивает 64Кб памяти до её освобождение. Также, ставится проверка на то, выделилась ли память, или произошла ошибка.

```
C:\>LAB3_4.COM
Available memory (b):648912
Expanded memory (Kb): 15420
Memory allocation error!
MCB:Address:016F   Address PSP:0008   Size:      16   SC/SD:
MCB:Address:0171   Address PSP:0000   Size:       64   SC/SD:
MCB:Address:0176   Address PSP:0040   Size:     256   SC/SD:
MCB:Address:0187   Address PSP:0192   Size:     144   SC/SD:
MCB:Address:0191   Address PSP:0192   Size:  648912   SC/SD:LAB3_4
```

Рис.4 – Работа COM модуля Lab3_4.asm

Вывелось сообщение об ошибке выделения памяти. В таком случае, флаг CF поменялся, а программе выделился максимальный блок памяти.

Выводы.

В ходе выполнения лабораторной работы была написана программа для вывода некоторой информации о памяти; Были изучены методы управления разделами памяти; Были изучены функции управления памятью ядра операционной системы.

Контрольные вопросы по лабораторной работе №3

1) Что означает "доступный объем памяти"?

Доступный объем памяти – это область памяти, которая не занята процессами системы и может выделяться для использования.

2) **Где МСВ блок Вашей программы в списке?**

МСВ блок программы располагается по адресу 0192.

3) **Какой размер памяти занимает программа в каждом случае.**

На 1 шаге программа занимает 648 912 байт.

На 2 шаге программа занимает 7 168 байт.

На 3 шаге программа занимает 7 168 байт.

На 4 шаге программа занимает 648 912 байт.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Lab3_1.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN
; Данные
    avb_mem db 'Available memory (b):
',0DH,0AH,'$'
    exp_mem db 'Expanded memory (Kb):
',0DH,0AH,'$'
    new_str db 0DH,0AH,'$'
    adr db 'Address: ', '$'
    mcb db 'MCB:', '$'
    adr_psp db 'Address PSP: ', '$'
    area_size db 'Size: ', '$'
    sc_sd db 'SC/SD:', '$'
; 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
    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
;-----
WRITESTRING PROC near
    push ax
    mov AH,09h
    int 21h
    pop ax
    ret
WRITESTRING ENDP
;-----
;-----
PRINT_AVB_MEM PROC near
    mov ah, 4ah
    mov bx, 0FFFFh
    int 21h
    mov ax, 10h
    mul bx
    mov si, offset avb_mem
    add si, 26
    call BYTE_TO_DEC
    mov dx, offset avb_mem
    call WRITESTRING
    xor dx,dx
    ret
PRINT_AVB_MEM ENDP
;-----
PRINT_EXP_MEM PROC near
    mov AL,30h
    out 70h,AL
    in AL,71h
    mov BL,AL;
    mov AL,31h
    out 70h,AL
    in AL,71h
    mov ah, al
    mov si, offset exp_mem
    add si, 26

```

```

    call BYTE_TO_DEC
    mov dx, offset exp_mem
    call WRITESTRING
    ret
PRINT_EXP_MEM ENDP
;-----
PRINT_MCB PROC near
    push ax
    push bx
    push es
    push dx
    push cx

    mov dx, offset mcb
    call WRITESTRING

    mov di, offset adr
    add di, 11
    call WRD_TO_HEX
    mov dx, offset adr
    call WRITESTRING

    mov ax, es:[1]
    mov di, offset adr_psp
    add di, 15
    call WRD_TO_HEX
    mov dx, offset adr_psp
    call WRITESTRING

    mov ax, es:[3]
    mov bx, 10h
    mul bx
    mov si, offset area_size
    add si, 11
    call BYTE_TO_DEC
    mov dx, offset area_size
    call WRITESTRING

    mov dx, offset sc_sd
    call WRITESTRING
    mov bx, 8
    mov cx, 7

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

    pop cx
    pop dx
    pop es
    pop bx
    pop ax
    ret
PRINT_MCB ENDP
;-----
PRINT_BLOCK_CHAIN PROC near
    push ax
    push bx
    push es
    push dx

```

```

    push cx

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

print:
    call print_mcb
    mov dx, offset new_str
    call WRITESTRING
    mov al, es:[0]
    cmp al, 5ah
je end_print

    mov bx, es:[3]
    mov ax, es
    add ax, bx
    inc ax
    mov es, ax
    jmp print

end_print:
    pop cx
    pop dx
    pop es
    pop bx
    pop ax
ret
PRINT_BLOCK_CHAIN ENDP
;-----
; Код
BEGIN:
    call PRINT_AVB_MEM
    call PRINT_EXP_MEM
    call PRINT_BLOCK_CHAIN
    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
; Данные
    avb_mem db 'Available memory (b):',0DH,0AH,'$'
    exp_mem db 'Expanded memory (Kb):',0DH,0AH,'$'
    new_str db 0DH,0AH,'$'
    adr db 'Address: ','$'
    mcb db 'MCB:', '$'
    adr_psp db 'Address PSP: ','$'
    area_size db 'Size: ','$'
    sc_sd db 'SC/SD:', '$'
    free db 0
; 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
        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
;-----
WRITESTRING PROC near
    push ax
    mov AH,09h
    int 21h
    pop ax
    ret
WRITESTRING ENDP
;-----

;-----
PRINT_AVB_MEM PROC near
    mov ah, 4ah
    mov bx, 0FFFFh
    int 21h
    mov ax, 10h
    mul bx
    mov si, offset avb_mem
    add si, 26
    call BYTE_TO_DEC
    mov dx, offset avb_mem
    call WRITESTRING
    xor dx,dx
    ret
PRINT_AVB_MEM ENDP
;-----
PRINT_EXP_MEM PROC near
    mov AL,30h
    out 70h,AL
    in AL,71h
    mov BL,AL;
    mov AL,31h
    out 70h,AL
    in AL,71h
    mov ah, al
    mov si, offset exp_mem
    add si, 26
    call BYTE_TO_DEC
    mov dx, offset exp_mem
    call WRITESTRING
    ret
PRINT_EXP_MEM ENDP
;-----
PRINT_MCB PROC near
    push ax
    push bx
    push es
    push dx
    push cx

    mov dx, offset mcb
    call WRITESTRING

    mov di, offset adr
    add di, 11
    call WRD_TO_HEX
    mov dx, offset adr
    call WRITESTRING

    mov ax, es:[1]
    mov di, offset adr_psp
    add di, 15

```

```

    call WRD_TO_HEX
    mov dx, offset adr_psp
    call WRITESTRING

    mov ax, es:[3]
    mov bx, 10h
    mul bx
    mov si, offset area_size
    add si, 11
    call BYTE_TO_DEC
    mov dx, offset area_size
    call WRITESTRING

    mov dx, offset sc_sd
    call WRITESTRING
    mov bx, 8
    mov cx, 7

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

    pop cx
    pop dx
    pop es
    pop bx
    pop ax
ret
PRINT_MCB ENDP
;-----
PRINT_BLOCK_CHAIN PROC near
    push ax
    push bx
    push es
    push dx
    push cx

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

print:
    call print_mcb
    mov dx, offset new_str
    call WRITESTRING
    mov al, es:[0]
    cmp al, 5ah
    je end_print

    mov bx, es:[3]
    mov ax, es
    add ax, bx
    inc ax
    mov es, ax
    jmp print

end_print:
    pop cx
    pop dx

```

```

    pop es
    pop bx
    pop ax
ret
PRINT_BLOCK_CHAIN ENDP
;-----
FREE_MEM PROC near
    mov ah, 4ah
    mov bx, offset free
    int 21h
ret
FREE_MEM ENDP
;-----
; Код
BEGIN:
    call PRINT_AVB_MEM
    call PRINT_EXP_MEM
    call FREE_MEM
    call PRINT_BLOCK_CHAIN
    xor AL,AL
    mov AH,4Ch
    int 21h
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
; Данные
    avb_mem db 'Available memory (b):',0DH,0AH,'$'
    exp_mem db 'Expanded memory (Kb):',0DH,0AH,'$'
    new_str db 0DH,0AH,'$'
    adr db 'Address: ','$'
    mcb db 'MCB:', '$'
    adr_psp db 'Address PSP: ','$'
    area_size db 'Size: ','$'
    sc_sd db 'SC/SD:', '$'
    free db 0

; 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
        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
;-----
WRITESTRING PROC near
        push ax
        mov AH,09h
        int 21h
        pop ax
        ret
WRITESTRING ENDP
;-----
;-----
PRINT_AVB_MEM PROC near
        mov ah, 4ah
        mov bx, 0FFFFh
        int 21h

```



```

    mov ax, 10h
    mul bx
    mov si, offset avb_mem
    add si, 26
    call BYTE_TO_DEC
    mov dx, offset avb_mem
    call WRITESTRING
    xor dx,dx
    ret
PRINT_AVB_MEM ENDP
;-----
PRINT_EXP_MEM PROC near
    mov AL,30h
    out 70h,AL
    in AL,71h
    mov BL,AL;
    mov AL,31h
    out 70h,AL
    in AL,71h
    mov ah, al
    mov si, offset exp_mem
    add si, 26
    call BYTE_TO_DEC
    mov dx, offset exp_mem
    call WRITESTRING
    ret
PRINT_EXP_MEM ENDP
;-----
PRINT_MCB PROC near
    push ax
    push bx
    push es
    push dx
    push cx

    mov dx, offset mcb
    call WRITESTRING

    mov di, offset adr
    add di, 11
    call WRD_TO_HEX
    mov dx, offset adr
    call WRITESTRING

    mov ax, es:[1]
    mov di, offset adr_psp
    add di, 15
    call WRD_TO_HEX
    mov dx, offset adr_psp
    call WRITESTRING

    mov ax, es:[3]
    mov bx, 10h
    mul bx
    mov si, offset area_size
    add si, 11
    call BYTE_TO_DEC
    mov dx, offset area_size
    call WRITESTRING

    mov dx, offset sc_sd
    call WRITESTRING
    mov bx, 8

```

```

        mov cx, 7

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

    pop cx
    pop dx
    pop es
    pop bx
    pop ax
ret
PRINT_MCB ENDP
;-----
PRINT_BLOCK_CHAIN PROC near
    push ax
    push bx
    push es
    push dx
    push cx

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

print:
    call print_mcb
    mov dx, offset new_str
    call WRITESTRING
    mov al, es:[0]
    cmp al, 5ah
je end_print

    mov bx, es:[3]
    mov ax, es
    add ax, bx
    inc ax
    mov es, ax
    jmp print

end_print:
    pop cx
    pop dx
    pop es
    pop bx
    pop ax
ret
PRINT_BLOCK_CHAIN ENDP
;-----
FREE_MEM PROC near
    mov ah, 4ah
    mov bx, offset free
    int 21h
ret
FREE_MEM ENDP
;-----
GET_MEM PROC near
    mov ah, 48h
    mov bx, 1000h

```

```

        int 21h
ret
GET_MEM ENDP
;-----
; Код
BEGIN:
    call PRINT_AVB_MEM
    call PRINT_EXP_MEM
    call FREE_MEM
    call GET_MEM
    call PRINT_BLOCK_CHAIN
    xor AL,AL
    mov AH,4Ch
    int 21H
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
; Данные
    avb_mem db 'Available memory (b):
',0DH,0AH,'$'
    exp_mem db 'Expanded memory (Kb):
',0DH,0AH,'$'
    new_str db 0DH,0AH,'$'
    adr db 'Address:      ','$'
    mcb db 'MCB:', '$'
    adr_psp db 'Address PSP:      ','$'
    area_size db 'Size:      ','$'
    sc_sd db 'SC/SD:', '$'
    free db 0

; 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
    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
;-----
WRITESTRING PROC near
    push ax
    mov AH,09h
    int 21h
    pop ax
    ret
WRITESTRING ENDP
;-----
;-----
PRINT_AVB_MEM PROC near
    mov ah, 4ah
    mov bx, 0FFFFh
    int 21h
    mov ax, 10h
    mul bx
    mov si, offset avb_mem
    add si, 26
    call BYTE_TO_DEC
    mov dx, offset avb_mem
    call WRITESTRING
    xor dx,dx

```

```

    ret
PRINT_AVB_MEM ENDP
;-----
PRINT_EXP_MEM PROC near
    mov AL,30h
    out 70h,AL
    in AL,71h
    mov BL,AL;
    mov AL,31h
    out 70h,AL
    in AL,71h
    mov ah, al
    mov si, offset exp_mem
    add si, 26
    call BYTE_TO_DEC
    mov dx, offset exp_mem
    call WRITESTRING
    ret
PRINT_EXP_MEM ENDP
;-----
PRINT_MCB PROC near
    push ax
    push bx
    push es
    push dx
    push cx

    mov dx, offset mcb
    call WRITESTRING

    mov di, offset adr
    add di, 11
    call WRD_TO_HEX
    mov dx, offset adr
    call WRITESTRING

    mov ax, es:[1]
    mov di, offset adr_psp
    add di, 15
    call WRD_TO_HEX
    mov dx, offset adr_psp
    call WRITESTRING

    mov ax, es:[3]
    mov bx, 10h
    mul bx
    mov si, offset area_size
    add si, 11
    call BYTE_TO_DEC
    mov dx, offset area_size
    call WRITESTRING

    mov dx, offset sc_sd
    call WRITESTRING
    mov bx, 8
    mov cx, 7

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

```

```

        pop cx
        pop dx
        pop es
        pop bx
        pop ax
    ret
PRINT_MCB ENDP
;-----
PRINT_BLOCK_CHAIN PROC near
    push ax
    push bx
    push es
    push dx
    push cx

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

print:
    call print_mcb
    mov dx, offset new_str
    call WRITESTRING
    mov al, es:[0]
    cmp al, 5ah
je end_print

    mov bx, es:[3]
    mov ax, es
    add ax, bx
    inc ax
    mov es, ax
    jmp print

end_print:
    pop cx
    pop dx
    pop es
    pop bx
    pop ax
    ret
PRINT_BLOCK_CHAIN ENDP
;-----
FREE_MEM PROC near
    mov ah, 4ah
    mov bx, offset free
    int 21h
    ret
FREE_MEM ENDP
;-----
GET_MEM PROC near
    mov ah, 48h
    mov bx, 1000h
    int 21h
    ret
GET_MEM ENDP
;-----
; Код
BEGIN:
    call PRINT_AVB_MEM
    call PRINT_EXP_MEM

```

```
call FREE_MEM
call GET_MEM
call PRINT_BLOCK_CHAIN
xor AL,AL
mov AH,4Ch
int 21H
TESTPC ENDS
END START
```