

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

**ОТЧЕТ**  
**по лабораторной работе №3**  
**по дисциплине «Операционные системы»**  
**Тема: Исследование организации управления основной памятью**

Студент гр. 0382

\_\_\_\_\_

Корсунов А.А.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2022

## Постановка задачи

### Цель работы.

Исследование организации управления основной памятью, изучение нестраничной памяти, исследование структур данных и работы функций управления памятью ядра операционной системы.

### Сведения о функциях и структурах данных управляющей программы.

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа ASCII (результат записывается в AL)
BYTE_TO_HEX	Перевод байта из AL переводится в два символа шестн. числа в AX (в AL старшая цифра в AH младшая цифра)
WRD_TO_HEX	Перевод в 16 с/с 16-ти разрядного числа (в AX – число, DI – адрес последнего символа)
BYTE_TO_DEC	Перевод в 10 с/с (SI – адрес поля младшей цифры)
WRITE_MESSAGE_WORD	Вывод строки на экран
WRITE_MESSAGE_BYTE	Вывод символа на экран
TO_DEC	Объем параграфа, преобразованный в байты, выводится в виде десятичных чисел
PRINT_AVAILABLE_MEMORY	Вывод количества доступной памяти
PRINT_EXTENDED_MEMORY	Вывод количества расширенной памяти
PRINT_MCB_TABLE	Вывод цепочки блоков управления 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. 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.

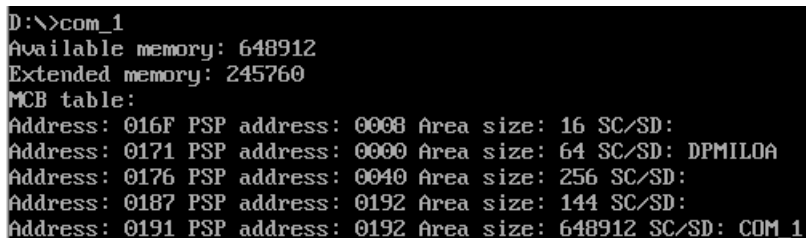
**Шаг 4.** Измените первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти. Обязательно обрабатывайте завершение функций ядра, проверяя флаг CF. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

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

### Выполнение работы

**Шаг 1.** Был написан и отлажен программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

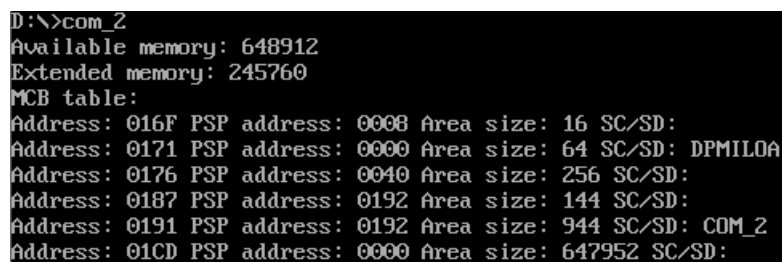
1. Количество доступной памяти.
2. Размер расширенной памяти.
3. Цепочку блоков управления памятью.



```
D:\>com_1
Available memory: 648912
Extended memory: 245760
MCB table:
Address: 016F PSP address: 0008 Area size: 16 SC/SD:
Address: 0171 PSP address: 0000 Area size: 64 SC/SD: DPMILOA
Address: 0176 PSP address: 0040 Area size: 256 SC/SD:
Address: 0187 PSP address: 0192 Area size: 144 SC/SD:
Address: 0191 PSP address: 0192 Area size: 648912 SC/SD: COM_1
```

Рисунок 1 — иллюстрация работы программного модуля из шага 1

**Шаг 2.** Программа была изменена таким образом, что она освобождает память, которую она не занимает.



```
D:\>com_2
Available memory: 648912
Extended memory: 245760
MCB table:
Address: 016F PSP address: 0008 Area size: 16 SC/SD:
Address: 0171 PSP address: 0000 Area size: 64 SC/SD: DPMILOA
Address: 0176 PSP address: 0040 Area size: 256 SC/SD:
Address: 0187 PSP address: 0192 Area size: 144 SC/SD:
Address: 0191 PSP address: 0192 Area size: 944 SC/SD: COM_2
Address: 01CD PSP address: 0000 Area size: 647952 SC/SD:
```

Рисунок 2 — иллюстрация работы программного модуля из шага 2

**Шаг 3.** Программа была изменена еще раз таким образом, что после освобождения памяти, программа запрашивает 64Кб памяти функцией 48H прерывания 21H.

```

D:\>com_3
Available memory: 648912
Extended memory: 245760
MCB table:
Address: 016F PSP address: 0008 Area size: 16 SC/SD:
Address: 0171 PSP address: 0000 Area size: 64 SC/SD: DPMILOA
Address: 0176 PSP address: 0040 Area size: 256 SC/SD:
Address: 0187 PSP address: 0192 Area size: 144 SC/SD:
Address: 0191 PSP address: 0192 Area size: 944 SC/SD: COM_3
Address: 01CD PSP address: 0192 Area size: 65536 SC/SD: COM_3
Address: 11CE PSP address: 0000 Area size: 582400 SC/SD: k±

```

Рисунок 3 - иллюстрация работы программного модуля из шага 3

**Шаг 4.** Был изменен первоначальный вариант программы, 64Кб памяти запрашивается функцией 48H прерывания 21H до освобождения памяти.

```

D:\>com_4
Available memory: 648912
Extended memory: 245760
Error... memory could not be allocated :(
MCB table:
Address: 016F PSP address: 0008 Area size: 16 SC/SD:
Address: 0171 PSP address: 0000 Area size: 64 SC/SD: DPMILOA
Address: 0176 PSP address: 0040 Area size: 256 SC/SD:
Address: 0187 PSP address: 0192 Area size: 144 SC/SD:
Address: 0191 PSP address: 0192 Area size: 1024 SC/SD: COM_4
Address: 01D2 PSP address: 0000 Area size: 647872 SC/SD: or

```

Рисунок 4 - иллюстрация работы программного модуля из шага 4

**Шаг 5.** Были предоставлены ответы на контрольные вопросы:

1) Что означает "доступный объем памяти"?

Ответ. Доступный объем памяти — это часть оперативной памяти, занимаемая и используемая программой.

2) Где MCB блок Вашей программы в списке?

Ответ.

Шаг 1 — пятый сверху (последний по списку, потому как в этой программе не было высвобождение неиспользованной памяти);

Шаг 2 — пятый сверху (предпоследний по списку, потому как в этой программе произошло высвобождение неиспользованной памяти, в последнем блоке — неиспользованная память);

Шаг 3 — пятый и шестой (потому как в этой программе произошло высвобождение неиспользованной памяти и выделило новый блок памяти в 64 кб);

Шаг 4 — пятый сверху (потому как программе не удалось выделить память до высвобождения, потому что весь доступный объем памяти занимает сама программа.

3) Какой размер памяти занимает программа в каждом случае?

Шаг 1 — программа занимает весь доступный объем памяти;

Шаг 2 — программа занимает только тот объем памяти, который ей необходим (944 байт);

Шаг 3 — программа занимает необходимый и запрошенный объемы памяти (944 байт + 64 килобайт);

Шаг 4 — программа не смогла выделить память, поэтому занимает тот объем памяти, который ей необходим (1024 байт);

### **Вывод.**

Было произведено исследование организации управления основной памятью, было изучено нестраничную память, исследование структур данных и работы функций управления памятью ядра операционной системы.

ПРИЛОЖЕНИЕ А  
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл com\_1.asm:

*TESTPC     SEGMENT*

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

*ORG 100H*

*START:     JMP BEGIN*

*AVAILABLE\_MEMORY db 'Available memory: \$'*

*EXTENDED\_MEMORY db 'Extended memory: \$'*

*MCB\_TABLE db 'MCB table:', 0DH, 0AH, '\$'*

*TABLE\_ADDRESS db 'Address:     \$'*

*SPACE db ' \$'*

*PSP\_ADDRESS db 'PSP address:     \$'*

*AREA\_SIZE db 'Area size: \$'*

*SC\_SD db 'SC/SD: \$'*

*TETR\_TO\_HEX PROC near*

*and AL, 0Fh*

*cmp AL, 09*

*jbe NEXT*

*add AL, 07*

*NEXT:     add AL, 30h*

*ret*

*TETR\_TO\_HEX ENDP*

*BYTE\_TO\_HEX    PROC near*

*push CX*

*mov AH, AL*

```

    call TETR_TO_HEX
    xchg AL, AH
    mov CL, 4
    shr AL, CL
    call TETR_TO_HEX
    pop CX
    ret
BYTE_TO_HEX ENDP

WRD_TO_HEX PROC near
    push BX
    mov BH, AH
    call BYTE_TO_HEX
    mov [DI], AH
    dec DI
    mov [DI], AL
    dec DI
    mov AL, BH
    call BYTE_TO_HEX
    mov [DI], AH
    dec DI
    mov [DI], AL
    pop BX
    ret
WRD_TO_HEX ENDP

BYTE_TO_DEC PROC near
    push CX
    push DX
    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

```

```

WRITE_MESSAGE_BYTE PROC near
push AX
mov AH, 02h
int 21h
pop AX
ret
WRITE_MESSAGE_BYTE ENDP

```

```

WRITE_MESSAGE_WORD PROC near ;вывести строку
push AX
mov AH, 09h
int 21h

```

```
pop AX
ret
WRITE_MESSAGE_WORD ENDP
```

*TO\_DEC PROC near ; доступная память в десятичном виде*

```
push BX
push CX
push AX
push DX
push DI
```

```
xor CX, CX
mov BX, 10
```

*lp1:*

```
div BX
```

```
push DX
```

```
xor DX, DX
```

```
inc CX
```

```
cmp AX, 0
```

*jne lp1 ;получение остатка числа при делении на 10 пока не  
останется 0, все заносится в стек*

*lp2:*

```
pop DX
```

```
add DL, '0' ;
```

```
call WRITE_MESSAGE_BYTE
```

```
loop lp2 ;извлечение из стека чисел, перевод в символы и вывод
```

```
pop DI
```

*pop DX*  
*pop AX*  
*pop CX*  
*pop BX*  
*ret*  
*TO\_DEC ENDP*

*PRINT\_AVAILABLE\_MEMORY PROC near ;вывести свободную память*  
*push DX*  
*push AX*  
*push BX*

*mov DX, offset AVAILABLE\_MEMORY*  
*call WRITE\_MESSAGE\_WORD*

*mov AH, 4ah ;освободить неиспользованную память, в BX заносится*  
*доступная память под программу*

*mov BX, 0ffffh*  
*int 21h*

*mov AX, BX ;в AX заносится доступная память под программу (в*  
*параграфах)*

*mov BX, 16*

*mul BX ;в BX заносится 16, в AX заносится результат перемножения*  
*содержимых AX\*BX (перевод параграфов в байты)*

*call TO\_DEC*

*mov DL, 0dh*  
*call WRITE\_MESSAGE\_BYTE*

*mov DL, 0ah*

*call WRITE\_MESSAGE\_BYTE*

*pop BX*

*pop AX*

*pop DX*

*ret*

*PRINT\_AVAILABLE\_MEMORY ENDP*

*PRINT\_EXTENDED\_MEMORY PROC near*

*push DX*

*push AX*

*push BX*

*mov DX, offset EXTENDED\_MEMORY*

*call WRITE\_MESSAGE\_WORD*

*mov AL, 30h ; запись адреса ячейки*

*out 70h, AL ; 30h в выходной порт 70h*

*in AL, 71h ; чтение младшего байта размера расширенной памяти (из входного порта в AL)*

*mov BL, AL ; в BL младший байт*

*mov AL, 31h*

*out 70h, AL*

*in AL, 71h ; чтение старшего байта размера расширенной памяти*

*mov BH, AL ; в BH старший байт*

*mov AX, BX*

*mov BX, 16*

*mul BX*

*call TO\_DEC*

*mov DL, 0dh*

*call WRITE\_MESSAGE\_BYTE*

*mov DL, 0ah*

*call WRITE\_MESSAGE\_BYTE*

*pop BX*

*pop AX*

*pop DX*

*ret*

*PRINT\_EXTENDED\_MEMORY ENDP*

*PRINT\_MCB\_TABLE PROC near*

*push DX*

*push AX*

*push BX*

*push CX*

*push SI*

*mov DX, offset MCB\_TABLE*

*call WRITE\_MESSAGE\_WORD*

*mov AH, 52h ; в AH список списков MCB таблицы*

*int 21h*

*mov AX, ES:[BX-2] ; адрес начала таблицы*

*mov ES, AX*

*lp3:*

```
mov AX, ES  
mov DI, offset TABLE_ADDRESS  
add DI, 12  
call WRD_TO_HEX  
mov DX, offset TABLE_ADDRESS  
call WRITE_MESSAGE_WORD ;вывод адреса
```

```
mov AX, ES:[1]  
mov DI, offset PSP_ADDRESS  
add DI, 16  
call WRD_TO_HEX  
mov DX, offset PSP_ADDRESS  
call WRITE_MESSAGE_WORD ;вывод PSP-адреса
```

```
mov DX, offset AREA_SIZE  
call WRITE_MESSAGE_WORD ;вывод размера участка  
mov AX, ES:[3]  
mov DI, 11  
mov BX, 16  
mul BX  
call TO_DEC
```

```
mov DX, offset SPACE  
call WRITE_MESSAGE_WORD
```

```
mov DX, offset SC_SD  
call WRITE_MESSAGE_WORD
```

```
mov BX, 8 ; для вывода последних 8 байт таблицы
```

*mov CX, 7*

*lp4:*

*mov DL, ES:[BX]*

*call WRITE\_MESSAGE\_BYTE*

*inc BX*

*loop lp4 ; вывод последних 8 байт (индексы в списке у них 1-*

*15)*

*mov DL, 0dh*

*call WRITE\_MESSAGE\_BYTE*

*mov DL, 0ah*

*call WRITE\_MESSAGE\_BYTE*

*mov AL, ES:[0h]*

*str AL, 5ah ; если в типе списка MCB записано 5ah - он последний  
(в других случаях 4dh)*

*je finish*

*mov AX, ES*

*inc AX*

*mov BX, ES:[3h]*

*add AX, BX*

*mov ES, AX ;переход к другому списку (по смещению размер  
текущего списка + 1)*

*jmp lp3*

*finish:*

*pop SI*

*pop CX*

```
    pop BX
    pop AX
    pop DX
    ret
PRINT_MCB_TABLE ENDP

BEGIN:
    call PRINT_AVAILABLE_MEMORY
    call PRINT_EXTENDED_MEMORY
    call PRINT_MCB_TABLE

    xor AL, AL
    mov AH, 4ch
    int 21h

TESTPC ENDS
END START
```



Файл com\_2.asm:

*TESTPC     SEGMENT*

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

*ORG 100H*

*START:     JMP BEGIN*

*AVAILABLE\_MEMORY db 'Available memory: \$'*

*EXTENDED\_MEMORY db 'Extended memory: \$'*

*MCB\_TABLE db 'MCB table:', 0DH, 0AH, '\$'*

*TABLE\_ADDRESS db 'Address:     \$'*

*SPACE db ' \$'*

*PSP\_ADDRESS db 'PSP address:     \$'*

*AREA\_SIZE db 'Area size: \$'*

*SC\_SD db 'SC/SD: \$'*

*TETR\_TO\_HEX PROC near*

*and AL, 0Fh*

*cmp AL, 09*

*jbe NEXT*

*add AL, 07*

*NEXT:     add AL, 30h*

*ret*

*TETR\_TO\_HEX ENDP*

*BYTE\_TO\_HEX   PROC near*

*push CX*

*mov AH, AL*

*call TETR\_TO\_HEX*

*xchg AL, AH*

*mov CL, 4*

```
    shr AL, CL
    call TETR_TO_HEX
    pop CX
    ret
BYTE_TO_HEX ENDP
```

```
WRD_TO_HEX PROC near
    push BX
    mov BH, AH
    call BYTE_TO_HEX
    mov [DI], AH
    dec DI
    mov [DI], AL
    dec DI
    mov AL, BH
    call BYTE_TO_HEX
    mov [DI], AH
    dec DI
    mov [DI], AL
    pop BX
    ret
WRD_TO_HEX ENDP
```

```
BYTE_TO_DEC PROC near
    push CX
    push DX
    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

```

```

WRITE_MESSAGE_BYTE PROC near
push AX
mov AH, 02h
int 21h
pop AX
ret
WRITE_MESSAGE_BYTE ENDP

```

```

WRITE_MESSAGE_WORD PROC near ;вывести строку
push AX
mov AH, 09h
int 21h
pop AX
ret
WRITE_MESSAGE_WORD ENDP

```

*TO\_DEC PROC near ; доступная память в десятичном виде*

*push BX*

*push CX*

*push AX*

*push DX*

*push DI*

*xor CX, CX*

*mov BX, 10*

*lp1:*

*div BX*

*push DX*

*xor DX, DX*

*inc CX*

*cmp AX, 0*

*jne lp1 ;получение остатка числа при делении на 10 пока не  
останется 0, все заносится в стек*

*lp2:*

*pop DX*

*add DL, '0' ;*

*call WRITE\_MESSAGE\_BYTE*

*loop lp2 ;извлечение из стека чисел, перевод в символы и вывод*

*pop DI*

*pop DX*

*pop AX*

*pop CX*

*pop BX*  
*ret*  
*TO\_DEC ENDP*

*PRINT\_AVAILABLE\_MEMORY PROC near ;вывести свободную память*  
*push DX*  
*push AX*  
*push BX*

*mov DX, offset AVAILABLE\_MEMORY*  
*call WRITE\_MESSAGE\_WORD*

*mov AH, 4ah ;освободить неиспользованную память, в BX заносится*  
*доступная память под программу*  
*mov BX, 0ffffh*  
*int 21h*

*mov AX, BX ;в AX заносится доступная память под программу (в*  
*параграфах)*  
*mov BX, 16*  
*mul BX ;в BX заносится 16, в AX заносится результат перемножения*  
*содержимых AX\*BX (перевод параграфов в байты)*

*call TO\_DEC*

*mov DL, 0dh*  
*call WRITE\_MESSAGE\_BYTE*  
*mov DL, 0ah*  
*call WRITE\_MESSAGE\_BYTE*

*pop BX*

*pop AX*

*pop DX*

*ret*

*PRINT\_AVAILABLE\_MEMORY ENDP*

*PRINT\_EXTENDED\_MEMORY PROC near*

*push DX*

*push AX*

*push BX*

*mov DX, offset EXTENDED\_MEMORY*

*call WRITE\_MESSAGE\_WORD*

*mov AL, 30h ; запись адреса ячейки*

*out 70h, AL ; 30h в выходной порт 70h*

*in AL, 71h ; чтение младшего байта размера расширенной памяти (из входного порта в AL)*

*mov BL, AL ; в BL младший байт*

*mov AL, 31h*

*out 70h, AL*

*in AL, 71h ; чтение старшего байта размера расширенной памяти*

*mov BH, AL ; в BH старший байт*

*mov AX, BX*

*mov BX, 16*

*mul BX*

*call TO\_DEC*

```

mov DL, 0dh
call WRITE_MESSAGE_BYTE
mov DL, 0ah
call WRITE_MESSAGE_BYTE

pop BX
pop AX
pop DX
ret
PRINT_EXTENDED_MEMORY ENDP

```

```

PRINT_MCB_TABLE PROC near
push DX
push AX
push BX
push CX
push SI

```

```

mov DX, offset MCB_TABLE
call WRITE_MESSAGE_WORD

```

```

mov AH, 52h ; в AH список списков MCB таблицы
int 21h
mov AX, ES:[BX-2] ; адрес начала таблицы
mov ES, AX

```

```

lp3:
    mov AX, ES
    mov DI, offset TABLE_ADDRESS

```

```

add DI, 12
call WRD_TO_HEX
mov DX, offset TABLE_ADDRESS
call WRITE_MESSAGE_WORD ;вывод адреса

mov AX, ES:[1]
mov DI, offset PSP_ADDRESS
add DI, 16
call WRD_TO_HEX
mov DX, offset PSP_ADDRESS
call WRITE_MESSAGE_WORD ;вывод PSP-адреса

mov DX, offset AREA_SIZE
call WRITE_MESSAGE_WORD ;вывод размера участка
mov AX, ES:[3]
mov DI, 11
mov BX, 16
mul BX
call TO_DEC

mov DX, offset SPACE
call WRITE_MESSAGE_WORD

mov DX, offset SC_SD
call WRITE_MESSAGE_WORD

mov BX, 8 ; для вывода последних 8 байт таблицы
mov CX, 7

lp4:

```



*mov DL, ES:[BX]*  
*call WRITE\_MESSAGE\_BYTE*  
*inc BX*  
*loop lp4 ; вывод последних 8 байт (индексы в списке у них 1-*  
15)

*mov DL, 0dh*  
*call WRITE\_MESSAGE\_BYTE*  
*mov DL, 0ah*  
*call WRITE\_MESSAGE\_BYTE*

*mov AL, ES:[0h]*  
*cmp AL, 5ah ; если в типе списка MCB записано 5ah - он последний*  
(в других случаях 4dh)  
*je finish*

*mov AX, ES*  
*inc AX*  
*mov BX, ES:[3h]*  
*add AX, BX*  
*mov ES, AX ;переход к другому списку (по смещению размер*  
текущего списка + 1)  
*jmp lp3*

*finish:*

*pop SI*  
*pop CX*  
*pop BX*  
*pop AX*  
*pop DX*

```

        ret

PRINT_MCB_TABLE ENDP

FREE PROC near
    push AX
    push BX
    push DX

    xor DX, DX
    lea AX, FINISH_CODE ;текущий адрес конца программы в AX
    mov BX, 10h
    div BX
    add AX, DX
    mov BX, AX ;В BX занятый размер памяти
    xor AX, AX

    mov AH, 4ah
    int 21h ;высвобождение неиспользованной памяти

    pop DX
    pop BX
    pop AX
    ret
FREE ENDP

BEGIN:
    call PRINT_AVAILABLE_MEMORY
    call PRINT_EXTENDED_MEMORY
    call FREE
    call PRINT_MCB_TABLE

```

```
xor AL, AL  
mov AH, 4ch  
int 21h
```

```
FINISH_CODE:  
TESTPC ENDS  
END START
```

файл com\_3.asm:

*TESTPC     SEGMENT*

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

*ORG 100H*

*START:     JMP BEGIN*

*AVAILABLE\_MEMORY db 'Available memory: \$'*

*EXTENDED\_MEMORY db 'Extended memory: \$'*

*MCB\_TABLE db 'MCB table:', 0DH, 0AH, '\$'*

*TABLE\_ADDRESS db 'Address:     \$'*

*SPACE db ' \$'*

*PSP\_ADDRESS db 'PSP address:     \$'*

*AREA\_SIZE db 'Area size: \$'*

*SC\_SD db 'SC/SD: \$'*

*TETR\_TO\_HEX PROC near*

*and AL, 0Fh*

*cmp AL, 09*

*jbe NEXT*

*add AL, 07*

*NEXT:     add AL, 30h*

*ret*

*TETR\_TO\_HEX ENDP*

*BYTE\_TO\_HEX   PROC near*

*push CX*

*mov AH, AL*

*call TETR\_TO\_HEX*

*xchg AL, AH*

*mov CL, 4*

```
    shr AL, CL
    call TETR_TO_HEX
    pop CX
    ret
BYTE_TO_HEX ENDP
```

```
WRD_TO_HEX PROC near
    push BX
    mov BH, AH
    call BYTE_TO_HEX
    mov [DI], AH
    dec DI
    mov [DI], AL
    dec DI
    mov AL, BH
    call BYTE_TO_HEX
    mov [DI], AH
    dec DI
    mov [DI], AL
    pop BX
    ret
WRD_TO_HEX ENDP
```

```
BYTE_TO_DEC PROC near
    push CX
    push DX
    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

```

```

WRITE_MESSAGE_BYTE PROC near
push AX
mov AH, 02h
int 21h
pop AX
ret
WRITE_MESSAGE_BYTE ENDP

```

```

WRITE_MESSAGE_WORD PROC near ;вывести строку
push AX
mov AH, 09h
int 21h
pop AX
ret
WRITE_MESSAGE_WORD ENDP

```

*TO\_DEC PROC near ; доступная память в десятичном виде*

*push BX*

*push CX*

*push AX*

*push DX*

*push DI*

*xor CX, CX*

*mov BX, 10*

*lp1:*

*div BX*

*push DX*

*xor DX, DX*

*inc CX*

*cmp AX, 0*

*jne lp1 ;получение остатка числа при делении на 10 пока не  
останется 0, все заносится в стек*

*lp2:*

*pop DX*

*add DL, '0' ;*

*call WRITE\_MESSAGE\_BYTE*

*loop lp2 ;извлечение из стека чисел, перевод в символы и вывод*

*pop DI*

*pop DX*

*pop AX*

*pop CX*

*pop BX*

*ret*

*TO\_DEC ENDP*

*PRINT\_AVAILABLE\_MEMORY PROC near ;вывести свободную память*

*push DX*

*push AX*

*push BX*

*mov DX, offset AVAILABLE\_MEMORY*

*call WRITE\_MESSAGE\_WORD*

*mov AH, 4ah ;освободить неиспользованную память, в BX заносится  
доступная память под программу*

*mov BX, 0ffffh*

*int 21h*

*mov AX, BX ;в AX заносится доступная память под программу (в  
параграфах)*

*mov BX, 16*

*mul BX ;в BX заносится 16, в AX заносится результат перемножения  
содержимых AX\*BX (перевод параграфов в байты)*

*call TO\_DEC*

*mov DL, 0dh*

*call WRITE\_MESSAGE\_BYTE*

*mov DL, 0ah*

*call WRITE\_MESSAGE\_BYTE*



*pop BX*

*pop AX*

*pop DX*

*ret*

*PRINT\_AVAILABLE\_MEMORY ENDP*

*PRINT\_EXTENDED\_MEMORY PROC near*

*push DX*

*push AX*

*push BX*

*mov DX, offset EXTENDED\_MEMORY*

*call WRITE\_MESSAGE\_WORD*

*mov AL, 30h ; запись адреса ячейки*

*out 70h, AL ; 30h в выходной порт 70h*

*in AL, 71h ; чтение младшего байта размера расширенной памяти (из входного порта в AL)*

*mov BL, AL ; в BL младший байт*

*mov AL, 31h*

*out 70h, AL*

*in AL, 71h ; чтение старшего байта размера расширенной памяти*

*mov BH, AL ; в BH старший байт*

*mov AX, BX*

*mov BX, 16*

*mul BX*

*call TO\_DEC*

```

mov DL, 0dh
call WRITE_MESSAGE_BYTE
mov DL, 0ah
call WRITE_MESSAGE_BYTE

pop BX
pop AX
pop DX
ret
PRINT_EXTENDED_MEMORY ENDP

```

```

PRINT_MCB_TABLE PROC near
push DX
push AX
push BX
push CX
push SI

```

```

mov DX, offset MCB_TABLE
call WRITE_MESSAGE_WORD

```

```

mov AH, 52h ; в AH список списков MCB таблицы
int 21h
mov AX, ES:[BX-2] ; адрес начала таблицы
mov ES, AX

```

```

lp3:
    mov AX, ES
    mov DI, offset TABLE_ADDRESS

```

```

add DI, 12
call WRD_TO_HEX
mov DX, offset TABLE_ADDRESS
call WRITE_MESSAGE_WORD ;вывод адреса

mov AX, ES:[1]
mov DI, offset PSP_ADDRESS
add DI, 16
call WRD_TO_HEX
mov DX, offset PSP_ADDRESS
call WRITE_MESSAGE_WORD ;вывод PSP-адреса

mov DX, offset AREA_SIZE
call WRITE_MESSAGE_WORD ;вывод размера участка
mov AX, ES:[3]
mov DI, 11
mov BX, 16
mul BX
call TO_DEC

mov DX, offset SPACE
call WRITE_MESSAGE_WORD

mov DX, offset SC_SD
call WRITE_MESSAGE_WORD

mov BX, 8 ; для вывода последних 8 байт таблицы
mov CX, 7

lp4:

```

*mov DL, ES:[BX]*  
*call WRITE\_MESSAGE\_BYTE*  
*inc BX*  
*loop lp4 ; вывод последних 8 байт (индексы в списке у них 1-*  
15)

*mov DL, 0dh*  
*call WRITE\_MESSAGE\_BYTE*  
*mov DL, 0ah*  
*call WRITE\_MESSAGE\_BYTE*

*mov AL, ES:[0h]*  
*cmp AL, 5ah ; если в типе списка MCB записано 5ah - он последний*  
*(в других случаях 4dh)*  
*je finish*

*mov AX, ES*  
*inc AX*  
*mov BX, ES:[3h]*  
*add AX, BX*  
*mov ES, AX ;переход к другому списку (по смещению размер*  
*текущего списка + 1)*  
*jmp lp3*

*finish:*

*pop SI*  
*pop CX*  
*pop BX*  
*pop AX*  
*pop DX*

```

        ret

PRINT_MCB_TABLE ENDP


FREE PROC near
    push AX
    push BX
    push DX

    xor DX, DX
    lea AX, FINISH_CODE ;текущий адрес конца программы в AX
    mov BX, 10h
    div BX
    add AX, DX
    mov BX, AX ;В BX занятый размер памяти
    xor AX, AX

    mov AH, 4ah
    int 21h ;высвобождение неиспользованной памяти

    pop DX
    pop BX
    pop AX
    ret
FREE ENDP


REQUEST_MEMORY PROC near
    push AX
    push BX

    mov     BX, 1000h

```

```

    mov    AH, 48h
    int    21h ;запрос на 1000 кб памяти

pop BX
pop AX
ret
REQUEST_MEMORY ENDP

BEGIN:
call PRINT_AVAILABLE_MEMORY
call PRINT_EXTENDED_MEMORY
call FREE
call REQUEST_MEMORY
call PRINT_MCB_TABLE

xor AL, AL
mov AH, 4ch
int 21h

FINISH_CODE:
TESTPC ENDS
END START

```

файл com\_4.asm:

*TESTPC     SEGMENT*

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

*ORG 100H*

*START:     JMP BEGIN*

*AVAILABLE\_MEMORY db 'Available memory: \$'*

*EXTENDED\_MEMORY db 'Extended memory: \$'*

*MCB\_TABLE db 'MCB table:', 0DH, 0AH, '\$'*

*TABLE\_ADDRESS db 'Address:     \$'*

*SPACE db ' \$'*

*PSP\_ADDRESS db 'PSP address:     \$'*

*AREA\_SIZE db 'Area size: \$'*

*SC\_SD db 'SC/SD: \$'*

*ERROR\_MEMORY\_REQUEST db 'Error... memory could not be allocated :(\$'*

*TETR\_TO\_HEX PROC near*

*and AL, 0Fh*

*cmp AL, 09*

*jbe NEXT*

*add AL, 07*

*NEXT:     add AL, 30h*

*ret*

*TETR\_TO\_HEX ENDP*

*BYTE\_TO\_HEX   PROC near*

*push CX*

*mov AH, AL*

*call TETR\_TO\_HEX*

*xchg AL, AH*

```

    mov CL, 4
    shr AL, CL
    call TETR_TO_HEX
    pop CX
    ret
BYTE_TO_HEX ENDP

```

```

WRD_TO_HEX PROC near
    push BX
    mov BH, AH
    call BYTE_TO_HEX
    mov [DI], AH
    dec DI
    mov [DI], AL
    dec DI
    mov AL, BH
    call BYTE_TO_HEX
    mov [DI], AH
    dec DI
    mov [DI], AL
    pop BX
    ret
WRD_TO_HEX ENDP

```

```

BYTE_TO_DEC PROC near
    push CX
    push DX
    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

```

```

WRITE_MESSAGE_BYTE PROC near
push AX
mov AH, 02h
int 21h
pop AX
ret
WRITE_MESSAGE_BYTE ENDP

```

```

WRITE_MESSAGE_WORD PROC near ;вывести строку
push AX
mov AH, 09h
int 21h
pop AX
ret

```

*WRITE\_MESSAGE\_WORD ENDP*

*TO\_DEC PROC near ; доступная память в десятичном виде*

*push BX*

*push CX*

*push AX*

*push DX*

*push DI*

*xor CX, CX*

*mov BX, 10*

*lp1:*

*div BX*

*push DX*

*xor DX, DX*

*inc CX*

*cmp AX, 0*

*jne lp1 ;получение остатка числа при делении на 10 пока не  
останется 0, все заносится в стек*

*lp2:*

*pop DX*

*add DL, '0' ;*

*call WRITE\_MESSAGE\_BYTE*

*loop lp2 ;извлечение из стека чисел, перевод в символы и вывод*

*pop DI*

*pop DX*

*pop AX*

*pop CX*  
*pop BX*  
*ret*  
*TO\_DEC ENDP*

*PRINT\_AVAILABLE\_MEMORY PROC near ;вывести свободную память*

*push DX*  
*push AX*  
*push BX*

*mov DX, offset AVAILABLE\_MEMORY*  
*call WRITE\_MESSAGE\_WORD*

*mov AH, 4ah ;освободить неиспользованную память, в BX заносится*  
*доступная память под программу*

*mov BX, 0ffffh*  
*int 21h*

*mov AX, BX ;в AX заносится доступная память под программу (в*  
*параграфах)*

*mov BX, 16*

*mul BX ;в BX заносится 16, в AX заносится результат перемножения*  
*содержимых AX\*BX (перевод параграфов в байты)*

*call TO\_DEC*

*mov DL, 0dh*

*call WRITE\_MESSAGE\_BYTE*

*mov DL, 0ah*

*call WRITE\_MESSAGE\_BYTE*

*pop BX*

*pop AX*

*pop DX*

*ret*

*PRINT\_AVAILABLE\_MEMORY ENDP*

*PRINT\_EXTENDED\_MEMORY PROC near*

*push DX*

*push AX*

*push BX*

*mov DX, offset EXTENDED\_MEMORY*

*call WRITE\_MESSAGE\_WORD*

*mov AL, 30h ; запись адреса ячейки*

*out 70h, AL ; 30h в выходной порт 70h*

*in AL, 71h ; чтение младшего байта размера расширенной памяти (из входного порта в AL)*

*mov BL, AL ; в BL младший байт*

*mov AL, 31h*

*out 70h, AL*

*in AL, 71h ; чтение старшего байта размера расширенной памяти*

*mov BH, AL ; в BH старший байт*

*mov AX, BX*

*mov BX, 16*

*mul BX*

*call TO\_DEC*

*mov DL, 0dh*

*call WRITE\_MESSAGE\_BYTE*

*mov DL, 0ah*

*call WRITE\_MESSAGE\_BYTE*

*pop BX*

*pop AX*

*pop DX*

*ret*

*PRINT\_EXTENDED\_MEMORY ENDP*

*PRINT\_MCB\_TABLE PROC near*

*push DX*

*push AX*

*push BX*

*push CX*

*push SI*

*mov DX, offset MCB\_TABLE*

*call WRITE\_MESSAGE\_WORD*

*mov AH, 52h ; в AH список списков MCB таблицы*

*int 21h*

*mov AX, ES:[BX-2] ; адрес начала таблицы*

*mov ES, AX*

*lp3:*

*mov AX, ES*

```

mov DI, offset TABLE_ADDRESS
add DI, 12
call WRD_TO_HEX
mov DX, offset TABLE_ADDRESS
call WRITE_MESSAGE_WORD ;вывод адреса

mov AX, ES:[1]
mov DI, offset PSP_ADDRESS
add DI, 16
call WRD_TO_HEX
mov DX, offset PSP_ADDRESS
call WRITE_MESSAGE_WORD ;вывод PSP-адреса

mov DX, offset AREA_SIZE
call WRITE_MESSAGE_WORD ;вывод размера участка
mov AX, ES:[3]
mov DI, 11
mov BX, 16
mul BX
call TO_DEC

mov DX, offset SPACE
call WRITE_MESSAGE_WORD

mov DX, offset SC_SD
call WRITE_MESSAGE_WORD

mov BX, 8 ; для вывода последних 8 байт таблицы
mov CX, 7

```

*lp4:*

*mov DL, ES:[BX]*

*call WRITE\_MESSAGE\_BYTE*

*inc BX*

*loop lp4 ; вывод последних 8 байт (индексы в списке у них 1-*

*15)*

*mov DL, 0dh*

*call WRITE\_MESSAGE\_BYTE*

*mov DL, 0ah*

*call WRITE\_MESSAGE\_BYTE*

*mov AL, ES:[0h]*

*стр AL, 5ah ; если в типе списка MCB записано 5ah - он последний  
(в других случаях 4dh)*

*je finish*

*mov AX, ES*

*inc AX*

*mov BX, ES:[3h]*

*add AX, BX*

*mov ES, AX ;переход к другому списку (по смещению размер  
текущего списка + 1)*

*jmp lp3*

*finish:*

*pop SI*

*pop CX*

*pop BX*

*pop AX*

```
    pop DX
    ret
PRINT_MCB_TABLE ENDP
```

```
FREE PROC near
```

```
    push AX
```

```
    push BX
```

```
    push DX
```

```
    xor DX, DX
```

```
    lea AX, FINISH_CODE ;текущий адрес конца программы в AX
```

```
    mov BX, 10h
```

```
    div BX
```

```
    add AX, DX
```

```
    mov BX, AX ;В BX занятый размер памяти
```

```
    xor AX, AX
```

```
    mov AH, 4ah
```

```
    int 21h ;высвобождение неиспользованной памяти
```

```
    pop DX
```

```
    pop BX
```

```
    pop AX
```

```
    ret
```

```
FREE ENDP
```

```
REQUEST_MEMORY PROC near
```

```
    push AX
```

```
    push BX
```

```
    PUSH DX
```



```
mov    BX, 1000h
      mov    AH, 48h
      int    21h ;запрос на 1000 кб памяти
```

```
jb very_sad ; если cf = 1
jmp very_good
```

```
very_sad:
mov DX, offset ERROR_MEMORY_REQUEST
call WRITE_MESSAGE_WORD
mov DL, 0dh
call WRITE_MESSAGE_BYTE
mov DL, 0ah
call WRITE_MESSAGE_BYTE
```

```
very_good:
      pop DX
      pop BX
      pop AX
ret
REQUEST_MEMORY ENDP
```

```
BEGIN:
call PRINT_AVAILABLE_MEMORY
call PRINT_EXTENDED_MEMORY
call REQUEST_MEMORY
call FREE
call PRINT_MCB_TABLE
```

*xor AL, AL*

*mov AH, 4ch*

*int 21h*

*FINISH\_CODE:*

*TESTPC ENDS*

*END START*