

МСВМИНОБРНАУКИ 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 интерфейсов программных модулей

Студент гр. 9383

Нистратов Д.Г.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Постановка задачи.

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.

Функции и структуры данных

Таблица 1 – Функции и структуры данных

Название функций	Описание функций
TETR_TO_HEX	Перевод из четверичной с/с в шестнадцатеричную с/с
BYTE_TO_HEX	Перевод из двоичной с/с в шестнадцатеричную с/с
WRD_TO_HEX	Перевод 2 байтов в шестнадцатеричную с/с
BYTE_TO_DEC	Перевод из двоичной с/с в десятичную с/с
WRD_TO_DEC	Перевод в 10 с/с 10ти разрядного числа
WRITE	Вывод строки на экран
PRINT_AVAILABLE_MEMORY	Вывод доступной памяти
PRINT_EXTENDED_MEMORY	Вывод расширенной памяти
MCB_TYPE	Вывод тип MCB

MCB_SIZE	Вывод размера MCB
SC_SD	Вывод SD или SC
PRINT_MCB_DATA	Вывод таблицы MCB
PSP_ADDRESS	Вывод адреса PSP

Последовательность действий

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

Сохраните результаты, полученные программой, и включите их в отчет.

Шаг 2. Измените программу, чтобы она освобождала память.

Сохраните результаты, полученные программой, и включите их в отчет.

Шаг 3. Измените программу, чтобы после освобождения памяти выделялось 64Кб памяти.

Сохраните результаты, полученные программой, и включите их в отчет.

Шаг 4. Измените первоначальный вариант программы, запросив 64Кб памяти.

Сохраните результаты, полученные программой, и включите их в отчет.

Выполнение работы.

Шаг 1. Был написан модуль типа .COM, распечатывающий информацию о количестве доступной памяти, размер расширенной памяти и цепочку блоков управления памятью. Результат работы программы см. Изображение 1

```
D:\LETI\OS\MASM>LAB3_1.COM
Available memory: 648912
Extended memory: 15360
Table:
4D          0008          16
4D          0000          64
4D          0040         256
4D          0192         144
5A          0192      648912      LAB3_1
```

Изображение 1

Шаг 2. Программа была изменена для освобождения памяти, которую она не занимает. Результат работы программы см. Изображение 2

```
D:\LET\OS\MASM>LAB3_2.COM
Available memory: 648912
Extended memory: 15360
Table:
4D      0008      16
4D      0000      64
4D      0040     256
4D      0192     144
4D      0192     848
5A      0000     648048
LAB3_2
ì>?  0 0
```

Изображение 2

Шаг3. Программа была изменена для выделения 64Кб памяти. Результат работы программы см. Изображение 3

```
D:\LETINOS\MASM>LAB3_3.COM
Available memory: 648912
Extended memory: 15360
Table:
    4D      0008      16
    4D      0000      64
    4D      0040     256
    4D      0192     144
    4D      0192     848    LAB3_3
    4D      0192    65536    LAB3_3
    5A      0000    582496    crossoft
```

Изображение 3

Шаг 4. Первоначальная программа была изменена для запроса на выделение 64Кб памяти. Результат работы программы см. Изображение 4

```
D:\LETI\OS\MASM>LAB3_4.COM
Available memory: 648912
Extended memory: 15360
Memory allocation error!
Table:
    4D      0008      16
    4D      0000      64
    4D      0040     256
    4D      0192     144
    5A      0192   648912   LAB3_4
```

Изображение 4

Ответы на вопросы:

1. Что означает «доступный объём памяти»?

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

2. Где МСВ блок Вашей программы в списке?

МСВ блок на первом, втором и четвертом шаге находится на 5 строке, в третьем шаге на 5 и 6 строке, т.к был запрошен дополнительный блок памяти.

3.Какой размер памяти занимает программа в каждом случае?

В первом, программа занимает весь доступный объём памяти.

Во втором, программа занимает только необходимый объём памяти, 848 Байта.

В третьем, программа занимает необходимый объём памяти, 848 Байта, и запрошенные 64Кб памяти.

В четвертом, программа занимает 144 байта, а запрошенные 64Кб небыли выделены.

Заключение.

В ходе лабораторной работы были исследованы структуры данных и работа функций управления памяти операционной системы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3_1.asm

; Исследование организации управления основной памятью

; 25.03.2021

; Nistratov Dmitry

; 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

```
AVAILABLE_MEM db 'Available memory: ', 10, 13, '$'
```

```
EXTENDED_MEM      db 'Extended memory:      ', 10, 13, '$'
```

TABLE db 'Table: ', 10, 13, '\$'

TABLE_DATA db ' ', 10, 13, '\$';

67

; 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 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 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

;

; Перевод в 10 с/с 10-ти разрядного числа

; в AX - число, DI - адрес последнего символа

WRD_TO_DEC PROC NEAR

push CX

push DX


```

    mov CX,10
loop_b: div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp AX,10
    jae loop_b
    cmp AL,00h
    je endl
    or AL,30h
    mov [SI],AL
endl:  pop DX
    pop CX
    ret
WRD_TO_DEC ENDP
;_____
; КОД
WRITE PROC NEAR
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE ENDP

WRITEBYTE PROC NEAR
    push ax
    mov ah, 02h
    int 21h

```

```

    pop ax
    ret
WRITEBYTE ENDP

EMPTY_MEMORY PROC NEAR
    push ax
    push bx
    push dx

    pop ax
    pop bx
    pop dx
    ret
EMPTY_MEMORY ENDP

PRINT_AVAILABLE_MEMORY PROC NEAR
    push ax
    push bx
    push dx
    push si

    xor ax, ax
    mov ah, 4Ah
    mov bx, 0FFFFh
    int 21h
    mov ax, 10h
    mul bx

    mov si, offset AVAILABLE_MEM
    add si, 23

```

```
call WRD_TO_DEC
mov dx, offset AVAILABLE_MEM
call WRITE
```

```
pop si
pop dx
pop bx
pop ax
ret
```

```
PRINT_AVAILABLE_MEMORY ENDP
```

```
PRINT_EXTENDED_MEMORY PROC NEAR
```

```
push ax
push bx
push dx
push si
```

```
xor dx, dx
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 al, bl
```

```
mov si, offset EXTENDED_MEM
add si, 21
```

```

call WRD_TO_DEC
mov dx, offset EXTENDED_MEM
call WRITE

pop si
pop dx
pop bx
pop ax
ret
PRINT_EXTENDED_MEMORY ENDP

; MCB type (1 Byte) 4Dh / 5Ah
MCB_TYPE PROC NEAR
    push ax
    push di

    mov di, offset TABLE_DATA
    add di, 5
    xor ah, ah
    mov al, es:[00h]
    call BYTE_TO_HEX
    mov [di], al
    inc di
    mov [di], ah

    pop di
    pop ax
    ret
MCB_TYPE ENDP

```

; PSP Address (2Byte)

PSP_ADDRESS PROC NEAR

push ax

push di

mov di, offset TABLE_DATA

mov ax, es:[01h]

add di, 19

call WRD_TO_HEX

pop di

pop ax

ret

PSP_ADDRESS ENDP

; Size (2 Byte)

MCB_SIZE PROC NEAR

push ax

push bx

push di

push si

mov di, offset TABLE_DATA

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 29

mov si, di

call WRD_TO_DEC

```

    pop si
    pop di
    pop bx
    pop ax
    ret
MCB_SIZE ENDP

; SC or SD (8 Byte)
SC_SD PROC NEAR
    push bx
    push dx
    push di

    mov di, offset TABLE_DATA
    add di, 33
    mov bx, 0h
JMP_NEXT:
    mov dl, es:[bx + 8]
    mov [di], dl
    inc di
    inc bx
    cmp bx, 8h
    jne JMP_NEXT

    pop di
    pop dx
    pop bx
    ret
SC_SD ENDP

```

PRINT_MCB_DATA PROC NEAR

push ax

push bx

push es

push dx

mov ah, 52h

int 21h

mov es, es:[bx - 2h]

NEXT_MCB:

call MCB_TYPE

call PSP_ADDRESS

call MCB_SIZE

call SC_SD

mov ax, es:[03h]

mov bl, es:[00h]

mov dx, offset TABLE_DATA

call WRITE

mov cx, es

add ax, cx

inc ax

mov es, ax

cmp bl, 4Dh ; if not last (5Ah)

je NEXT_MCB

pop dx

pop es

pop bx

```

    pop ax

    ret

PRINT_MCB_DATA ENDP

BEGIN:
; Код
    call PRINT_AVAILABLE_MEMORY
    call PRINT_EXTENDED_MEMORY
    mov dx, offset TABLE
    call WRITE
    call PRINT_MCB_DATA
; Выход в DOS
    xor AL,AL
    mov AH,4Ch
    int 21H
TESTPC ENDS

    END START ; Конец модуля, START - точка входа

```

Название файла: lab3_2.asm

; Исследование организации управления основной памятью

; 25.03.2021

; Nistratov Dmitry

; 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


```

AVAILABLE_MEM      db 'Available memory:      ', 10, 13, '$'
EXTENDED_MEM       db 'Extended memory:       ', 10, 13, '$'
TABLE              db 'Table: ', 10, 13, '$'
TABLE_DATA         db '                        ', 10, 13, '$';

```

67

; 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 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 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

;

; Перевод в 10 с/с 10-ти разрядного числа

; в AX - число, DI - адрес последнего символа

WRD_TO_DEC PROC NEAR

```

push CX
push DX
mov CX,10
loop_b: div CX
or DL,30h
mov [SI],DL
dec SI
xor DX,DX
cmp AX,10
jae loop_b
cmp AL,00h
je endl
or AL,30h
mov [SI],AL
endl: pop DX

```

```

        pop CX
        ret
WRD_TO_DEC ENDP

;_____
; КОД
WRITE PROC NEAR
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE ENDP

WRITEBYTE PROC NEAR
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
WRITEBYTE ENDP

EMPTY_MEMORY PROC NEAR
    push ax
    push bx
    push dx

    pop ax
    pop bx
    pop dx
    ret

```

EMPTY_MEMORY ENDP

PRINT_AVAILABLE_MEMORY PROC NEAR

push ax

push bx

push dx

push si

xor ax, ax

mov ah, 4Ah

mov bx, 0FFFFh

int 21h

mov ax, 10h

mul bx

mov si, offset AVAILABLE_MEM

add si, 23

call WRD_TO_DEC

mov dx, offset AVAILABLE_MEM

call WRITE

pop si

pop dx

pop bx

pop ax

ret

PRINT_AVAILABLE_MEMORY ENDP

PRINT_EXTENDED_MEMORY PROC NEAR

push ax

push bx

push dx

push si

xor dx, dx

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 al, bl

mov si, offset EXTENDED_MEM

add si, 21

call WRD_TO_DEC

mov dx, offset EXTENDED_MEM

call WRITE

pop si

pop dx

pop bx

pop ax

ret

PRINT_EXTENDED_MEMORY ENDP

; MCB type (1 Byte) 4Dh / 5Ah

MCB_TYPE PROC NEAR

```

push ax
push di

mov di, offset TABLE_DATA
add di, 5
xor ah, ah
mov al, es:[00h]
call BYTE_TO_HEX
mov [di], al
inc di
mov [di], ah

pop di
pop ax
ret

```

MCB_TYPE ENDP

; PSP Address (2Byte)

PSP_ADDRESS PROC NEAR

```

push ax
push di

mov di, offset TABLE_DATA
mov ax, es:[01h]
add di, 19
call WRD_TO_HEX

pop di
pop ax
ret

```

PSP_ADDRESS ENDP

; Size (2 Byte)

MCB_SIZE PROC NEAR

push ax

push bx

push di

push si

mov di, offset TABLE_DATA

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 29

mov si, di

call WRD_TO_DEC

pop si

pop di

pop bx

pop ax

ret

MCB_SIZE ENDP

; SC or SD (8 Byte)

SC_SD PROC NEAR

push bx

push dx

push di


```
mov di, offset TABLE_DATA
add di, 33
mov bx, 0h
JMP_NEXT:
mov dl, es:[bx + 8]
    mov [di], dl
    inc di
    inc bx
    cmp bx, 8h
jne JMP_NEXT
```

```
pop di
pop dx
pop bx
ret
```

SC_SD ENDP

PRINT_MCB_DATA PROC NEAR

```
push ax
push bx
push es
push dx
```

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

NEXT_MCB:

```
call MCB_TYPE
call PSP_ADDRESS
call MCB_SIZE
```

```
call SC_SD
mov ax, es:[03h]
mov bl, es:[00h]
mov dx, offset TABLE_DATA
call WRITE
```

```
mov cx, es
add ax, cx
inc ax
mov es, ax
```

```
cmp bl, 4Dh ; if not last (5Ah)
je NEXT_MCB
```

```
pop dx
pop es
pop bx
pop ax
```

```
ret
```

```
PRINT_MCB_DATA ENDP
```

```
FREE_UNUSED_MEMORY PROC
```

```
push ax
push bx
push cx
push dx
```

```
lea ax, END_POINT
mov bx, 10h
```

```
xor dx,dx
div bx
inc ax
mov bx,ax
mov al,0
mov ah,4Ah
int 21h
```

```
pop dx
pop cx
pop bx
pop ax
ret
```

FREE_UNUSED_MEMORY ENDP

BEGIN:

; Код

```
call PRINT_AVAILABLE_MEMORY
call PRINT_EXTENDED_MEMORY
mov dx, offset TABLE
call WRITE
call FREE_UNUSED_MEMORY
call PRINT_MCB_DATA
```

; Выход в DOS

```
xor AL,AL
mov AH,4Ch
int 21H
```

END_POINT:

TESTPC ENDS

END START ; Конец модуля, START - точка входа

Название файла: lab3_3.asm

; Исследование организации управления основной памятью

; 25.03.2021

; Nistratov Dmitry

; 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

AVAILABLE_MEM db 'Available memory: ', 10, 13, '\$'

EXTENDED_MEM db 'Extended memory: ', 10, 13, '\$'

TABLE db 'Table: ', 10, 13, '\$'

TABLE_DATA db ' ', 10, 13, '\$';

69

; 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 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 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

;

; Перевод в 10 с/с 10-ти разрядного числа

; в AX - число, DI - адрес последнего символа

WRD_TO_DEC PROC NEAR

push CX

push DX

mov CX,10

```

loop_b: div CX
        or DL,30h
        mov [SI],DL
        dec SI
        xor DX,DX
        cmp AX,10
        jae loop_b
        cmp AL,00h
        je endl
        or AL,30h
        mov [SI],AL
endl:   pop DX
        pop CX
        ret
WRD_TO_DEC ENDP

;_____
; КОД
WRITE PROC NEAR
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
WRITE ENDP

WRITEBYTE PROC NEAR
        push ax
        mov ah, 02h
        int 21h
        pop ax

```

```

    ret
WRITEBYTE ENDP

EMPTY_MEMORY PROC NEAR
    push ax
    push bx
    push dx

    pop ax
    pop bx
    pop dx
    ret
EMPTY_MEMORY ENDP

PRINT_AVAILABLE_MEMORY PROC NEAR
    push ax
    push bx
    push dx
    push si

    xor ax, ax
    mov ah, 4Ah
    mov bx, 0FFFFh
    int 21h
    mov ax, 10h
    mul bx

    mov si, offset AVAILABLE_MEM
    add si, 23
    call WRD_TO_DEC

```



```
mov dx, offset AVAILABLE_MEM  
call WRITE
```

```
pop si  
pop dx  
pop bx  
pop ax  
ret
```

```
PRINT_AVAILABLE_MEMORY ENDP
```

```
PRINT_EXTENDED_MEMORY PROC NEAR
```

```
push ax  
push bx  
push dx  
push si
```

```
xor dx, dx  
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 al, bl
```

```
mov si, offset EXTENDED_MEM  
add si, 21  
call WRD_TO_DEC
```

```
mov dx, offset EXTENDED_MEM
call WRITE
```

```
pop si
pop dx
pop bx
pop ax
ret
```

```
PRINT_EXTENDED_MEMORY ENDP
```

```
; MCB type (1 Byte) 4Dh / 5Ah
```

```
MCB_TYPE PROC NEAR
```

```
push ax
push di
```

```
mov di, offset TABLE_DATA
add di, 5
xor ah, ah
mov al, es:[00h]
call BYTE_TO_HEX
mov [di], al
inc di
mov [di], ah
```

```
pop di
pop ax
ret
```

```
MCB_TYPE ENDP
```

```
; PSP Address (2Byte)
```

PSP_ADDRESS PROC NEAR

push ax

push di

mov di, offset TABLE_DATA

mov ax, es:[01h]

add di, 19

call WRD_TO_HEX

pop di

pop ax

ret

PSP_ADDRESS ENDP

; Size (2 Byte)

MCB_SIZE PROC NEAR

push ax

push bx

push di

push si

mov di, offset TABLE_DATA

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 29

mov si, di

call WRD_TO_DEC

pop si

```
    pop di
    pop bx
    pop ax
    ret
MCB_SIZE ENDP
```

; SC or SD (8 Byte)

```
SC_SD PROC NEAR
```

```
    push bx
    push dx
    push di
```

```
    mov di, offset TABLE_DATA
```

```
    add di, 33
```

```
    mov bx, 0h
```

```
JMP_NEXT:
```

```
    mov dl, es:[bx + 8]
```

```
        mov [di], dl
```

```
        inc di
```

```
        inc bx
```

```
        cmp bx, 8h
```

```
    jne JMP_NEXT
```

```
    pop di
```

```
    pop dx
```

```
    pop bx
```

```
    ret
```

```
SC_SD ENDP
```

```
PRINT_MCB_DATA PROC NEAR
```

push ax

push bx

push es

push dx

mov ah, 52h

int 21h

mov es, es:[bx - 2h]

NEXT_MCB:

call MCB_TYPE

call PSP_ADDRESS

call MCB_SIZE

call SC_SD

mov ax, es:[03h]

mov bl, es:[00h]

mov dx, offset TABLE_DATA

call WRITE

mov cx, es

add ax, cx

inc ax

mov es, ax

cmp bl, 4Dh ; if not last (5Ah)

je NEXT_MCB

pop dx

pop es

pop bx

pop ax

```

        ret
PRINT_MCB_DATA ENDP

FREE_UNUSED_MEMORY PROC
    push ax
    push bx
    push cx
    push dx

    lea ax, END_POINT
    mov bx,10h
    xor dx,dx
    div bx
    inc ax
    mov bx,ax
    mov al,0
    mov ah,4Ah
    int 21h

    pop dx
    pop cx
    pop bx
    pop ax
    ret
FREE_UNUSED_MEMORY ENDP

BEGIN:
; Код
    call PRINT_AVAILABLE_MEMORY

```

```

call PRINT_EXTENDED_MEMORY
mov dx, offset TABLE
call WRITE
call FREE_UNUSED_MEMORY

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

call PRINT_MCB_DATA
; Выход в DOS
xor AL,AL
mov AH,4Ch
int 21H
END_POINT:
TESTPC ENDS

END START ; Конец модуля, START - точка входа

Название файла: lab3_4.asm
; Исследование организации управления основной памятью
; 25.03.2021
; Nistratov Dmitry

; 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
AVAILABLE_MEM    db 'Available memory:      ', 10, 13, '$'

```

```

EXTENDED_MEM      db 'Extended memory:      ', 10, 13, '$'
TABLE      db 'Table: ', 10, 13, '$'
TABLE_DATA  db '                                ', 10, 13, '$' ;

```

67

```

MEMORY_MSG db 'Memory allocation error!', 10, 13, '$'

```

```

; 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 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 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
;_____
; Перевод в 10 с/с 10-ти разрядного числа
; в AX - число, DI - адрес последнего символа
WRD_TO_DEC PROC NEAR
    push CX
    push DX
    mov CX,10
loop_b: div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp AX,10
    jae loop_b
    cmp AL,00h
    je endl
    or AL,30h
    mov [SI],AL
endl:  pop DX

```

```

    pop CX
    ret
WRD_TO_DEC ENDP

;_____
; КОД
WRITE PROC NEAR
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE ENDP

WRITEBYTE PROC NEAR
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
WRITEBYTE ENDP

EMPTY_MEMORY PROC NEAR
    push ax
    push bx
    push dx

    pop ax
    pop bx
    pop dx
    ret

```

EMPTY_MEMORY ENDP

PRINT_AVAILABLE_MEMORY PROC NEAR

push ax

push bx

push dx

push si

xor ax, ax

mov ah, 4Ah

mov bx, 0FFFFh

int 21h

mov ax, 10h

mul bx

mov si, offset AVAILABLE_MEM

add si, 23

call WRD_TO_DEC

mov dx, offset AVAILABLE_MEM

call WRITE

pop si

pop dx

pop bx

pop ax

ret

PRINT_AVAILABLE_MEMORY ENDP

PRINT_EXTENDED_MEMORY PROC NEAR

push ax

push bx

push dx

push si

xor dx, dx

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 al, bl

mov si, offset EXTENDED_MEM

add si, 21

call WRD_TO_DEC

mov dx, offset EXTENDED_MEM

call WRITE

pop si

pop dx

pop bx

pop ax

ret

PRINT_EXTENDED_MEMORY ENDP

; MCB type (1 Byte) 4Dh / 5Ah

MCB_TYPE PROC NEAR

```

push ax
push di

mov di, offset TABLE_DATA
add di, 5
xor ah, ah
mov al, es:[00h]
call BYTE_TO_HEX
mov [di], al
inc di
mov [di], ah

pop di
pop ax
ret
MCB_TYPE ENDP

; PSP Address (2Byte)
PSP_ADDRESS PROC NEAR
push ax
push di

mov di, offset TABLE_DATA
mov ax, es:[01h]
add di, 19
call WRD_TO_HEX

pop di
pop ax
ret

```

PSP_ADDRESS ENDP

; Size (2 Byte)

MCB_SIZE PROC NEAR

push ax

push bx

push di

push si

mov di, offset TABLE_DATA

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 29

mov si, di

call WRD_TO_DEC

pop si

pop di

pop bx

pop ax

ret

MCB_SIZE ENDP

; SC or SD (8 Byte)

SC_SD PROC NEAR

push bx

push dx

push di

```

mov di, offset TABLE_DATA
add di, 33
    mov bx, 0h
JMP_NEXT:
    mov dl, es:[bx + 8]
    mov [di], dl
    inc di
    inc bx
    cmp bx, 8h
jne JMP_NEXT

pop di
pop dx
pop bx
ret
SC_SD ENDP

PRINT_MCB_DATA PROC NEAR
    push ax
    push bx
    push es
    push dx

    mov ah, 52h
    int 21h
    mov es, es:[bx - 2h]
NEXT_MCB:
    call MCB_TYPE
    call PSP_ADDRESS
    call MCB_SIZE

```



```
call SC_SD
mov ax, es:[03h]
mov bl, es:[00h]
mov dx, offset TABLE_DATA
call WRITE
```

```
mov cx, es
add ax, cx
inc ax
mov es, ax
```

```
cmp bl, 4Dh ; if not last (5Ah)
je NEXT_MCB
```

```
pop dx
pop es
pop bx
pop ax
```

```
ret
```

```
PRINT_MCB_DATA ENDP
```

```
BEGIN:
```

```
; Код
```

```
call PRINT_AVAILABLE_MEMORY
call PRINT_EXTENDED_MEMORY
```

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

```

jnc GOOD
mov dx,offset MEMORY_MSG
call WRITE

GOOD:
mov dx, offset TABLE
call WRITE
call PRINT_MCB_DATA
; Выход в DOS
xor AL,AL
mov AH,4Ch
int 21H
TESTPC ENDS

END START ; Конец модуля, START - точка входа

```