

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

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

Студентка гр 9383

Сергиенкова А.А.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2021

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. 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 4АН»). 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.

Были написаны строки для вывода информации:

- `ACCESSED_MEMORY db 13,10,'Size of accessed memory: $';`
- `EXTENDED_MEMORY db 13,10,'Size of extended memory: $';`
- `STR_BYTE db ' byte $';`
- `STR_MCB db 13,10,'MCB:0 $';`
- `ADRESS db 'Adress: $';`
- `ADRESS_PSP db 'PSP adress: $';`
- `STR_SIZE db 'Size: $';`
- `MCB_SD_SC db ' SD/SC: $' ;`
- `STR_ERROR db 13,10,'Memory Error!$';`
- `STR_SUCCECC db 13,10,'Success!$'.`

Были добавлены функции:

- `TETR_TO_HEX` – Перевод десятичной цифры в код символа.
- `BYTE_TO_HEX` – Перевод байта в 16-ной с/с в символьный код.
- `WRD_TO_HEX` – Перевод слова в 16-ной с/с в символьный код.
- `BYTE_TO_DEC` – Перевод байта в 16-ной с/с в символьный код в 10-ной с/с.

Были написаны функции:

- `WRITE_STRING` – Вывод строки на экран.
- `WRITE_SIZE` – Запись десятичного числа в строку.
- `PRINT_MCB` – Вывод информации о цепочке блоков управления памятью.
- `FREE_UNUSED_MEM` – Освобождение неиспользуемой памяти.
- `GET_EXTRA_MEM` – Выделение дополнительной памяти.

В результате выполнения были получены следующие значения(рис.1-4):

```
F:\>lab3_1.com

Size of accessed memory: 648912 byte
Size of extended memory: 245760 byte
MCB:01 Address: 016F PSP address: 0008 Size: 16 SD/SC:
MCB:02 Address: 0171 PSP address: 0000 Size: 64 SD/SC:
MCB:03 Address: 0176 PSP address: 0040 Size: 256 SD/SC:
MCB:04 Address: 0187 PSP address: 0192 Size: 144 SD/SC:
MCB:05 Address: 0191 PSP address: 0192 Size: 648912 SD/SC: LAB3_1
```

Рисунок 1 – результат работы программы по условиям 1 пункта

```
F:\>lab3_2.com

Size of accessed memory: 648912 byte
Size of extended memory: 245760 byte
MCB:01 Address: 016F PSP address: 0008 Size: 16 SD/SC:
MCB:02 Address: 0171 PSP address: 0000 Size: 64 SD/SC:
MCB:03 Address: 0176 PSP address: 0040 Size: 256 SD/SC:
MCB:04 Address: 0187 PSP address: 0192 Size: 144 SD/SC:
MCB:05 Address: 0191 PSP address: 0192 Size: 864 SD/SC: LAB3_2
MCB:06 Address: 01CB PSP address: 0000 Size: 648032 SD/SC:
F:\>
```

Рисунок 2 – результат работы программы по условиям 2 пункта

```
F:\>lab3_3.com

Size of accessed memory: 648912 byte
Success!
Size of extended memory: 245760 byte
MCB:01 Address: 016F PSP address: 0008 Size: 16 SD/SC:
MCB:02 Address: 0171 PSP address: 0000 Size: 64 SD/SC:
MCB:03 Address: 0176 PSP address: 0040 Size: 256 SD/SC:
MCB:04 Address: 0187 PSP address: 0192 Size: 144 SD/SC:
MCB:05 Address: 0191 PSP address: 0192 Size: 912 SD/SC: LAB3_3
MCB:06 Address: 01CB PSP address: 0192 Size: 65536 SD/SC: LAB3_3
MCB:07 Address: 110C PSP address: 0000 Size: 582432 SD/SC:
F:\>
```

Рисунок 3 – результат работы программы по условиям 3 пункта

```
F:\>lab3_4.com

Size of accessed memory: 648912 byte
Memory Error!
Size of extended memory: 245760 byte
MCB:01 Address: 016F PSP address: 0008 Size: 16 SD/SC:
MCB:02 Address: 0171 PSP address: 0000 Size: 64 SD/SC:
MCB:03 Address: 0176 PSP address: 0040 Size: 256 SD/SC:
MCB:04 Address: 0187 PSP address: 0192 Size: 144 SD/SC:
MCB:05 Address: 0191 PSP address: 0192 Size: 912 SD/SC: LAB3_4
MCB:06 Address: 01CB PSP address: 0000 Size: 647984 SD/SC:
F:\>
```

Рисунок 4 – результат работы программы по условиям 4 пункта

Выводы.

Исследованы структуры данных и работа функций управления памятью ядра операционной системы.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что означает “Доступный объём памяти”?

Часть оперативной памяти выделяемой системой программе для её корректной работы.

2. Где МСВ блок Вашей программы в списке?

На первом рисунке – МСВ блок программы находится в конце списка.

На втором рисунке – МСВ блок программы находится на предпоследнем месте т.к. последнее место занимает блок с высвобожденной неиспользуемой памятью.

На третьем рисунке – МСВ блок программы находится на пятом и шестом месте, т.к. сначала мы высвобождаем неиспользуемую память, а после выделяем от неё небольшую часть под программу.

На четвёртом рисунке – МСВ блок программы находится предпоследним т.к. мы не смогли выделить дополнительную память не освободив незанятую.

3. Какой размер памяти занимает программа в каждом случае?

В первом случае: 648912 байт.

Во втором: 864 байт.

В третьем: $864 + 65536 = 66400$ байт.

В четвёртом: 864 байта.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл lab3_1.asm

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

; Данные

ACCESSED_MEMORY db 13,10,'Size of accessed memory: \$'

EXTENDED_MEMORY db 13,10,'Size of extended memory: \$'

STR_BYTE db ' byte \$'

STR_MCB db 13,10,'MCB:0 \$'

ADRESS db 'Adress: \$'

ADRESS_PSP db 'PSP adress: \$'

STR_SIZE db 'Size: \$'

MCB_SD_SC db ' SD/SC: \$'

STR_ERROR db 13,10,'Memory Error!\$'

STR_SUCCECC db 13,10,'Success!\$'

; 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_1
    or AL,30h
    mov [SI],AL
end_1:
    pop DX
    pop CX
    ret

```

BYTE_TO_DEC ENDP

;-----

WRITE_STRING PROC near

push AX

mov AH,09h

int 21h

pop AX

ret

WRITE_STRING ENDP

;-----

WRITE_SIZE PROC

push ax

push bx

push cx

push dx

push si

mov bx,10h

mul bx

mov bx,0ah

xor cx,cx

separation:

div bx

push dx

inc cx

xor dx,dx

cmp ax,0h

jnz separation

write_symbol:

pop dx
or dl,30h
mov [si], dl
inc si
loop write_symbol

pop si
pop dx
pop cx
pop bx
pop ax
ret

WRITE_SIZE ENDP

;-----

PRINT_MCB PROC

push ax
push bx
push cx
push dx
push si

mov ah,52h
int 21h
mov ax,es:[bx-2]
mov es,ax
mov cl,1

pargraph_MCB:

lea si, STR_MCB

add si, 7

mov al,cl

push cx

call BYTE_TO_DEC

lea dx, STR_MCB

call WRITE_STRING

mov ax,es

lea di,ADRESS

add di,12

call WRD_TO_HEX

lea dx,ADRESS

call WRITE_STRING

xor ah,ah

mov al,es:[0]

push ax

mov ax,es:[1]

lea di, ADRESS_PSP

add di, 15

call WRD_TO_HEX

lea dx, ADRESS_PSP

call WRITE_STRING

mov ax,es:[3]

lea si,STR_SIZE

```

add si, 6
call WRITE_SIZE
lea dx,STR_SIZE
call WRITE_STRING
xor dx, dx
lea dx , MCB_SD_SC
call WRITE_STRING
mov cx,8
xor di,di

```

write_char:

```

mov dl,es:[di+8]
mov ah,02h
int 21h
inc di
loop write_char

```

```

mov ax,es:[3]
mov bx,es
add bx,ax
inc bx
mov es,bx
pop ax
pop cx
inc cx
cmp al,5Ah ; проверка на не последний ли это сегмент
je exit
cmp al,4Dh
jne exit

```

```
    jmp pargraph_MCB
```

```
exit:
```

```
pop si
```

```
pop dx
```

```
pop cx
```

```
pop bx
```

```
pop ax
```

```
ret
```

```
PRINT_MCB ENDP
```

```
PRINT_MEM_SIZE proc near
```

```
push ax
```

```
push bx
```

```
push si
```

```
mov AL,30h
```

```
out 70h,AL
```

```
in AL,71h
```

```
mov BL,AL
```

```
mov AL,31h
```

```
out 70h,AL
```

```
in AL,71h
```

```
mov bh,al
```

```
mov ax,bx
```

```
lea si,EXTENDED_MEMORY
```

```
add si, 27
```

```
call WRITE_SIZE
lea dx,EXTENDED_MEMORY
call WRITE_STRING
lea dx,STR_BYTE
call WRITE_STRING
```

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

```
PRINT_MEM_SIZE ENDP
```

```
PRINT_AVAILABLE_MEM_SIZE PROC near
```

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

```
;доступная память
```

```
mov ah,4ah
mov bx,0ffffh
int 21h
mov ax,bx
lea si, ACCESSED_MEMORY
add si, 27 ;смещение для числа
call WRITE_SIZE
lea dx, ACCESSED_MEMORY
call WRITE_STRING
```



```
lea dx,STR_BYTE
call WRITE_STRING
```

```
pop si
pop bx
pop si
ret
```

```
PRINT_AVAILABLE_MEM_SIZE ENDP
```

```
; Код
```

```
BEGIN:
```

```
    ;доступная память
```

```
    call PRINT_AVAILABLE_MEM_SIZE
```

```
    ; расширенная память
```

```
    call PRINT_MEM_SIZE
```

```
    ;MCB
```

```
    call PRINT_MCB
```

```
    xor AL,AL
```

```
    mov AH,4Ch
```

```
    int 21H
```

```
TESTPC ENDS
```

```
END START
```

Файл lab3_2.asm

```
TESTPC SEGMENT
```

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

```
    ORG 100H
```

START: JMP BEGIN

; Данные

ACCESSED_MEMORY db 13,10,'Size of accessed memory: \$'

EXTENDED_MEMORY db 13,10,'Size of extended memory: \$'

STR_BYTE db ' byte \$'

STR_MCB db 13,10,'MCB:0 \$'

ADRESS db 'Adress: \$'

ADRESS_PSP db 'PSP adress: \$'

STR_SIZE db 'Size: \$'

MCB_SD_SC db ' SD/SC: \$'

STR_ERROR db 13,10,'Memory Error!\$'

STR_SUCCECC db 13,10,'Success!\$'

; 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_1

or AL,30h

mov [SI],AL

end_1:

pop DX

pop CX

ret

BYTE_TO_DEC ENDP

;-----

WRITE_STRING PROC near

push AX

mov AH,09h

int 21h

pop AX

```
    ret
WRITE_STRING ENDP
;-----
```

```
WRITE_SIZE PROC
```

```
    push ax
```

```
    push bx
```

```
    push cx
```

```
    push dx
```

```
    push si
```

```
        mov bx,10h
```

```
        mul bx
```

```
        mov bx,0ah
```

```
        xor cx,cx
```

```
separation:
```

```
    div bx
```

```
    push dx
```

```
    inc cx
```

```
    xor dx,dx
```

```
    cmp ax,0h
```

```
    jnz separation
```

```
write_symbol:
```

```
    pop dx
```

```
    or dl,30h
```

```
    mov [si], dl
```

```
    inc si
```

```

        loop write_symbol

    pop si
    pop dx
    pop cx
    pop bx
    pop ax
    ret
WRITE_SIZE ENDP
;-----
PRINT_MCB PROC
    push ax
    push bx
    push cx
    push dx
    push si

    mov ah,52h
    int 21h
    mov ax,es:[bx-2]
    mov es,ax
    mov cl,1

pagraph_MCB:
    lea si, STR_MCB
    add si, 7

    mov al,cl
    push cx

```

```
call BYTE_TO_DEC
lea dx, STR_MCB
call WRITE_STRING
```

```
mov ax,es
lea di,ADRESS
add di,12
call WRD_TO_HEX
lea dx,ADRESS
call WRITE_STRING
```

```
xor ah,ah
mov al,es:[0]
push ax
mov ax,es:[1]
lea di, ADRESS_PSP
add di, 15
call WRD_TO_HEX
lea dx, ADRESS_PSP
call WRITE_STRING
mov ax,es:[3]
lea si,STR_SIZE
add si, 6
call WRITE_SIZE
lea dx,STR_SIZE
call WRITE_STRING
xor dx, dx
lea dx , MCB_SD_SC
call WRITE_STRING
```

```
mov cx,8
```

```
xor di,di
```

```
write_char:
```

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

```
mov ah,02h
```

```
int 21h
```

```
inc di
```

```
loop write_char
```

```
mov ax,es:[3]
```

```
mov bx,es
```

```
add bx,ax
```

```
inc bx
```

```
mov es,bx
```

```
pop ax
```

```
pop cx
```

```
inc cx
```

```
cmp al,5Ah ; проверка на не последний ли это сегмент
```

```
je exit
```

```
cmp al,4Dh
```

```
jne exit
```

```
jmp paragraph_MCB
```

```
exit:
```

```
pop si
```

```
pop dx
```

```
pop cx
```

```
pop bx
```



```
    pop ax
    ret
PRINT_MCB ENDP
```

```
PRINT_MEM_SIZE proc near
```

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

```
    mov AL,30h
    out 70h,AL
    in AL,71h
    mov BL,AL
    mov AL,31h
    out 70h,AL
    in AL,71h
```

```
    mov bh,al
    mov ax,bx
    lea si,EXTENDED_MEMORY
    add si, 27
    call WRITE_SIZE
    lea dx,EXTENDED_MEMORY
    call WRITE_STRING
    lea dx,STR_BYTE
    call WRITE_STRING
```

```
    pop si
```

```
pop bx
pop ax
ret
```

```
PRINT_MEM_SIZE ENDP
```

```
PRINT_AVAILABLE_MEM_SIZE PROC near
```

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

```
;доступная память
```

```
mov ah,4ah
mov bx,0ffffh
int 21h
mov ax,bx
lea si, ACCESSED_MEMORY
add si, 27 ;смещение для числа
call WRITE_SIZE
lea dx, ACCESSED_MEMORY
call WRITE_STRING
lea dx,STR_BYTE
call WRITE_STRING
```

```
pop si
pop bx
pop si
ret
```

PRINT_AVAILABLE_MEM_SIZE ENDP

; Код

BEGIN:

;доступная память

call PRINT_AVAILABLE_MEM_SIZE

; расширенная память

call PRINT_MEM_SIZE

;MCB

call PRINT_MCB

xor AL,AL

mov AH,4Ch

int 21H

TESTPC ENDS

END START

Файл lab3_3.asm

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

; Данные

ACCESSED_MEMORY db 13,10,'Size of accessed memory: \$'

EXTENDED_MEMORY db 13,10,'Size of extended memory: \$'

STR_BYTE db ' byte \$'

STR_MCB db 13,10,'MCB:0 \$'

ADRESS db 'Adress: \$'

```

ADDRESS_PSP db 'PSP adress:    $'
STR_SIZE db 'Size:    $'
MCB_SD_SC db ' SD/SC: $'
STR_ERROR db 13,10,'Memory Error!$'
STR_SUCCECC db 13,10,'Success!$'

```

```

; 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
;-----

WRITE_STRING PROC near
    push AX
    mov AH,09h
    int 21h
    pop AX
    ret
WRITE_STRING ENDP
;-----

WRITE_SIZE PROC near
    push ax
    push bx

```

push cx

push dx

push si

mov bx,10h

mul bx

mov bx,0ah

xor cx,cx

delenie:

div bx

push dx

inc cx

xor dx,dx

cmp ax,0h

jnz delenie

write_symbol:

pop dx

or dl,30h

mov [si], dl

inc si

loop write_symbol

pop si

pop dx

pop cx

pop bx

pop ax

ret

WRITE_SIZE ENDP

;-----

PRINT_MCB PROC near

push ax

push bx

push cx

push dx

push si

mov ah,52h

int 21h

mov ax,es:[bx-2]

mov es,ax

xor cx,cx

inc cx

pargraph_MCB:

lea si, STR_MCB

add si, 7

mov al,cl

push cx

call BYTE_TO_DEC

lea dx, STR_MCB

call WRITE_STRING

mov ax,es

lea di,ADDRESS

add di,12

call WRD_TO_HEX


```
lea dx,ADRESS
call WRITE_STRING
```

```
xor ah,ah
mov al,es:[0]
push ax
mov ax,es:[1]
lea di, ADDRESS_PSP
add di, 15
call WRD_TO_HEX
lea dx, ADDRESS_PSP
call WRITE_STRING
mov ax,es:[3]
lea si,STR_SIZE
add si, 6
call WRITE_SIZE
lea dx,STR_SIZE
call WRITE_STRING
xor dx, dx
lea dx , MCB_SD_SC
call WRITE_STRING
mov cx,8
xor di,di
```

write_char:

```
mov dl,es:[di+8]
mov ah,02h
int 21h
inc di
```

```
loop write_char
```

```
mov ax,es:[3]
```

```
mov bx,es
```

```
add bx,ax
```

```
inc bx
```

```
mov es,bx
```

```
pop ax
```

```
pop cx
```

```
inc cx
```

```
cmp al,5Ah ; проверка на не последний ли это сегмент
```

```
je exit
```

```
cmp al,4Dh
```

```
jne exit
```

```
jmp paragraph_MCB
```

```
exit:
```

```
pop si
```

```
pop dx
```

```
pop cx
```

```
pop bx
```

```
pop ax
```

```
ret
```

```
PRINT_MCB ENDP
```

```
FREE_UNUSED_MEM PROC near
```

```
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_MEM ENDP

GET_EXTRA_MEM PROC near

push ax

push bx

push dx

mov bx,1000h

mov ah,48h

int 21h

jc ERROR1 ;проверка флага CF

jmp SUCCECC

ERROR1:

lea dx,STR_ERROR
call WRITE_STRING
JMP END1

SUCCECC:

lea dx,STR_SUCCECC
call WRITE_STRING

END1:

pop dx
pop bx
pop ax
ret

GET_EXTRA_MEM ENDP

PRINT_AVAILABLE_MEM_SIZE PROC near

push ax
push bx
push si

;доступная память

mov ah,4ah
mov bx,0ffffh
int 21h
mov ax,bx
lea si, ACCESSED_MEMORY
add si, 27 ;смещение для числа

```
call WRITE_SIZE
lea dx, ACCESSED_MEMORY
call WRITE_STRING
lea dx,STR_BYTE
call WRITE_STRING
```

```
pop si
pop bx
pop si
ret
```

```
PRINT_AVAILABLE_MEM_SIZE ENDP
```

```
PRINT_MEM_SIZE proc near
```

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

```
mov AL,30h
out 70h,AL
in AL,71h
mov BL,AL
mov AL,31h
out 70h,AL
in AL,71h
```

```
mov bh,al
mov ax,bx
```

```
lea si,EXTENDED_MEMORY
add si, 27
call WRITE_SIZE
lea dx,EXTENDED_MEMORY
call WRITE_STRING
lea dx,STR_BYTE
call WRITE_STRING
```

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

```
PRINT_MEM_SIZE ENDP
```

```
; Код
```

```
BEGIN:
```

```
;доступная память
call PRINT_AVAILABLE_MEM_SIZE
call FREE_UNUSED_MEM
call GET_EXTRA_MEM
; расширенная память
call PRINT_MEM_SIZE
;MCB
call PRINT_MCB
```

```
xor AL,AL
mov AH,4Ch
```

```

        int 21H
    end_point:
    TESTPC ENDS
END START

```

Файл lab3_4.asm

```

TESTPC SEGMENT

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

    ORG 100H

START: JMP BEGIN

; Данные
ACCESSED_MEMORY db 13,10,'Size of accessed memory:    $'
EXTENDED_MEMORY db 13,10,'Size of extended memory:    $'
STR_BYTE db ' byte $'
STR_MCB db 13,10,'MCB:0  $'
ADRESS db 'Adress:    $'
ADRESS_PSP db 'PSP adress:    $'
STR_SIZE db 'Size:    $'
MCB_SD_SC db ' SD/SC: $'
STR_ERROR db 13,10,'Memory Error!$'
STR_SUCCECC db 13,10,'Success!$'

; 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_1
or AL,30h
mov [SI],AL
end_1:
pop DX

```

```

    pop CX
    ret
BYTE_TO_DEC ENDP
;-----
WRITE_STRING PROC near
    push AX
    mov AH,09h
    int 21h
    pop AX
    ret
WRITE_STRING ENDP
;-----

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

    mov bx,10h
    mul bx
    mov bx,0ah
    xor cx,cx
delenie:
    div bx
    push dx
    inc cx
    xor dx,dx

```

```
    cmp ax,0h
```

```
    jnz delenie
```

```
write_symbol:
```

```
    pop dx
```

```
    or dl,30h
```

```
    mov [si], dl
```

```
    inc si
```

```
    loop write_symbol
```

```
    pop si
```

```
    pop dx
```

```
    pop cx
```

```
    pop bx
```

```
    pop ax
```

```
    ret
```

```
WRITE_SIZE ENDP
```

```
;-----
```

```
PRINT_MCB PROC near
```

```
    push ax
```

```
    push bx
```

```
    push cx
```

```
    push dx
```

```
    push si
```

```
    mov ah,52h
```

```
    int 21h
```

```
    mov ax,es:[bx-2]
```

```
    mov es,ax
```

xor cx,cx

inc cx

pargraph_MCB:

lea si, STR_MCB

add si, 7

mov al,cl

push cx

call BYTE_TO_DEC

lea dx, STR_MCB

call WRITE_STRING

mov ax,es

lea di,ADRESS

add di,12

call WRD_TO_HEX

lea dx,ADRESS

call WRITE_STRING

xor ah,ah

mov al,es:[0]

push ax

mov ax,es:[1]

lea di, ADRESS_PSP

add di, 15

call WRD_TO_HEX

lea dx, ADRESS_PSP

call WRITE_STRING

mov ax,es:[3]

```
lea si,STR_SIZE
add si, 6
call WRITE_SIZE
lea dx,STR_SIZE
call WRITE_STRING
xor dx, dx
lea dx , MCB_SD_SC
call WRITE_STRING
mov cx,8
xor di,di
```

write_char:

```
mov dl,es:[di+8]
mov ah,02h
int 21h
inc di
loop write_char
```

```
mov ax,es:[3]
```

```
mov bx,es
```

```
add bx,ax
```

```
inc bx
```

```
mov es,bx
```

```
pop ax
```

```
pop cx
```

```
inc cx
```

```
cmp al,5Ah ; проверка на не последний ли это сегмент
```

```
je exit
```

```
cmp al,4Dh
```

```
jne exit  
jmp pargraph_MCB
```

exit:

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

PRINT_MCB ENDP

FREE_UNUSED_MEM PROC near

```
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_MEM ENDP

GET_EXTRA_MEM PROC near
    push ax
    push bx
    push dx

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

    jc ERROR1 ;проверка флага CF
    jmp SUCCECC

ERROR1:
    lea dx,STR_ERROR
    call WRITE_STRING
    JMP END1
SUCCECC:
    lea dx,STR_SUCCECC
    call WRITE_STRING
END1:
    pop dx
    pop bx
    pop ax

```

```

    ret
GET_EXTRA_MEM ENDP

PRINT_AVAILABLE_MEM_SIZE PROC near

    push ax
    push bx
    push si

;доступная память
    mov ah,4ah
    mov bx,0ffffh
    int 21h
    mov ax,bx
    lea si, ACCESSED_MEMORY
    add si, 27 ;смещение для числа
    call WRITE_SIZE
    lea dx, ACCESSED_MEMORY
    call WRITE_STRING
    lea dx,STR_BYTE
    call WRITE_STRING

    pop si
    pop bx
    pop si
    ret

PRINT_AVAILABLE_MEM_SIZE ENDP

```


PRINT_MEM_SIZE proc near

push ax

push bx

push si

mov AL,30h

out 70h,AL

in AL,71h

mov BL,AL

mov AL,31h

out 70h,AL

in AL,71h

mov bh,al

mov ax,bx

lea si,EXTENDED_MEMORY

add si, 27

call WRITE_SIZE

lea dx,EXTENDED_MEMORY

call WRITE_STRING

lea dx,STR_BYTE

call WRITE_STRING

pop si

pop bx

pop ax

ret

```

PRINT_MEM_SIZE ENDP

; Код
BEGIN:
;доступная память
    call PRINT_AVAILABLE_MEM_SIZE
    call GET_EXTRA_MEM
    call FREE_UNUSED_MEM
    ; расширенная память
    call PRINT_MEM_SIZE
    ;MCB
    call PRINT_MCB

    xor AL,AL
    mov AH,4Ch
    int 21H
end_point:
TESTPC ENDS
END START

```