

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

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

Студент гр. 0382

Гудов Н.Р.

Преподаватели

Ефремов М.А.

Санкт-Петербург

2022

### **Цель работы.**

Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованный в компьютере и способ организации, принятый в ОС. В лабораторной работе рассматривается 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.

### **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 МСВ выводятся как символы, не следует преобразовывать их в шестнадцатеричные числа.

Запустите программу и внимательно оцените результаты. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

**Шаг 2.** Измените программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah прерывания 21h. 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.**

Программный код см. в Приложении А

При работе были созданы следующие процедуры:

REQ\_MEM-процедура запроса памяти  
FREE\_UP\_MEM-процедура освобождения памяти  
PRINT\_MCB-процедура вывода блоков MCB  
PRINT\_MEM-процедура вывода доступной памяти  
PRINT-процедура печати строки.

*Шаг1* Запуск программы без освобождения памяти.

```
B:\>lab31
Available size:      648912
Expanded size:      245760
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: LAB31
```

На этом этапе программа занимает 648912 байт.

*Шаг2* Запуск программы с освобождением неиспользуемой памяти

```
B:\>lab32
Available size:      648912
Expanded size:      245760
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:   800  SD/SC: LAB32
MCB:06 Address:01C4  PSP address:0000  Size: 648096  SD/SC:
```

Блок программы стал меньше. Появился блок с освобожденным местом.

*Шаг3* Запуск программы с запросом дополнительной памяти

```
B:\>lab33
Available size:      648912
Expanded size:      245760
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: LAB33
MCB:06 Address:01C8  PSP address:0192  Size: 65536  SD/SC: LAB33
MCB:07 Address:11C9  PSP address:0000  Size: 582480  SD/SC:
```

Создан новый блок, запрашиваемый программой из ресурсов свободного пространства, ранее выделяемых в отдельный блок.

*Шаг4* Запуск программы с запросом памяти до освобождения

```
B:\>lab34
Available size:      648912
Expanded size:      245760
-warning- memory allocation failed
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: LAB34
MCB:06 Address:01C8  PSP address:0000  Size: 648032  SD/SC:
```

На этом шаге происходит ошибка, ввиду отсутствия блока со свободной памятью, необходимой программе. Блок со свободной памятью создается в конце программы.

### **Контрольные Вопросы.**

1) Что означает "доступный объем памяти"?

*Доступный объём памяти – размер оперативной памяти, который может быть использован программой.*

2) Где MCB блок Вашей программы в списке?

*На каждом шаге MCB блоки программы в графе SD/SC помечены как названием файла.*

3) Какой размер памяти занимает программа в каждом случае?

*Размер считается как сумма размеров участков указанных в MCB программы:*

*– в первом случае – 648912 байт,*

*– во втором – 800 байт,*

*– в третьем – 66400 байт,*

*– четвёртом – 864 байта;*

### **Выводы.**

В результате лабораторной работы была рассмотрена нестраничная память и способ управления динамическими разделами. Были исследованы структуры данных и работа функций управления памятью ядра операционной системы.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЕ КОДЫ ПРОГРАММ

Название файла: lab31.asm

CODE SEGMENT

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

ORG 100H

START: jmp BEGIN

MEM\_FREE DB "Available size: bytes",0DH,0AH,'\$'

MEM\_EXP DB "Expanded size: bytes",0DH,0AH,'\$'

MCB DB "MCB:0 Adress: PSP adress: Size:  
SD/SC: \$"

NEWLINE DB 0DH,0AH,'\$'

;ПРОЦЕДУРЫ

;-----

TETR\_TO\_HEX PROC NEAR

and AL,0Fh

cmp AL,09

jbe next

add AL,07

next: add AL,30h

ret

TETR\_TO\_HEX ENDP

;-----

BYTE\_TO\_HEX PROC NEAR

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_1

```

```

    or AL,30h
    mov [SI],AL
    end_1: pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

;-----
SIZE_TO_DEC PROC NEAR
    push AX
    push BX
    push DX
    push SI
    add SI, 7
    mov BX,10h
    mul BX
    mov BX,10
    write_loop:
        div BX
        or dl,30h
        mov [SI], dl
        dec SI
        xor DX,DX
        cmp AX,0h
        jnz write_loop
    pop SI
    pop DX
    pop BX
    pop AX
    ret
SIZE_TO_DEC ENDP

;-----
PRINT PROC NEAR
    push AX
    mov AH, 09h
    int 21h

```



```

    pop AX
    ret
PRINT ENDP

;-----
PRINT_MEM PROC NEAR
    push AX
    push BX
    push DX
    push SI

    mov AH, 4Ah
    mov BX, 0FFFFh
    int 21h
    mov AX, BX
    mov DX, offset MEM_FREE
    mov SI, DX
    add SI, 22
    call SIZE_TO_DEC
    call PRINT

    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
    mov DX, offset MEM_EXP
    mov SI, DX
    add SI, 22
    call SIZE_TO_DEC
    call PRINT

```

```

    pop SI
    pop DX
    pop BX
    pop AX
    ret
PRINT_MEM ENDP

;-----
PRINT_MCB PROC NEAR
    push AX
    push BX
    push CX
    push DX
    push DI
    push SI

    mov AH, 52h
    int 21h
    mov AX, ES:[BX-2]
    mov ES, AX
    xor CX,CX

    MCB_block:
        inc CX
        mov AL, CL
        mov DX, offset MCB
        mov SI, DX
        add SI, 5
        call BYTE_TO_DEC

        mov AX, ES
        mov DI, SI
        add DI, 14
        call WRD_TO_HEX

        mov AX, ES:[1]

```

```

    add DI, 21
    call WRD_TO_HEX

    mov AX, ES:[3]
    mov SI, DI
    add SI, 11
    call SIZE_TO_DEC
    call PRINT

    xor DI,DI
write_char:
    mov DL, ES:[DI+8]
    mov AH, 02h
    int 21h
    inc DI
    cmp DI, 8
    jl write_char
    mov DX, offset NEWLINE
    call PRINT

    mov AL, ES:[0]
    cmp AL, 4Dh
    jne exit
    mov BX, ES
    add BX, ES:[3]
    inc BX
    mov ES, BX
    jmp MCB_block
exit:
    pop SI
    pop DI
    pop DX
    pop CX
    pop BX
    pop AX

```

```

        ret
PRINT_MCB ENDP

;-----
BEGIN:

    call PRINT_MEM
    call PRINT_MCB
    xor AL,AL
    mov AH, 4Ch
    int 21H
CODE ENDS
END START

```

## Название файла: lab32.asm

```

CODE SEGMENT
    ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
    ORG 100H
    START: jmp BEGIN

    MEM_FREE DB "Available size:          bytes",0DH,0AH,'$'
    MEM_EXP  DB "Expanded size:           bytes",0DH,0AH,'$'
    MCB      DB "MCB:0   Adress:         PSP adress:         Size:"
    SD/SC: '$'
    NEWLINE  DB 0DH,0AH,'$'

;ПРОЦЕДУРЫ
;-----
TETR_TO_HEX PROC NEAR
    and AL,0Fh
    cmp AL,09
    jbe next
    add AL,07
    next: add AL,30h
    ret
TETR_TO_HEX ENDP

;-----

```

```

BYTE_TO_HEX PROC NEAR
    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_1
    or AL,30h
    mov [SI],AL
    end_1: pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

```

-----

```

SIZE_TO_DEC PROC NEAR
    push AX
    push BX
    push DX
    push SI
    add SI, 7
    mov BX,10h
    mul BX
    mov BX,10
    write_loop:
        div BX
        or dl,30h
        mov [SI], dl
        dec SI
        xor DX,DX
        cmp AX,0h
        jnz write_loop
    pop SI
    pop DX

```

```

        pop BX
        pop AX
        ret
SIZE_TO_DEC ENDP

;-----

PRINT PROC NEAR
        push AX
        mov AH, 09h
        int 21h
        pop AX
        ret
PRINT ENDP

;-----

PRINT_MEM PROC NEAR
        push AX
        push BX
        push DX
        push SI

        mov AH, 4Ah
        mov BX, 0FFFFh
        int 21h
        mov AX, BX
        mov DX, offset MEM_FREE
        mov SI, DX
        add SI, 22
        call SIZE_TO_DEC
        call PRINT

        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
    mov DX, offset MEM_EXP
    mov SI, DX
    add SI, 22
    call SIZE_TO_DEC
    call PRINT

    pop SI
    pop DX
    pop BX
    pop AX
    ret
PRINT_MEM ENDP

;-----
PRINT_MCB PROC NEAR
    push AX
    push BX
    push CX
    push DX
    push DI
    push SI

    mov AH, 52h
    int 21h
    mov AX, ES:[BX-2]
    mov ES, AX
    xor CX,CX

    MCB_block:
        inc CX
        mov AL, CL
        mov DX, offset MCB
        mov SI, DX

```



```

add SI, 5
call BYTE_TO_DEC

mov AX, ES
mov DI, SI
add DI, 14
call WRD_TO_HEX

mov AX, ES:[1]
add DI, 21
call WRD_TO_HEX

mov AX, ES:[3]
mov SI, DI
add SI, 11
call SIZE_TO_DEC
call PRINT

xor DI,DI
write_char:
mov DL, ES:[DI+8]
mov AH, 02h
int 21h
inc DI
cmp DI, 8
jl write_char
mov DX, offset NEWLINE
call PRINT

mov AL, ES:[0]
cmp AL, 4Dh
jne exit
mov BX, ES
add BX, ES:[3]
inc BX

```

```

        mov ES, BX
        jmp MCB_block

exit:
pop SI
pop DI
pop DX
pop CX
pop BX
pop AX
ret

PRINT_MCB ENDP

;-----

FREE_UP_MEM PROC NEAR

    push AX
    push BX
    push DX

    mov AX, offset end_address
    mov BX, 10h
    xor DX, DX
    div BX
    add AX, 4
    mov BX, AX
    mov AH, 4Ah
    int 21h

    pop DX
    pop BX
    pop AX
    ret

FREE_UP_MEM ENDP

;-----

BEGIN:

    call PRINT_MEM
    call FREE_UP_MEM

```

```

        call PRINT_MCB
        xor AL,AL
        mov AH, 4Ch
        int 21H
end_address:
CODE ENDS
END START

```

## Название файла: lab33.asm

```

CODE SEGMENT

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

ORG 100H

START: jmp BEGIN


MEM_FREE DB "Available size:          bytes",0DH,0AH,'$'
MEM_EXP  DB "Expanded size:           bytes",0DH,0AH,'$'
MCB      DB "MCB:0   Adress:          PSP adress:          Size:"
SD/SC: '$'

MEM_FAIL DB "-warning- memory allocation failed",0DH,0AH,'$'
NEWLINE  DB 0DH,0AH,'$'


;ПРОЦЕДУРЫ
;-----

TETR_TO_HEX PROC NEAR
    and AL,0Fh
    cmp AL,09
    jbe next
    add AL,07
    next: add AL,30h
    ret
TETR_TO_HEX ENDP
;-----

BYTE_TO_HEX PROC NEAR
    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_1
    or AL,30h
    mov [SI],AL
    end_1: pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

;-----
SIZE_TO_DEC PROC NEAR
    push AX
    push BX
    push DX
    push SI
    add SI, 7
    mov BX,10h
    mul BX
    mov BX,10
    write_loop:
        div BX
        or dl,30h
        mov [SI], dl
        dec SI
        xor DX,DX
        cmp AX,0h
        jnz write_loop
    pop SI
    pop DX
    pop BX
    pop AX
    ret
SIZE_TO_DEC ENDP

```

```

;-----
PRINT PROC NEAR
    push AX
    mov AH, 09h
    int 21h
    pop AX
    ret
PRINT ENDP
;-----
PRINT_MEM PROC NEAR
    push AX
    push BX
    push DX
    push SI

    mov AH, 4Ah
    mov BX, 0FFFFh
    int 21h
    mov AX, BX
    mov DX, offset MEM_FREE
    mov SI, DX
    add SI, 22
    call SIZE_TO_DEC
    call PRINT

    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
    mov DX, offset MEM_EXP

```

```

    mov SI, DX
    add SI, 22
    call SIZE_TO_DEC
    call PRINT

    pop SI
    pop DX
    pop BX
    pop AX
    ret
PRINT_MEM ENDP

;-----
PRINT_MCB PROC NEAR
    push AX
    push BX
    push CX
    push DX
    push DI
    push SI

    mov AH, 52h
    int 21h
    mov AX, ES:[BX-2]
    mov ES, AX
    xor CX,CX

    MCB_block:
        inc CX
        mov AL, CL
        mov DX, offset MCB
        mov SI, DX
        add SI, 5
        call BYTE_TO_DEC

        mov AX, ES

```

```

mov DI, SI
add DI, 14
call WRD_TO_HEX

mov AX, ES:[1]
add DI, 21
call WRD_TO_HEX

mov AX, ES:[3]
mov SI, DI
add SI, 11
call SIZE_TO_DEC
call PRINT

xor DI,DI
write_char:
mov DL, ES:[DI+8]
mov AH, 02h
int 21h
inc DI
cmp DI, 8
jl write_char
mov DX, offset NEWLINE
call PRINT

mov AL, ES:[0]
cmp AL, 4Dh
jne exit
mov BX, ES
add BX, ES:[3]
inc BX
mov ES, BX
jmp MCB_block

exit:
pop SI

```



```

        pop DI
        pop DX
        pop CX
        pop BX
        pop AX
        ret
PRINT_MCB ENDP

;-----
FREE_UP_MEM PROC NEAR

        push AX
        push BX
        push DX

        mov AX, offset end_address
        mov BX, 10h
        xor DX, DX
        div BX
        add AX, 4
        mov BX, AX
        mov AH, 4Ah
        int 21h

        pop DX
        pop BX
        pop AX
        ret
FREE_UP_MEM ENDP

;-----
REQ_MEM PROC near

        push AX
        push BX
        push DX

        mov BX, 1000h
        mov AH, 48h

```

```

        int 21h
        jnc exit_
        mov DX, offset MEM_FAIL
        call PRINT

exit_:
        pop DX
        pop BX
        pop AX
        ret
REQ_MEM ENDP

;-----
BEGIN:
        call PRINT_MEM
        call FREE_UP_MEM
        call REQ_MEM
        call PRINT_MCB
        xor AL,AL
        mov AH, 4Ch
        int 21H

end_address:
CODE ENDS
END START

```

**Название файла: lab34.asm**

```

CODE SEGMENT
        ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
        ORG 100H
        START: jmp BEGIN

MEM_FREE DB "Available size:          bytes",0DH,0AH,'$'
MEM_EXP  DB "Expanded size:           bytes",0DH,0AH,'$'
MCB      DB "MCB:0   Adress:         PSP adress:         Size:
SD/SC: $"

```

```
MEM_FAIL DB "-warning- memory allocation failed",0DH,0AH,'$'  
NEWLINE  DB 0DH,0AH,'$'
```

```
;ПРОЦЕДУРЫ
```

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

```
TETR_TO_HEX PROC NEAR
```

```
    and AL,0Fh
```

```
    cmp AL,09
```

```
    jbe next
```

```
    add AL,07
```

```
next: add AL,30h
```

```
    ret
```

```
TETR_TO_HEX ENDP
```

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

```
BYTE_TO_HEX PROC NEAR
```

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

```

```

BYTE_TO_DEC ENDP

```

```

;-----

```

```

SIZE_TO_DEC PROC NEAR

```

```

    push AX
    push BX
    push DX
    push SI

```

```

    add SI, 7
    mov BX, 10h
    mul BX
    mov BX, 10
write_loop:
    div BX
    or dl, 30h
    mov [SI], dl
    dec SI
    xor DX, DX
    cmp AX, 0h
    jnz write_loop

    pop SI
    pop DX
    pop BX
    pop AX
    ret
SIZE_TO_DEC ENDP
;-----

PRINT PROC NEAR
    push AX
    mov AH, 09h
    int 21h
    pop AX
    ret
PRINT ENDP
;-----

PRINT_MEM PROC NEAR
    push AX
    push BX
    push DX
    push SI

    mov AH, 4Ah
    mov BX, 0FFFFh

```

```

    int 21h
    mov AX, BX
    mov DX, offset MEM_FREE
    mov SI, DX
    add SI, 22
    call SIZE_TO_DEC
    call PRINT

    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
    mov DX, offset MEM_EXP
    mov SI, DX
    add SI, 22
    call SIZE_TO_DEC
    call PRINT

    pop SI
    pop DX
    pop BX
    pop AX
    ret
PRINT_MEM ENDP

;-----
PRINT_MCB PROC NEAR
    push AX
    push BX
    push CX
    push DX

```

```

push DI
push SI

mov AH, 52h
int 21h
mov AX, ES:[BX-2]
mov ES, AX
xor CX,CX

MCB_block:
    inc CX
    mov AL, CL
    mov DX, offset MCB
    mov SI, DX
    add SI, 5
    call BYTE_TO_DEC

    mov AX, ES
    mov DI, SI
    add DI, 14
    call WRD_TO_HEX

    mov AX, ES:[1]
    add DI, 21
    call WRD_TO_HEX

    mov AX, ES:[3]
    mov SI, DI
    add SI, 11
    call SIZE_TO_DEC
    call PRINT

    xor DI,DI
write_char:
    mov DL, ES:[DI+8]

```

```

        mov AH, 02h
        int 21h
        inc DI
        cmp DI, 8
        jl write_char
        mov DX, offset NEWLINE
        call PRINT

        mov AL, ES:[0]
        cmp AL, 4Dh
        jne exit
        mov BX, ES
        add BX, ES:[3]
        inc BX
        mov ES, BX
        jmp MCB_block

exit:
pop SI
pop DI
pop DX
pop CX
pop BX
pop AX
ret
PRINT_MCB ENDP

;-----
FREE_UP_MEM PROC NEAR
    push AX
    push BX
    push DX

    mov AX, offset end_address
    mov BX, 10h
    xor DX, DX
    div BX

```



```

    add AX, 4
    mov BX, AX
    mov AH, 4Ah
    int 21h

    pop DX
    pop BX
    pop AX
    ret
FREE_UP_MEM ENDP
;-----
REQ_MEM PROC near
    push AX
    push BX
    push DX

    mov BX, 1000h
    mov AH, 48h
    int 21h
    jnc exit_
    mov DX, offset MEM_FAIL
    call PRINT

    exit_:
    pop DX
    pop BX
    pop AX
    ret
REQ_MEM ENDP
;-----
BEGIN:
    call PRINT_MEM
    call REQ_MEM
    call FREE_UP_MEM
    call PRINT_MCB

```

```
        xor AL,AL
        mov AH, 4Ch
        int 21H
end_address:
CODE ENDS
END START
```