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(ка) гр. 9382 _____ Голубева В.П.

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 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 MCB 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.

Необходимые сведения для составления программы

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

MCB имеет следующую структуру:

Смещение	Длина поля (байт)	Содержимое поля
00h	1	тип MCB: 5Ah, если последний в списке, 4Dh, если не последний
01h	2	Сегментный адрес PSP владельца участка памяти, либо 0000h - свободный участок, 0006h - участок принадлежит драйверу OS XMS UMB 0007h - участок является исключенной верхней памятью драйверов 0008h - участок принадлежит MS DOS FFFAh - участок занят управляющим блоком 386MAX UMB FFFDh - участок заблокирован 386MAX FFFEh - участок принадлежит 386MAX UMB
03h	2	Размер участка в параграфах
05h	3	Зарезервирован
08h	8	"SC" - если участок принадлежит MS DOS, то в нем системный код "SD" - если участок принадлежит MS DOS, то в нем системные данные

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

Адрес первого MCB хранится во внутренней структуре MS DOS, называемой "List of Lists" (список списков). Доступ к указателю на эгу структуру можно получить используя функцию 52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX—2] и есть адрес самого первого MCB.

Размер расширенной памяти находится в ячейках 3011, 3111 CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта. Размер расширенной памяти в Кбайтах можно определить обращаясь к ячейкам CMOS следующим образом:

```
mov AL,30h ; запись адреса ячейки CMOS
out 70h,AL
in AL,7lh ; чтение младшего байта
mov BL,AL ; размера расширенной памяти
```

mov AL,3lh ; запись адреса ячейки CMOS

out 70h,AL

in AL,7lh ; чтение старшего байта

; размера расширенной памяти

Выполнение работы.

Сначала была написана программа, выполняющая действия в шаге 1. Результаты можно увидеть в Рисунке 1. Квадратики — так отображаются пустые символы. Исходный код программы можно посмотреть в Приложении А.

```
Количество доступной памяти в байтах: 648912
Размер расширенной памяти Кб: 15360

0008h - участок принадлежит MS DOS
Размер участка в байтах: 16
Последовательность символов: □□□□□□□□
-----------------------------------

0000h - свободный участок
Размер участка в байтах: 64
Последовательность символов: □□□□□□□□
-----------------------------------

Пользовательский участок
Размер участка в байтах: 256
Последовательность символов: □□□□□□□□
-----------------------------------

Пользовательский участок
Размер участка в байтах: 144
Последовательность символов: □□□□□□□□
-----------------------------------

Пользовательский участок
Размер участка в байтах: 648912
Последовательность символов: LAB3□□□□
-----------------------------------
```

Рисунок 1. Демонстрация работы программы на шаге 1

Затем программа была модифицирована, чтобы очищать неиспользуемую память. Код можно посмотреть в Приложении Б, результат в Рисунке А.

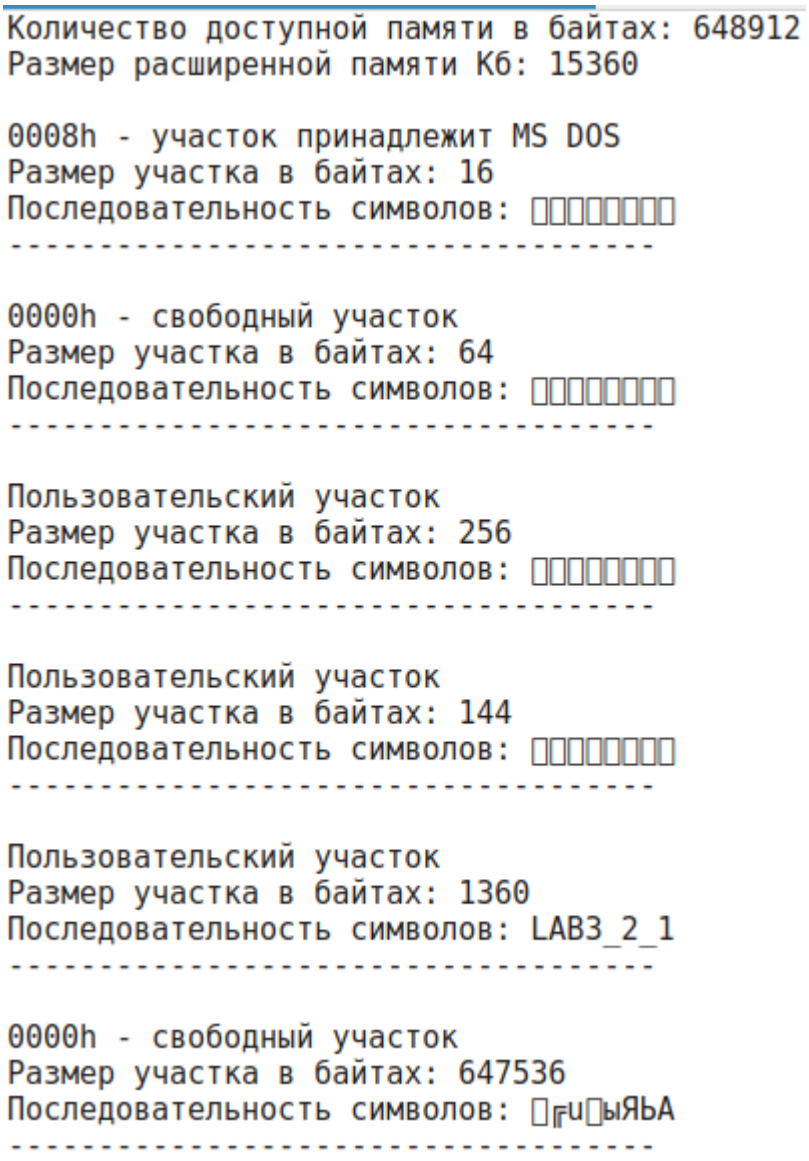

Рисунок А. Освобождение программой неиспользуемой памяти

Затем, после освобождения памяти выделялся блок памяти, размером 64 Кб. Результат можно увидеть в Рисунке 2. Исходный код программы можно посмотреть в Приложении В. Видно, что блок памяти уменьшился, а за ним появился ещё один — размером 64 Кб.

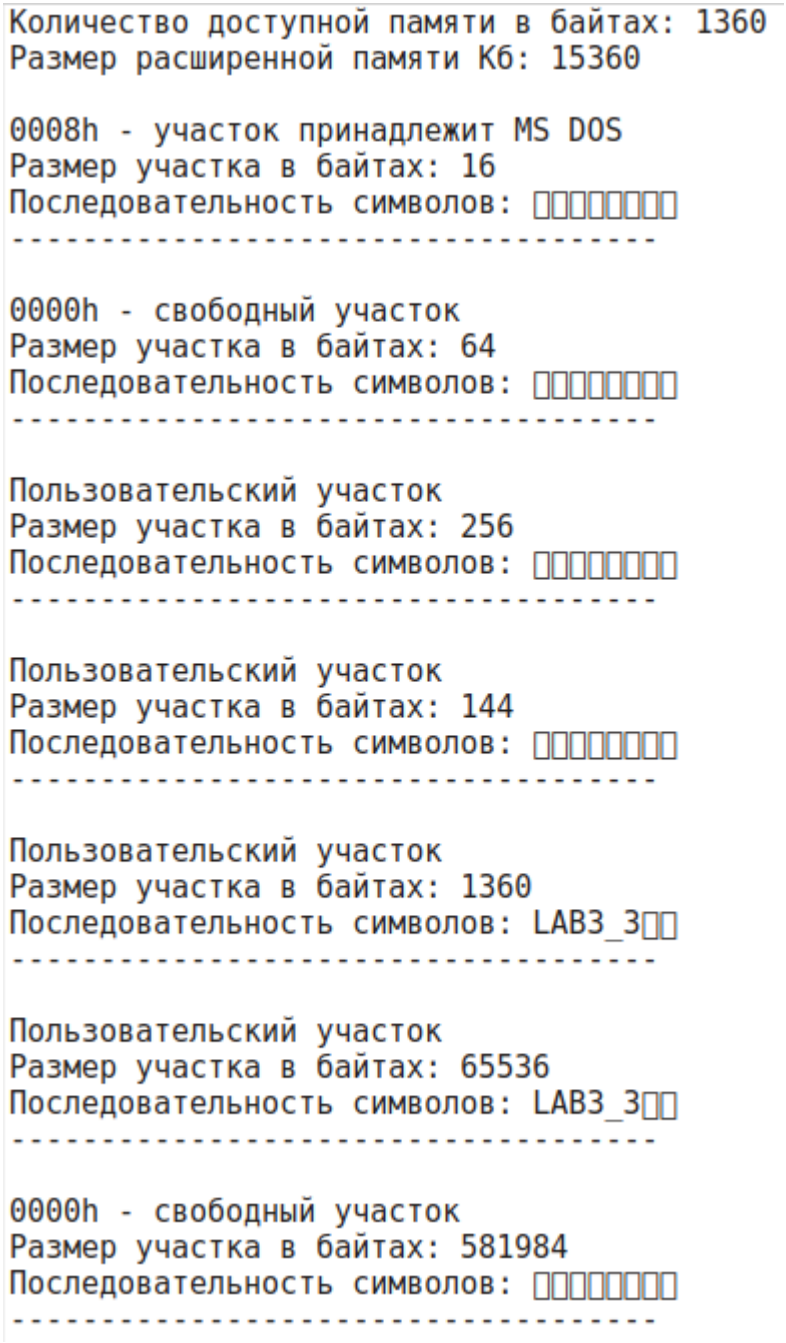

Рисунок 2. Демонстрация работы программы на шаге 3

Затем, выполним действия из шага 4. Результат можно увидеть в Рисунке 3. Исходный код программы можно посмотреть в Приложении Г.

После выжеления и очистки памяти происходит проверка корректности работы функции 4ah прерывания 21h. Выделение происходит корректно, а затем происходит ошибка, потом что вся доступная память для программы уже выделена.

```
Функция 4ah прерывания 21h завершилась корректно
Функция 4ah прерывания 21h завершилась с ошибкой, код ошибки: 0192

Количество доступной памяти в байтах: 648912
Размер расширенной памяти Кб: 15360

0008h - участок принадлежит MS DOS
Размер участка в байтах: 16
Последовательность символов: □□□□□□□□
-----------------------------------

0000h - свободный участок
Размер участка в байтах: 64
Последовательность символов: □□□□□□□□
-----------------------------------

Пользовательский участок
Размер участка в байтах: 256
Последовательность символов: □□□□□□□□
-----------------------------------

Пользовательский участок
Размер участка в байтах: 144
Последовательность символов: □□□□□□□□
-----------------------------------

Пользовательский участок
Размер участка в байтах: 648912
Последовательность символов: LAB3□□□□
-----------------------------------
```

Рисунок 3. Демонстрация работы программы на шаге 4

Ответы на контрольные вопросы

Контрольные вопросы по лабораторной работе №3

1) Что означает "доступный объем памяти"?

Ответ: объём памяти, который занимает и использует программа

2) Где МСВ блок Вашей программы в списке?

Ответ: он находится в конце списка МСВ блоков, перед ним располагается среда, которая тоже принадлежит программе, его можно отследить по одинаковому сегментному адресу PSP

3) Какой размер памяти занимает программа в каждом случае?

Ответ: изначально, программа занимает всю доступную память(649056 байта), потом только то, что ей оставил программист(1404 байта) после очищения неиспользуемой памяти, затем этот же размер, а также 64 Кб, которые выделили, а потом снова всю доступную память(649056 байта).

Выводы.

Были изучены способы управления динамическими разделами памяти, была изучена структура MCB блоков, была исследована работа функций управления памятью ядра операционной системы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3_1.asm

```
LAB3 SEGMENT
    ASSUME CS:LAB3, DS:LAB3, ES:NOTHING, SS:NOTHING
ORG 100H
    START: JMP BEGIN

    str_avail_mem db 'Количество доступной памяти в байтах: ', '$'
    str_exp_mem db 'Размер расширенной памяти Кб: ', '$'
    str_seg1 db 0DH, 0AH, '0000h - свободный участок',0DH, 0AH,'$'
    str_seg2 db 0DH, 0AH, '0006h - участок принадлежит драйверу OS XMS UMB',0DH, 0AH,'$'
    str_seg3 db 0DH, 0AH, '0007h - участок является исключенной верхней памятью драйверов',0DH, 0AH,'$'
    str_seg4 db 0DH, 0AH, '0008h - участок принадлежит MS DOS',0DH, 0AH,'$'
    str_seg5 db 0DH, 0AH, 'FFFAh - участок занят управляющим блоком 386MAX UMB',0DH, 0AH,'$'
    str_seg6 db 0DH, 0AH, 'FFFDh - участок заблокирован 386MAX',0DH, 0AH,'$'
    str_seg7 db 0DH, 0AH, 'FFFEh - участок принадлежит 386MAX UMB',0DH, 0AH,'$'
    str_wr db 0DH, 0AH, 'Пользовательский участок',0DH, 0AH,'$'
    str_size_b db 'Размер участка в байтах: ', '$'
    str_sequ db 'Последовательность символов: ', '$'
    str_ent db ' ', 0DH, 0AH, '$'
    str_div db 0DH, 0AH, '-----------------------------------', 0DH, 0AH, '$'



    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
;byte AL translate in two symbols on 16cc numbers in AX
    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
;translate in 16cc a 16 discharge number
;in AL - number, DI - the address of the last symbol
    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
;translate in 10cc, SI - the adress of the field of younger digit
    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
;--------------------------------

print_addr_psp proc near
    push bx

    mov BH, AH

    mov dl, al
    mov ah, 02h
    int 21h

    mov dl, bh
    mov ah, 02h
    int 21h
```

```
    pop bx


ret
print_addr_psp endp



addr_psp proc near

    cmp ax, 0000h
    mov dx, offset str_seg1
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0006h
    mov dx, offset str_seg2
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0007h
    mov dx, offset str_seg3
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0008h
    mov dx, offset str_seg4
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0FFFAh
    mov dx, offset str_seg5
    je end_addr


    mov [di], ax
    cmp word ptr [di], 0FFFDh
```

```
    mov dx, offset str_seg6
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0FFFEh
    mov dx, offset str_seg7
    je end_addr

    mov dx, offset str_wr

end_addr:



    mov ah, 09h
    int 21h


ret
addr_psp endp

print_number proc near
     mov bx,0Ah
     xor cx,cx
divis:
     div bx
     push dx
     inc cx
     xor dx,dx
     cmp ax,0
     jne divis

print_simb:
     pop dx
     or dl,30h

     mov ah,02h
     int 21h
```

```
    loop print_simb

    call enter

ret
print_number endp

enter proc near
    mov dx, offset str_ent
    mov ah, 09h
    int 21h


ret
enter endp

avail_mem proc near
    mov dx, offset str_avail_mem
    mov ah, 09h
    int 21h

    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, bx

    mov bx, 16
    mul bx

    call print_number


ret
avail_mem endp
```

```
BEGIN:
    call avail_mem


    mov dx, offset str_exp_mem
    mov ah, 09h
    int 21h

    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 bx,1h
     mul bx

    call print_number


    mov ah, 52h
    int 21h


    mov ax, es:[bx-2]
    mov es, ax; in es the address of first mcb

loop_list:
    mov ax, es:[1]

    call addr_psp
```

```
    mov dx, offset str_size_b
    mov ah, 09h
    int 21h

    mov ax, es:[3h]
    mov bx, 16
    mul bx
    call print_number

    mov dx, offset str_sequ
    mov ah, 09h
    int 21h

    xor di, di
    mov cx, 8h

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

    mov dx, offset str_div
    mov ah, 09h
    int 21h


    mov bx, es:[3h]
    mov al,es:[0h]
    cmp al, 5Ah
    je end_pr

    mov ax, es
     add ax,bx
```

```
      inc ax
      mov es,ax; in es the address of follow mcb

    jmp loop_list

end_pr:

    xor AL,AL
    mov AH,4Ch
    int 21H
end_this_code:

LAB3    ENDS
          END START
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3_2.asm

```
TESTPC  SEGMENT

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

ORG 100H

START: JMP begin

str_avail_mem db 'Количество доступной памяти в байтах: ','$'
str_exp_mem db 'Размер расширенной памяти Кб: ', '$'
str_seg1 db 0DH, 0AH, '0000h - свободный участок',0DH, 0AH,'$'
str_seg2 db 0DH, 0AH, '0006h - участок принадлежит драйверу OS XMS UMB',0DH, 0AH,'$'
str_seg3 db 0DH, 0AH, '0007h - участок является исключенной верхней памятью драйверов',0DH, 0AH,'$'
str_seg4 db 0DH, 0AH, '0008h - участок принадлежит MS DOS',0DH, 0AH,'$'
str_seg5 db 0DH, 0AH, 'FFFAh - участок занят управляющим блоком 386MAX UMB',0DH, 0AH,'$'
str_seg6 db 0DH, 0AH, 'FFFDh - участок заблокирован 386MAX',0DH, 0AH,'$'
str_seg7 db 0DH, 0AH, 'FFFEh - участок принадлежит 386MAX UMB',0DH, 0AH,'$'
str_wr db 0DH, 0AH, 'Пользовательский участок',0DH, 0AH,'$'
str_size_b db 'Размер участка в байтах: ', '$'
str_sequ db 'Последовательность символов: ', '$'
str_ent db ' ', 0DH, 0AH, '$'
```

```
    str_div db 0DH, 0AH, '-----------------------------------', 0DH,
0AH, '$'

    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
```

```
PARAGRAPH2BYTES PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH SI

     MOV BX, 10H
     MUL BX
     MOV BX, 0AH
     XOR CX, CX

DIVISION:
     DIV BX
     PUSH DX
     INC CX
     XOR DX, DX
     CMP AX, 0H
     JNZ DIVISION

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
PARAGRAPH2BYTES ENDP

empty_func proc near
    mov ax, es
```

```
    push dx
    mul bx
    add bx, cx
    div cx


    pop dx
ret
empty_func endp


WRITE_STRING PROC NEAR
    PUSH AX
    MOV AH, 9H
    INT 21H
    POP AX
    RET
WRITE_STRING ENDP


avail_mem proc near
    push dx
    push bx
    push ax
    mov dx, offset str_avail_mem
    mov ah, 09h
    int 21h

    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, bx

    mov bx, 16
    mul bx
```

```
    call print_number

    mov dx, offset str_exp_mem
    mov ah, 09h
    int 21h

    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 bx,1h
     mul bx

    call print_number

    pop ax
    pop bx
    pop dx


ret
avail_mem endp

print_number proc near
    push bx
    push cx
    push ax
     mov bx,0Ah
     xor cx,cx
```

```
divis:
      div bx
      push dx
      inc cx
      xor dx,dx
      cmp ax,0
      jne divis

print_simb:
      pop dx
      or dl,30h

      mov ah,02h
      int 21h
     loop print_simb

     call enter
     pop ax
     pop cx
     pop bx

ret
print_number endp


OFFSET_DECIMAL_NUMBER PROC NEAR
OFFSET_LOOP:
    CMP BYTE PTR [SI], ' '
    JNE EXIT_OFFSET_DECIMAL
    INC SI
    JMP OFFSET_LOOP

EXIT_OFFSET_DECIMAL:
    RET
OFFSET_DECIMAL_NUMBER ENDP

addr_psp proc near
```

```
push dx

cmp ax, 0000h
mov dx, offset str_seg1
je end_addr

mov [di], ax
cmp word ptr [di], 0006h
mov dx, offset str_seg2
je end_addr

mov [di], ax
cmp word ptr [di], 0007h
mov dx, offset str_seg3
je end_addr

mov [di], ax
cmp word ptr [di], 0008h
mov dx, offset str_seg4
je end_addr

mov [di], ax
cmp word ptr [di], 0FFFAh
mov dx, offset str_seg5
je end_addr


mov [di], ax
cmp word ptr [di], 0FFFDh
mov dx, offset str_seg6
je end_addr

mov [di], ax
cmp word ptr [di], 0FFFEh
mov dx, offset str_seg7
je end_addr
```

```
    mov dx, offset str_wr

end_addr:


    mov ah, 09h
    int 21h
    pop dx
    mov ax, bx
    mov bx, cx
    mov cx, dx
    mov dx, ax

ret
addr_psp endp

enter proc near
    push dx
    push ax
    mov dx, offset str_ent
    mov ah, 09h
    int 21h
    pop ax
    pop dx

ret
enter endp



PRINT_MCB_CHAIN PROC NEAR
    push es

    mov ah, 52h
    int 21h
```

```
    mov ax, es:[bx-2]
    mov es, ax; in es the address of first mcb

loop_list:
    mov ax, es:[1]

    call addr_psp


    mov dx, offset str_size_b
    mov ah, 09h
    int 21h

    mov ax, es:[3h]
    mov bx, 16
    mul bx
    call print_number

    mov dx, offset str_sequ
    mov ah, 09h
    int 21h

    xor di, di
    mov cx, 8h

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

    mov dx, offset str_div
    mov ah, 09h
    int 21h
```

```
    mov bx, es:[3h]
    mov al,es:[0h]
    cmp al, 5Ah
    je end_pr

    mov ax, es
     add ax,bx
     inc ax
     mov es,ax; in es the address of follow mcb

    jmp loop_list


end_pr:
    pop es
ret
PRINT_MCB_CHAIN endp

FREE_MEM PROC NEAR
    push ax
    push bx
    push dx

    LEA AX, end_this_code
    MOV BX, 10H
    XOR DX, DX
    DIV BX
    INC AX
    MOV BX, AX
    add bx, 5h
    MOV AL, 0
    MOV AH, 4AH
    INT 21H

    pop dx
    pop bx
    pop ax
```

```
    RET
FREE_MEM ENDP


begin:
    CALL avail_mem
    CALL free_mem
    CALL PRINT_MCB_CHAIN

    XOR AL, AL
    MOV AH, 4CH
    INT 21H


end_this_code:
TESTPC  ENDS
END START
```

ПРИЛОЖЕНИЕ В
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3_3.asm

```
LAB3 SEGMENT
    ASSUME CS:LAB3, DS:LAB3, ES:NOTHING, SS:NOTHING
ORG 100H
    START: JMP BEGIN

str_avail_mem db 'Количество доступной памяти в байтах: ', '$'
str_exp_mem db 'Размер расширенной памяти Кб: ', '$'
str_seg1 db 0DH, 0AH, '0000h - свободный участок',0DH, 0AH,'$'
str_seg2 db 0DH, 0AH, '0006h - участок принадлежит драйверу OS XMS UMB',0DH, 0AH,'$'
str_seg3 db 0DH, 0AH, '0007h - участок является исключенной верхней памятью драйверов',0DH, 0AH,'$'
str_seg4 db 0DH, 0AH, '0008h - участок принадлежит MS DOS',0DH, 0AH,'$'
str_seg5 db 0DH, 0AH, 'FFFAh - участок занят управляющим блоком 386MAX UMB',0DH, 0AH,'$'
str_seg6 db 0DH, 0AH, 'FFFDh - участок заблокирован 386MAX',0DH, 0AH,'$'
str_seg7 db 0DH, 0AH, 'FFFEh - участок принадлежит 386MAX UMB',0DH, 0AH,'$'
str_wr db 0DH, 0AH, 'Пользовательский участок',0DH, 0AH,'$'
str_size_b db 'Размер участка в байтах: ', '$'
str_sequ db 'Последовательность символов: ', '$'
str_ent db ' ', 0DH, 0AH, '$'
str_div db 0DH, 0AH, '-----------------------------------', 0DH, 0AH, '$'



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
;byte AL translate in two symbols on 16cc numbers in AX
    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
;translate in 16cc a 16 discharge number
;in AL - number, DI - the address of the last symbol
    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
;translate in 10cc, SI - the adress of the field of younger digit
    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
;--------------------------------

print_addr_psp proc near
    push bx

    mov BH, AH

    mov dl, al
    mov ah, 02h
    int 21h

    mov dl, bh
    mov ah, 02h
    int 21h
```

```
    pop bx


ret
print_addr_psp endp



addr_psp proc near

    cmp ax, 0000h
    mov dx, offset str_seg1
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0006h
    mov dx, offset str_seg2
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0007h
    mov dx, offset str_seg3
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0008h
    mov dx, offset str_seg4
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0FFFAh
    mov dx, offset str_seg5
    je end_addr


    mov [di], ax
    cmp word ptr [di], 0FFFDh
```

```
    mov dx, offset str_seg6
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0FFFEh
    mov dx, offset str_seg7
    je end_addr

    mov dx, offset str_wr

end_addr:



    mov ah, 09h
    int 21h


ret
addr_psp endp

print_number proc near
      mov bx,0Ah
      xor cx,cx
divis:
      div bx
      push dx
      inc cx
      xor dx,dx
      cmp ax,0
      jne divis

print_simb:
      pop dx
      or dl,30h

      mov ah,02h
      int 21h
```

```
    loop print_simb

    call enter

ret
print_number endp

enter proc near
    mov dx, offset str_ent
    mov ah, 09h
    int 21h



ret
enter endp

get_mem proc near
    mov ah, 48h
    mov bx, 1000h;get 64 Kb of memory
    int 21h
ret
get_mem endp

free_mem proc near

    mov ax, cs
    mov es, ax

    mov bx, offset end_this_code
    mov ax, es
    sub bx, ax

    mov ah, 4ah
    int 21h

ret
free_mem endp
```

```
avail_mem proc near
    mov dx, offset str_avail_mem
    mov ah, 09h
    int 21h

    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, bx

    mov bx, 16
    mul bx

    call print_number


ret
avail_mem endp


BEGIN:

    call free_mem
    call get_mem


    call avail_mem


    mov dx, offset str_exp_mem
    mov ah, 09h
    int 21h

    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 bx,1h
     mul bx

    call print_number

    mov ah, 52h
    int 21h

    mov ax, es:[bx-2]
    mov es, ax; in es the address of first mcb

loop_list:
    mov ax, es:[1]

    call addr_psp


    mov dx, offset str_size_b
    mov ah, 09h
    int 21h

    mov ax, es:[3h]
    mov bx, 16
    mul bx
    call print_number

    mov dx, offset str_sequ
```

```
    mov ah, 09h
    int 21h

    xor di, di
    mov cx, 8h

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

    mov dx, offset str_div
    mov ah, 09h
    int 21h

    mov bx, es:[3h]
    mov al,es:[0h]
    cmp al, 5Ah
    je end_pr

    mov ax, es
     add ax,bx
     inc ax
     mov es,ax; in es the address of follow mcb

    jmp loop_list

end_pr:
    xor AL,AL
    mov AH,4Ch
    int 21H
end_this_code:

LAB3    ENDS
          END START
```

ПРИЛОЖЕНИЕ Г
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3_4.asm

```
LAB3 SEGMENT
    ASSUME CS:LAB3, DS:LAB3, ES:NOTHING, SS:NOTHING
ORG 100H
    START: JMP BEGIN

    str_avail_mem db 'Количество доступной памяти в байтах: ', '$'
    str_exp_mem db 'Размер расширенной памяти Кб: ', '$'
    str_seg1 db 0DH, 0AH, '0000h - свободный участок',0DH, 0AH,'$'
    str_seg2 db 0DH, 0AH, '0006h - участок принадлежит драйверу OS XMS UMB',0DH, 0AH,'$'
    str_seg3 db 0DH, 0AH, '0007h - участок является исключенной верхней памятью драйверов',0DH, 0AH,'$'
    str_seg4 db 0DH, 0AH, '0008h - участок принадлежит MS DOS',0DH, 0AH,'$'
    str_seg5 db 0DH, 0AH, 'FFFAh - участок занят управляющим блоком 386MAX UMB',0DH, 0AH,'$'
    str_seg6 db 0DH, 0AH, 'FFFDh - участок заблокирован 386MAX',0DH, 0AH,'$'
    str_seg7 db 0DH, 0AH, 'FFFEh - участок принадлежит 386MAX UMB',0DH, 0AH,'$'
    str_wr db 0DH, 0AH, 'Пользовательский участок',0DH, 0AH,'$'
    str_size_b db 'Размер участка в байтах: ', '$'
    str_sequ db 'Последовательность символов: ', '$'
    str_ent db ' ', 0DH, 0AH, '$'
    str_div db 0DH, 0AH, '-----------------------------------', 0DH, 0AH, '$'
    str_fall_code db 'Функция 4ah прерывания 21h завершилась с ошибкой, код ошибки:     ', 0AH, 0DH, '$'
    str_norm db 'Функция 4ah прерывания 21h завершилась корректно', 0DH, 0AH, '$'
```

```
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
;byte AL translate in two symbols on 16cc numbers in AX
    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
;translate in 16cc a 16 discharge number
;in AL - number, DI - the address of the last symbol
    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
;translate in 10cc, SI - the adress of the field of younger digit
    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
;--------------------------------

print_addr_psp proc near
    push bx

    mov BH, AH

    mov dl, al
    mov ah, 02h
    int 21h
```

```
    mov dl, bh
    mov ah, 02h
    int 21h
    pop bx


ret
print_addr_psp endp



addr_psp proc near

    cmp ax, 0000h
    mov dx, offset str_seg1
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0006h
    mov dx, offset str_seg2
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0007h
    mov dx, offset str_seg3
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0008h
    mov dx, offset str_seg4
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0FFFAh
    mov dx, offset str_seg5
    je end_addr
```

```
    mov [di], ax
    cmp word ptr [di], 0FFFDh
    mov dx, offset str_seg6
    je end_addr

    mov [di], ax
    cmp word ptr [di], 0FFFEh
    mov dx, offset str_seg7
    je end_addr

    mov dx, offset str_wr

end_addr:


    mov ah, 09h
    int 21h


ret
addr_psp endp

print_number proc near
     mov bx,0Ah
     xor cx,cx
divis:
     div bx
     push dx
     inc cx
     xor dx,dx
     cmp ax,0
     jne divis

print_simb:
     pop dx
```

```
      or dl,30h

      mov ah,02h
      int 21h
     loop print_simb

     call enter

ret
print_number endp

enter proc near
    mov dx, offset str_ent
    mov ah, 09h
    int 21h



ret
enter endp

get_mem proc near
    mov ah, 48h
    mov bx, 1000h;get 64 Kb of memory
    int 21h
ret
get_mem endp

free_mem proc near

    mov ax, cs
    mov es, ax

    mov bx, offset end_this_code
    mov ax, es
    sub bx, ax

    mov ah, 4ah
```

```
    int 21h

ret
free_mem endp

avail_mem proc near
    mov dx, offset str_avail_mem
    mov ah, 09h
    int 21h

    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, bx

    mov bx, 16
    mul bx

    call print_number


ret
avail_mem endp

check proc near

    jae wrong
    mov dx, offset str_norm
    mov ah, 09h
    int 21h
    jmp end_check

wrong:

    mov di, offset str_norm - 4
    call WRD_TO_HEX
```

```
    mov dx, offset str_fall_code
    mov ah, 09h
    int 21h

end_check:

ret
check endp


BEGIN:


    call get_mem
    call check
    call free_mem
    call check

    call avail_mem

    mov dx, offset str_exp_mem
    mov ah, 09h
    int 21h

    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 bx,1h
     mul bx

    call print_number


    mov ah, 52h
    int 21h


    mov ax, es:[bx-2]
    mov es, ax; in es the address of first mcb

loop_list:
    mov ax, es:[1]

    call addr_psp


    mov dx, offset str_size_b
    mov ah, 09h
    int 21h

    mov ax, es:[3h]
    mov bx, 16
    mul bx
    call print_number

    mov dx, offset str_sequ
    mov ah, 09h
    int 21h

    xor di, di
    mov cx, 8h

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

```
    mov ah, 02h
    int 21h
    inc di
    loop loo_

    mov dx, offset str_div
    mov ah, 09h
    int 21h


    mov bx, es:[3h]
    mov al,es:[0h]
    cmp al, 5Ah
    je end_pr

    mov ax, es
     add ax,bx
     inc ax
     mov es,ax; in es the address of follow mcb

    jmp loop_list




end_pr:


    xor AL,AL
    mov AH,4Ch
    int 21H
end_this_code:

LAB3    ENDS
          END START
```