**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. 6381 |  | Сергухин В.Ю. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

## **Цель работы**

Исследование организации управления памятью, исследование структур данных и работы функций управления памятью ядра операционной системы.

**Постановка задачи:**

**Шаг 1.** Необходимо написать и отладить программный модуль типа .COM, 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.

**Шаг 2.** Далее необходимо изменить программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah прерывания 21h (пример в разделе «Использование функции 4АН»).

**Шаг 3.** Затем необходимо изменить программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48Н прерывания 21Н.

**Шаг 4.** Далее нужно изменить первоначальный вариант программы, запросив 64Кб памяти функцией 48Н прерывания 21Н до освобождения памяти.

**Шаг 5.** Оформить отчёт и ответить на контрольные вопросы.

**Необходимые сведения для составления программы:**

Учёт 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 этим 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 эту структуру можно получить, используя функцию f52h "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 30h, 31h 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,71h ; чтение младшего байта

mov BL,AL ; размера расширенной памяти

mov AL,31h ; запись адреса ячейки CMOS

out 70h,AL

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

## **Структура данных управляющей программы**

|  |  |
| --- | --- |
| **Название** | **Назначение** |
| AvailableMemory | Строка, содержащая «Amount of available memory: » |
| STRavailableMemory | Строка, содержащая количество доступной памяти |
| ExtendedMemory | Строка, содержащая «Size of extended memory: » |
| STRextendedMemory | Строка, содержащая размер расширенной памяти |
| ControlBlocks | Строка, содержащая «Chain of memory control blocks:» |
| INFOcontrolBlocks | Строка, содержащая «ADDRESS OWNER SIZE NAME» |
| STRcontrolBlocks | Строка, содержащая цепочку блоков управления памятью |
| STRerror | Строка, содержащая «Can`t free memory» |
| Error2 | Строка, содержащая «Error while allocating memory. Size of a free memory: » |
| STRerror2 | Строка, содержащая размер свободной памяти в 16 с.с. |
| FreeSpace | Строка, содержащая «0000h - FreeSpace» |
| BelongDriver | Строка, содержащая «0006h - Space belongs to the driver» |
| ExclusionDriver | Строка, содержащая «0007h - Space is the excluded upper memory of drivers » |
| BelingMsDos | Строка, содержащая «0008h - Space belongs MS DOS » |
| Occur386MAX | Строка, содержащая «FFFAh - Space is occupied by the control block 386MAX UMB» |
| Block386MAX | Строка, содержащая «FFFDh - Space blocks 386MAX » |
| Belong386MAX | Строка, содержащая «FFFEh - Space belongs 386MAX UMB » |
| FreeMemSucs | Строка, содержащая «Memory is free sucsesfull » |

## **Функции управляющей программы**

|  |  |
| --- | --- |
| **Название** | **Назначение** |
| TETR\_TO\_HEX | Перевод числа из 2-ой в 16-ую с.с. |
| BYTE\_TO\_HEX |
| WRD\_TO\_HEX |
| BYTE\_TO\_DEC | Перевод AL в 10 с/с |
| TWO\_BYTE\_TO\_DEC | Перевод DX:AX в 10 с/с |
| PRINT\_A\_STR | Вывод строки на экран |
| Available\_Mem | Определение и вывод на экран доступной памяти |
| Extended\_Mem | Определение и вывод на экран расширенной памяти |
| Control\_Blocks | Определение и вывод цепочки блоков управления памятью, печать необходимой информации |
| Free\_Mem | Освобождает память память, которую программа не занимает |

# **Ход работы**

1. Написан и отлажен **.COM** модуль, который определяет количество доступной памяти, размер расширенной памяти и выводит цепочку блоков управления памятью. Все доступные 648912 байт отдаются программе. Результаты выполнения программы:

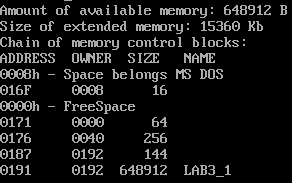


Рисунок 1 – Результаты выполнения программы LAB3\_1.com

1. Программа изменена таким образом, что она освобождает память, которую не занимает:

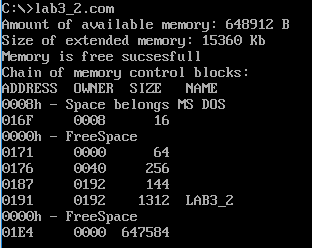


Рисунок 2 – Результаты выполнения программы LAB3\_2.com

1. Далее программа изменена так, что после освобождения памяти она запрашивает 64Кб (65536 байт) памяти функцией 48H прерывания 21H. Результаты:

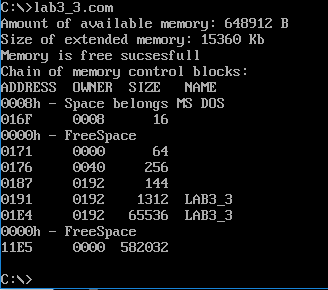


Рисунок 3 – Результаты выполнения программы LAB3\_3.com

1. Изменен первоначальный вариант программы, запрашивается 64Кб памяти, при этом происходит обработка завершения функций ядра с проверкой флага CF. Однако выдаётся ошибка, так как запрос памяти происходит в тот момент, когда вся доступная память занята программой. Затем происходит освобождение памяти, аналогично второму случаю.

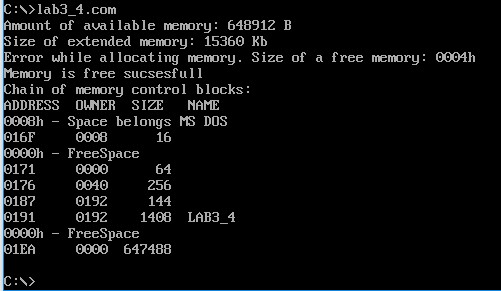


Рисунок 4 – Результаты выполнения программы LAB3\_4.com

1. Ответы на контрольные вопросы:
   1. Что означает «доступный объем памяти»?

Объем памяти, который предоставлен программе после ее запуска.

* 1. Где MCB блок Вашей программы в списке?

По адресу 0191h; В модуле, написанном в первом пункте хода работы, он находится в конце списка. Во втором – на втором месте с конца, поскольку после него располагается освобожденная память, в третьем – третий с конца, т.к. после освобождения памяти и дальнейшего запроса появляется дополнительный блок MCB с тем же адресом владельца и именем.

* 1. Какой размер памяти занимает программа в каждом случае?

1) 648912 B

2) 1312 В (произошло освобождение памяти)

3) 1312 + 65536 = 66848 B (после освобождения происходило выделение 64Кб)

4) 1408 B

**С учетом MCB и областью среды :**

1) 649088 ( 648912 +144 +16\*2) B

2) 1488 Bs

3) 67040 B

4) 1584 B

С учетом вычета MCB:

1) 648912-16= 648896 B

2) 1312-16= 1296 В

3) 66848-16-16= 66816 B

4) 1408-16= 1392 B

# **ЗАКЛЮЧЕНИЕ**

В ходе работы было проведено исследование структуры данных и работы функций управления памятью ядра операционной системы, а также рассмотрены нестраничная организация памяти и способы управления динамическими разделами.

**Приложения**

**Код программы Lab3\_1.asm**

TESTPC SEGMENT

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

ORG 100H

START:

jmp BEGIN

;ДАННЫЕ

AvailableMemory db 'Amount of available memory: $'

STRavailableMemory db '?????? B',10,13,'$'

ExtendedMemory db 'Size of extended memory: $'

STRextendedMemory db '????? Kb',10,13,'$'

ControlBlocks db 'Chain of memory control blocks:',10,13,'$'

INFOcontrolBlocks db 'ADDRESS OWNER SIZE NAME ',10,13,'$'

STRcontrolBlocks db ' ',10,13,'$'

STRerror db 'Error.',10,13,'$'

FreeSpace db '0000h - FreeSpace',10,13,'$'

BelongDriver db '0006h - Space belongs to the driver',10,13,'$'

ExclusionDriver db '0007h - Space is the excluded upper memory of drivers',10,13,'$'

BelingMsDos db '0008h - Space belongs MS DOS',10,13,'$'

Occur386MAX db 'FFFAh - Space is occupied by the control block 386MAX UMB',10,13,'$'

Block386MAX db 'FFFDh - Space blocks 386MAX ',10,13,'$'

Belong386MAX db 'FFFEh - Space belongs 386MAX UMB',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 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 10 с/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

;--------------------------------------------------

TWO\_BYTE\_TO\_DEC PROC near

;перевод в 10 с/с, SI-адрес поля младшей цифры

;DX:AX

push AX

push CX

push DX

mov CX,10

loop\_bd2: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd2

cmp AL,00h

je end\_l2

or AL,30h

mov [SI],AL

end\_l2: pop DX

pop CX

pop AX

ret

TWO\_BYTE\_TO\_DEC ENDP

;--------------------------------------------------

PRINT\_A\_STR PROC near

mov AH,09h

int 21h

ret

PRINT\_A\_STR ENDP

;--------------------------------------------------

; Количество доступной памяти

Available\_Mem PROC near

mov dx, offset AvailableMemory

call PRINT\_A\_STR

mov ah,4ah

mov bx,0ffffh ;Заведомо большая память

int 21h ;Получение объема доступной памяти в регистр bx

mov ax,bx

xor dx,dx

mov bx,10h

mul bx ;DX:AX = AX \* BX (умножаем на 16, параграф = 16 байт)

mov si, offset STRavailableMemory + 5

call TWO\_BYTE\_TO\_DEC

mov dx, offset STRavailableMemory

call PRINT\_A\_STR

ret

Available\_Mem ENDP

;--------------------------------------------------

; Размер расширенной памяти

Extended\_Mem PROC near

mov dx, offset ExtendedMemory

call PRINT\_A\_STR

mov al,30h ; запись адреса ячейки CMOS

out 70h, al

in al, 71h ; чтение младшего байта

mov bl,al ; размер расширенной памяти

mov al, 31h ; запись адреса ячейки CMOS

out 70h, al

in al, 71h ; чтение старшего байта

; размер расширенной памяти

mov ah, al

mov al, bl ; в ax - размер допустимой памяти

xor dx, dx

mov si, offset STRextendedMemory+4

call TWO\_BYTE\_TO\_DEC

mov dx, offset STRextendedMemory

call PRINT\_A\_STR

ret

Extended\_Mem ENDP

;--------------------------------------------------

; Цепочка блоков управления памятью

Control\_Blocks PROC near

mov dx, offset ControlBlocks

call PRINT\_A\_STR

mov dx, offset INFOcontrolBlocks

call PRINT\_A\_STR

mov ah, 52h

int 21h

mov ax, es:[bx-2]

mov es, ax

mov dx, es

xor bx, bx

Cycle:

push ax

push dx

push bx

push si

push es

push di

mov di,offset STRcontrolBlocks+3 ; ADDRESS

mov ax,es

call WRD\_TO\_HEX

mov di,offset STRcontrolBlocks+12 ; OWNER

mov ax,es:[01h]

cmp ax, 0000h

je Con1

cmp ax, 0006h

je Con2

cmp ax, 0007h

je Con3

cmp ax, 0008h

je Con4

cmp ax, 0FFFAh

je Con5

cmp ax, 0FFFDh

je Con6

cmp ax, 0FFFEh

je Con7

jmp Continion

Con1:

push ax

push dx

mov dx, offset FreeSpace

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con2:

push ax

push dx

mov dx, offset BelongDriver

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con3:

push ax

push dx

mov dx, offset ExclusionDriver

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con4:

push ax

push dx

mov dx, offset BelingMsDos

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con5:

push ax

push dx

mov dx, offset Occur386MAX

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con6:

push ax

push dx

mov dx, offset Block386MAX

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con7:

push ax

push dx

mov dx, offset Belong386MAX

call PRINT\_A\_STR

pop dx

pop ax

Continion:

call WRD\_TO\_HEX

mov si,offset STRcontrolBlocks+20 ; SIZE

mov ax,es:[03h]

mov bx,10h

mul bx

call TWO\_BYTE\_TO\_DEC

mov si,offset STRcontrolBlocks+23 ; NAME

mov bx, 08h

mov cx, 4

Cycle2:

mov ax, es:[bx]

mov [si], ax

add bx, 2h

add si, 2h

loop Cycle2

mov dx,offset STRcontrolBlocks

call PRINT\_A\_STR

pop di

pop es

pop si

pop bx

pop dx

pop ax

cmp byte ptr es:[00h],5Ah

je Exit

inc dx

add dx,es:[03h]

mov es,dx

jmp Cycle

Exit:

ret

Control\_Blocks ENDP

;--------------------------------------------------

;КОД

BEGIN:

call Available\_Mem

call Extended\_Mem

call Control\_Blocks

;Вывод в DOS

xor AL,AL

mov AH,4Ch

int 21H

TESTPC ENDS

END START ;конец модуля , START - точка входа

**Код программы Lab3\_2.asm**

TESTPC SEGMENT

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

ORG 100H

START:

jmp BEGIN

;ДАННЫЕ

AvailableMemory db 'Amount of available memory: $'

STRavailableMemory db '?????? B',10,13,'$'

ExtendedMemory db 'Size of extended memory: $'

STRextendedMemory db '????? Kb',10,13,'$'

ControlBlocks db 'Chain of memory control blocks:',10,13,'$'

INFOcontrolBlocks db 'ADDRESS OWNER SIZE NAME ',10,13,'$'

STRcontrolBlocks db ' ',10,13,'$'

STRerror db 'Cant free memory',10,13,'$'

FreeSpace db '0000h - FreeSpace',10,13,'$'

BelongDriver db '0006h - Space belongs to the driver',10,13,'$'

ExclusionDriver db '0007h - Space is the excluded upper memory of drivers',10,13,'$'

BelingMsDos db '0008h - Space belongs MS DOS',10,13,'$'

Occur386MAX db 'FFFAh - Space is occupied by the control block 386MAX UMB',10,13,'$'

Block386MAX db 'FFFDh - Space blocks 386MAX ',10,13,'$'

Belong386MAX db 'FFFEh - Space belongs 386MAX UMB',10,13,'$'

FreeMemSucs db 'Memory is free sucsesfull',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 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 10 с/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

;--------------------------------------------------

TWO\_BYTE\_TO\_DEC PROC near

;перевод в 10 с/с, SI-адрес поля младшей цифры

;DX:AX

push AX

push CX

push DX

mov CX,10

loop\_bd2: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd2

cmp AL,00h

je end\_l2

or AL,30h

mov [SI],AL

end\_l2: pop DX

pop CX

pop AX

ret

TWO\_BYTE\_TO\_DEC ENDP

;--------------------------------------------------

PRINT\_A\_STR PROC near

mov AH,09h

int 21h

ret

PRINT\_A\_STR ENDP

;--------------------------------------------------

; Количество доступной памяти

Available\_Mem PROC near

mov dx, offset AvailableMemory

call PRINT\_A\_STR

mov ah,4ah

mov bx,0ffffh ;Заведомо большая память

int 21h ;Получение объема доступной памяти в регистр bx

mov ax,bx

xor dx,dx

mov bx,10h

mul bx ;DX:AX = AX \* BX (умножаем на 16, параграф = 16 байт)

mov si, offset STRavailableMemory + 5

call TWO\_BYTE\_TO\_DEC

mov dx, offset STRavailableMemory

call PRINT\_A\_STR

ret

Available\_Mem ENDP

;--------------------------------------------------

; Размер расширенной памяти

Extended\_Mem PROC near

mov dx, offset ExtendedMemory

call PRINT\_A\_STR

mov al,30h ; запись адреса ячейки CMOS

out 70h, al

in al, 71h ; чтение младшего байта

mov bl,al ; размер расширенной памяти

mov al, 31h ; запись адреса ячейки CMOS

out 70h, al

in al, 71h ; чтение старшего байта

; размер расширенной памяти

mov ah, al

mov al, bl ; в ax - размер допустимой памяти

xor dx, dx

mov si, offset STRextendedMemory+4

call TWO\_BYTE\_TO\_DEC

mov dx, offset STRextendedMemory

call PRINT\_A\_STR

ret

Extended\_Mem ENDP

;--------------------------------------------------

; Освобождение памяти

Free\_Mem PROC near

push ax

push bx

push dx

mov ax, offset end\_lr3com

mov bx,10h

xor dx,dx

div bx

inc ax

mov bx,ax

mov ah,4Ah

int 21h

jnc success

mov dx, offset STRerror

call PRINT\_A\_STR

mov ah,4Ah

int 21h

success:

mov dx, offset FreeMemSucs

call PRINT\_A\_STR

pop dx

pop bx

pop ax

ret

Free\_Mem ENDP

;--------------------------------------------------

; Цепочка блоков управления памятью

Control\_Blocks PROC near

mov dx, offset ControlBlocks

call PRINT\_A\_STR

mov dx, offset INFOcontrolBlocks

call PRINT\_A\_STR

mov ah, 52h

int 21h

mov ax, es:[bx-2]

mov es, ax

mov dx, es

xor bx, bx

Cycle:

push ax

push dx

push bx

push si

push es

push di

mov di,offset STRcontrolBlocks+3 ; ADDRESS

mov ax,es

call WRD\_TO\_HEX

mov di,offset STRcontrolBlocks+12 ; OWNER

mov ax,es:[01h]

mov cx, 1

cmp ax, 0000h

je Con1

cmp ax, 0006h

je Con2

cmp ax, 0007h

je Con3

cmp ax, 0008h

je Con4

cmp ax, 0FFFAh

je Con5

cmp ax, 0FFFDh

je Con6

cmp ax, 0FFFEh

je Con7

jmp Continion

Con1:

mov cx, 0

push ax

push dx

mov dx, offset FreeSpace

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con2:

push ax

push dx

mov dx, offset BelongDriver

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con3:

push ax

push dx

mov dx, offset ExclusionDriver

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con4:

push ax

push dx

mov dx, offset BelingMsDos

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con5:

push ax

push dx

mov dx, offset Occur386MAX

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con6:

push ax

push dx

mov dx, offset Block386MAX

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con7:

push ax

push dx

mov dx, offset Belong386MAX

call PRINT\_A\_STR

pop dx

pop ax

Continion:

call WRD\_TO\_HEX

mov si, offset STRcontrolBlocks+20 ; SIZE

mov ax,es:[03h]

mov bx,10h

mul bx

call TWO\_BYTE\_TO\_DEC

cmp cx, 0

je ThroughputСycle

mov si, offset STRcontrolBlocks+23 ; NAME

mov cx, 4

mov bx, 08h

Cycle2:

mov ax, es:[bx]

mov [si], ax

add bx, 2h

add si, 2h

loop Cycle2

ThroughputСycle:

mov dx, offset STRcontrolBlocks

call PRINT\_A\_STR

mov al,' '

mov ah,' '

mov si,offset STRcontrolBlocks ; Чистка строки

mov [si+23],ax

mov [si+25],ax

mov [si+27],ax

mov [si+29],ax

pop di

pop es

pop si

pop bx

pop dx

pop ax

cmp byte ptr es:[00h],5Ah

je Exit

inc dx

add dx,es:[03h]

mov es,dx

jmp Cycle

Exit:

ret

Control\_Blocks ENDP

;--------------------------------------------------

;КОД

BEGIN:

call Available\_Mem

call Extended\_Mem

call Free\_Mem

call Control\_Blocks

;Вывод в DOS

xor AL,AL

mov AH,4Ch

int 21H

end\_lr3com:

TESTPC ENDS

END START ;конец модуля , START - точка входа

**Код программы Lab3\_3.asm**

TESTPC SEGMENT

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

ORG 100H

START:

jmp BEGIN

;ДАННЫЕ

AvailableMemory db 'Amount of available memory: $'

STRavailableMemory db '?????? B',10,13,'$'

ExtendedMemory db 'Size of extended memory: $'

STRextendedMemory db '????? Kb',10,13,'$'

ControlBlocks db 'Chain of memory control blocks:',10,13,'$'

INFOcontrolBlocks db 'ADDRESS OWNER SIZE NAME ',10,13,'$'

STRcontrolBlocks db ' ',10,13,'$'

STRerror db 'Cant free memory',10,13,'$'

FreeSpace db '0000h - FreeSpace',10,13,'$'

BelongDriver db '0006h - Space belongs to the driver',10,13,'$'

ExclusionDriver db '0007h - Space is the excluded upper memory of drivers',10,13,'$'

BelingMsDos db '0008h - Space belongs MS DOS',10,13,'$'

Occur386MAX db 'FFFAh - Space is occupied by the control block 386MAX UMB',10,13,'$'

Block386MAX db 'FFFDh - Space blocks 386MAX ',10,13,'$'

Belong386MAX db 'FFFEh - Space belongs 386MAX UMB',10,13,'$'

FreeMemSucs db 'Memory is free sucsesfull',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 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 10 с/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

;--------------------------------------------------

TWO\_BYTE\_TO\_DEC PROC near

;перевод в 10 с/с, SI-адрес поля младшей цифры

;DX:AX

push AX

push CX

push DX

mov CX,10

loop\_bd2: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd2

cmp AL,00h

je end\_l2

or AL,30h

mov [SI],AL

end\_l2: pop DX

pop CX

pop AX

ret

TWO\_BYTE\_TO\_DEC ENDP

;--------------------------------------------------

PRINT\_A\_STR PROC near

mov AH,09h

int 21h

ret

PRINT\_A\_STR ENDP

;--------------------------------------------------

; Количество доступной памяти

Available\_Mem PROC near

mov dx, offset AvailableMemory

call PRINT\_A\_STR

mov ah,4ah

mov bx,0ffffh ;Заведомо большая память

int 21h ;Получение объема доступной памяти в регистр bx

mov ax,bx

xor dx,dx

mov bx,10h

mul bx ;DX:AX = AX \* BX (умножаем на 16, параграф = 16 байт)

mov si, offset STRavailableMemory + 5

call TWO\_BYTE\_TO\_DEC

mov dx, offset STRavailableMemory

call PRINT\_A\_STR

ret

Available\_Mem ENDP

;--------------------------------------------------

; Размер расширенной памяти

Extended\_Mem PROC near

mov dx, offset ExtendedMemory

call PRINT\_A\_STR

mov al,30h ; запись адреса ячейки CMOS

out 70h, al

in al, 71h ; чтение младшего байта

mov bl,al ; размер расширенной памяти

mov al, 31h ; запись адреса ячейки CMOS

out 70h, al

in al, 71h ; чтение старшего байта

; размер расширенной памяти

mov ah, al

mov al, bl ; в ax - размер допустимой памяти

xor dx, dx

mov si, offset STRextendedMemory+4

call TWO\_BYTE\_TO\_DEC

mov dx, offset STRextendedMemory

call PRINT\_A\_STR

ret

Extended\_Mem ENDP

;--------------------------------------------------

; Освобождение памяти и запрос 64 Кб

Free\_Mem PROC near

push ax

push bx

push dx

mov ax, offset end\_lr3com

mov bx,10h

xor dx,dx

div bx

inc ax

mov bx,ax

mov ah,4Ah

int 21h

jnc success

mov dx, offset STRerror

call PRINT\_A\_STR

mov ah,4Ah

int 21h

success:

mov dx, offset FreeMemSucs

call PRINT\_A\_STR

mov bx,1000h ; Запрос 64 Кб

mov ah,48h

int 21h

pop dx

pop bx

pop ax

ret

Free\_Mem ENDP

;--------------------------------------------------

; Цепочка блоков управления памятью

Control\_Blocks PROC near

mov dx, offset ControlBlocks

call PRINT\_A\_STR

mov dx, offset INFOcontrolBlocks

call PRINT\_A\_STR

mov ah, 52h

int 21h

mov ax, es:[bx-2]

mov es, ax

mov dx, es

xor bx, bx

Cycle:

push ax

push dx

push bx

push si

push es

push di

mov di,offset STRcontrolBlocks+3 ; ADDRESS

mov ax,es

call WRD\_TO\_HEX

mov di,offset STRcontrolBlocks+12 ; OWNER

mov ax,es:[01h]

mov cx, 1

cmp ax, 0000h

je Con1

cmp ax, 0006h

je Con2

cmp ax, 0007h

je Con3

cmp ax, 0008h

je Con4

cmp ax, 0FFFAh

je Con5

cmp ax, 0FFFDh

je Con6

cmp ax, 0FFFEh

je Con7

jmp Continion

Con1:

mov cx, 0

push ax

push dx

mov dx, offset FreeSpace

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con2:

push ax

push dx

mov dx, offset BelongDriver

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con3:

push ax

push dx

mov dx, offset ExclusionDriver

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con4:

push ax

push dx

mov dx, offset BelingMsDos

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con5:

push ax

push dx

mov dx, offset Occur386MAX

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con6:

push ax

push dx

mov dx, offset Block386MAX

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con7:

push ax

push dx

mov dx, offset Belong386MAX

call PRINT\_A\_STR

pop dx

pop ax

Continion:

call WRD\_TO\_HEX

mov si, offset STRcontrolBlocks+20 ; SIZE

mov ax,es:[03h]

mov bx,10h

mul bx

call TWO\_BYTE\_TO\_DEC

cmp cx, 0

je ThroughputСycle

mov si, offset STRcontrolBlocks+23 ; NAME

mov cx, 4

mov bx, 08h

Cycle2:

mov ax, es:[bx]

mov [si], ax

add bx, 2h

add si, 2h

loop Cycle2

ThroughputСycle:

mov dx, offset STRcontrolBlocks

call PRINT\_A\_STR

mov al,' '

mov ah,' '

mov si,offset STRcontrolBlocks ; Чистка строки

mov [si+23],ax

mov [si+25],ax

mov [si+27],ax

mov [si+29],ax

pop di

pop es

pop si

pop bx

pop dx

pop ax

cmp byte ptr es:[00h],5Ah

je Exit

inc dx

add dx,es:[03h]

mov es,dx

jmp Cycle

Exit:

ret

Control\_Blocks ENDP

;--------------------------------------------------

;КОД

BEGIN:

call Available\_Mem

call Extended\_Mem

call Free\_Mem

call Control\_Blocks

;Вывод в DOS

xor AL,AL

mov AH,4Ch

int 21H

end\_lr3com:

TESTPC ENDS

END START ;конец модуля , START - точка входа

**Код программы Lab3\_4.asm**

TESTPC SEGMENT

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

ORG 100H

START:

jmp BEGIN

;ДАННЫЕ

AvailableMemory db 'Amount of available memory: $'

STRavailableMemory db '?????? B',10,13,'$'

ExtendedMemory db 'Size of extended memory: $'

STRextendedMemory db '????? Kb',10,13,'$'

ControlBlocks db 'Chain of memory control blocks:',10,13,'$'

INFOcontrolBlocks db 'ADDRESS OWNER SIZE NAME ',10,13,'$'

STRcontrolBlocks db ' ',10,13,'$'

STRerror db 'Cant free memory',10,13,'$'

FreeSpace db '0000h - FreeSpace',10,13,'$'

BelongDriver db '0006h - Space belongs to the driver',10,13,'$'

ExclusionDriver db '0007h - Space is the excluded upper memory of drivers',10,13,'$'

BelingMsDos db '0008h - Space belongs MS DOS',10,13,'$'

Occur386MAX db 'FFFAh - Space is occupied by the control block 386MAX UMB',10,13,'$'

Block386MAX db 'FFFDh - Space blocks 386MAX ',10,13,'$'

Belong386MAX db 'FFFEh - Space belongs 386MAX UMB',10,13,'$'

FreeMemSucs db 'Memory is free sucsesfull',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 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 10 с/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

;--------------------------------------------------

TWO\_BYTE\_TO\_DEC PROC near

;перевод в 10 с/с, SI-адрес поля младшей цифры

;DX:AX

push AX

push CX

push DX

mov CX,10

loop\_bd2: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd2

cmp AL,00h

je end\_l2

or AL,30h

mov [SI],AL

end\_l2: pop DX

pop CX

pop AX

ret

TWO\_BYTE\_TO\_DEC ENDP

;--------------------------------------------------

PRINT\_A\_STR PROC near

mov AH,09h

int 21h

ret

PRINT\_A\_STR ENDP

;--------------------------------------------------

; Количество доступной памяти

Available\_Mem PROC near

mov dx, offset AvailableMemory

call PRINT\_A\_STR

mov ah,4ah

mov bx,0ffffh ;Заведомо большая память

int 21h ;Получение объема доступной памяти в регистр bx

mov ax,bx

xor dx,dx

mov bx,10h

mul bx ;DX:AX = AX \* BX (умножаем на 16, параграф = 16 байт)

mov si, offset STRavailableMemory + 5

call TWO\_BYTE\_TO\_DEC

mov dx, offset STRavailableMemory

call PRINT\_A\_STR

ret

Available\_Mem ENDP

;--------------------------------------------------

; Размер расширенной памяти

Extended\_Mem PROC near

mov dx, offset ExtendedMemory

call PRINT\_A\_STR

mov al,30h ; запись адреса ячейки CMOS

out 70h, al

in al, 71h ; чтение младшего байта

mov bl,al ; размер расширенной памяти

mov al, 31h ; запись адреса ячейки CMOS

out 70h, al

in al, 71h ; чтение старшего байта

; размер расширенной памяти

mov ah, al

mov al, bl ; в ax - размер допустимой памяти

xor dx, dx

mov si, offset STRextendedMemory+4

call TWO\_BYTE\_TO\_DEC

mov dx, offset STRextendedMemory

call PRINT\_A\_STR

ret

Extended\_Mem ENDP

;--------------------------------------------------

; Освобождение памяти и запрос 64 Кб

Free\_Mem PROC near

push ax

push bx

push dx

mov ax, offset end\_lr3com

mov bx,10h

xor dx,dx

div bx

inc ax

mov bx,ax

mov ah,4Ah

int 21h

jnc success

mov dx, offset STRerror

call PRINT\_A\_STR

mov ah,4Ah

int 21h

success:

mov dx, offset FreeMemSucs

call PRINT\_A\_STR

mov bx,1000h ; Запрос 64 Кб

mov ah,48h

int 21h

pop dx

pop bx

pop ax

ret

Free\_Mem ENDP

;--------------------------------------------------

; Цепочка блоков управления памятью

Control\_Blocks PROC near

mov dx, offset ControlBlocks

call PRINT\_A\_STR

mov dx, offset INFOcontrolBlocks

call PRINT\_A\_STR

mov ah, 52h

int 21h

mov ax, es:[bx-2]

mov es, ax

mov dx, es

xor bx, bx

Cycle:

push ax

push dx

push bx

push si

push es

push di

mov di,offset STRcontrolBlocks+3 ; ADDRESS

mov ax,es

call WRD\_TO\_HEX

mov di,offset STRcontrolBlocks+12 ; OWNER

mov ax,es:[01h]

mov cx, 1

cmp ax, 0000h

je Con1

cmp ax, 0006h

je Con2

cmp ax, 0007h

je Con3

cmp ax, 0008h

je Con4

cmp ax, 0FFFAh

je Con5

cmp ax, 0FFFDh

je Con6

cmp ax, 0FFFEh

je Con7

jmp Continion

Con1:

mov cx, 0

push ax

push dx

mov dx, offset FreeSpace

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con2:

push ax

push dx

mov dx, offset BelongDriver

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con3:

push ax

push dx

mov dx, offset ExclusionDriver

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con4:

push ax

push dx

mov dx, offset BelingMsDos

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con5:

push ax

push dx

mov dx, offset Occur386MAX

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con6:

push ax

push dx

mov dx, offset Block386MAX

call PRINT\_A\_STR

pop dx

pop ax

jmp Continion

Con7:

push ax

push dx

mov dx, offset Belong386MAX

call PRINT\_A\_STR

pop dx

pop ax

Continion:

call WRD\_TO\_HEX

mov si, offset STRcontrolBlocks+20 ; SIZE

mov ax,es:[03h]

mov bx,10h

mul bx

call TWO\_BYTE\_TO\_DEC

cmp cx, 0

je ThroughputСycle

mov si, offset STRcontrolBlocks+23 ; NAME

mov cx, 4

mov bx, 08h

Cycle2:

mov ax, es:[bx]

mov [si], ax

add bx, 2h

add si, 2h

loop Cycle2

ThroughputСycle:

mov dx, offset STRcontrolBlocks

call PRINT\_A\_STR

mov al,' '

mov ah,' '

mov si,offset STRcontrolBlocks ; Чистка строки

mov [si+23],ax

mov [si+25],ax

mov [si+27],ax

mov [si+29],ax

pop di

pop es

pop si

pop bx

pop dx

pop ax

cmp byte ptr es:[00h],5Ah

je Exit

inc dx

add dx,es:[03h]

mov es,dx

jmp Cycle

Exit:

ret

Control\_Blocks ENDP

;--------------------------------------------------

;КОД

BEGIN:

call Available\_Mem

call Extended\_Mem

call Free\_Mem

call Control\_Blocks

;Вывод в DOS

xor AL,AL

mov AH,4Ch

int 21H

end\_lr3com:

TESTPC ENDS

END START ;конец модуля , START - точка входа