**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6381 |  | Лопатина А.С. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Постановка задачи**

**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.

**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 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

**Сведения о функциях и структурах данных управляющей программы.**

Сведения о функциях:

|  |  |
| --- | --- |
| Название | Назначение |
| TETR\_TO\_HEX | перевод полвины байта из 2-ой СС в 16-ую СС |
| BYTE\_TO\_HEX | перевод байта, расположенного в AL, в 2 символа 16-ой числа в AX |
| WRD\_TO\_HEX | перевод в 16-ую СС 16-ти разрядного числа |
| BYTE\_TO\_DEC | перевод в 10-ую СС |
| OUTPUT\_PROC | вывод на экран |
| DET\_AVAIL\_MEM | определение доступной памяти |
| DET\_EXT\_MEM | определение расширенной памяти |
| DET\_DATA | определение цепочки блоков управления памятью |
| OUTPUT\_DATA | вывод цепочки блоков управления памятью |

Сведения о структурах данных:

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| AVAIL\_MEM | db | Доступная память |
| Название | Тип | Назначение |
| EXT\_MEM | db | Расширенная память |
| TABLE\_HEAD | db | Заголовок таблицы |
| TABLE\_DATA | db | Данные таблицы |
| NEW\_LINE | db | Переход на новую строку |

**Последовательность действий, выполняемых программой.**

Модуль .СОМ выбирает и выводит на экран:

1. Количество доступной памяти;
2. Размер расширенной памяти;
3. Выводит цепочку блоков управления памятью;

Вторая программа при этом освобождает память, которую она не занимает. Третья программа после освобождения запрашивает 64 Кб памяти функцией 48H прерывания 21H. Четвертая программа является измененной первой программой, она запрашивает 64 Кб памяти функцией 48H прерывания 21H до освобождения памяти.

**Результат работы программы.**

1. Основная программа

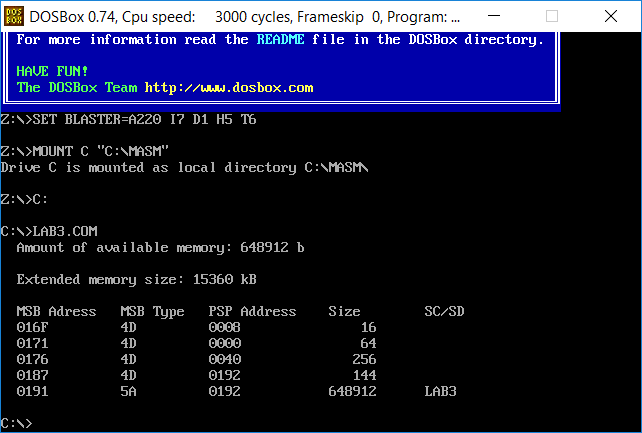


рис. 1

1. Программа, освобождающая память, которую она не занимает

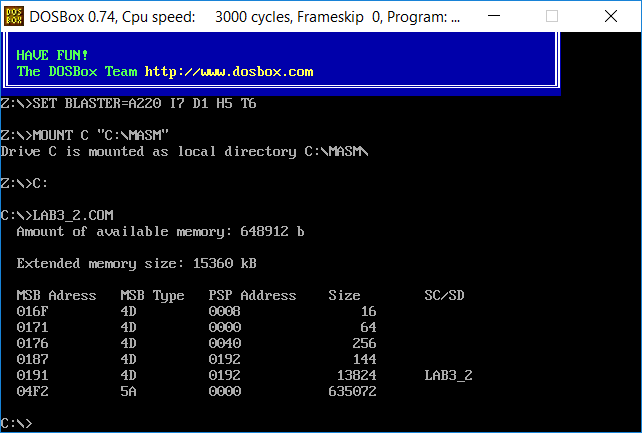


рис. 2

1. Программа, освобождающая память и после запрашивающая 64Кб памяти функцией 48Н прерывания 21Н

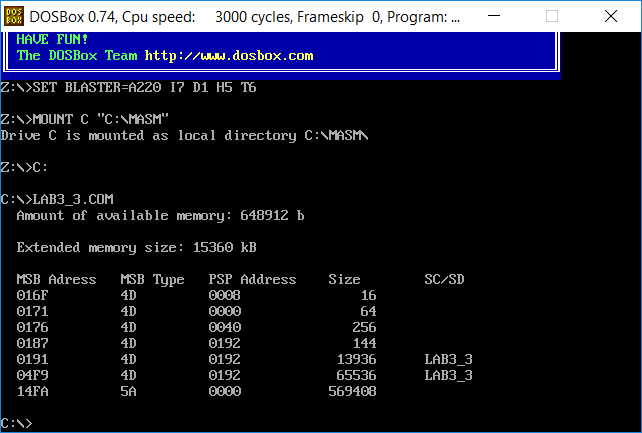


рис. 3

1. Программа, запрашивающая 64Кб памяти функцией 48Н прерывания 21Н и после освобождающая её

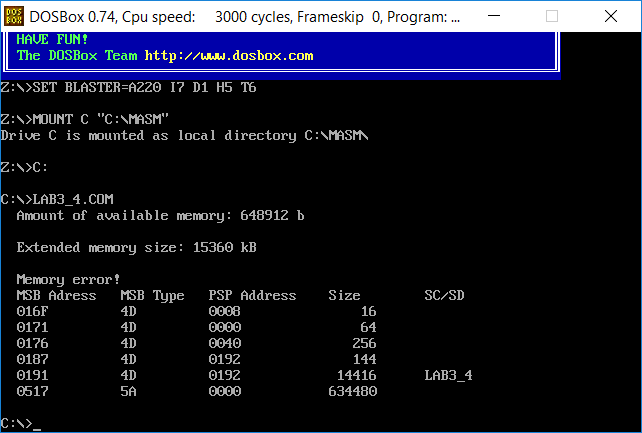


рис. 4

**Описание результатов исследования проблем, поставленных в лабораторной работе**

1. Что означает «доступный объём памяти»?

Ответ: «доступный объём памяти» - это объем оперативной памяти, выделенный программе для её выполнения;

1. Где MCB блок Вашей программы в списке?

Ответ: в первой программе блок программы последний и занимает всю доступную память. Во второй программе освобождается вся лишняя память, которая помещается в новый блок, отмеченный как свободный. Блок программы оказывается предпоследним, так как новый блок располагается после блока программы. В третьей блок программы второй (МСВ дополнительно выделенной памяти) и третий снизу. Программа сначала освобождает неиспользуемую память, а затем запрашивает 64 Кб памяти. А в четвертой программе запрос нового блока памяти посылается до того, как будет освобождена неиспользуемая память, однако так как запрашиваемый объем памяти выделен быть не может – выводится сообщение об ошибке;

1. Какой размер памяти занимает программа в каждом случае?

Ответ: 1. Всю доступную память - 648912 байт;

2. Необходимый программе объем памяти: 648912 – 635072 – 16 = 13824 байт;

3. 648912 – 569408 – 65536 – 2\*16 = 13936 байт;

4. 648912 – 634480 – 16 = 14416 байт;

**Заключение**

В результате выполнения лабораторной работы были рассмотрены нестраничная память и способ управления динамическими разделами, а также исследованы структуры данных и работа функций управления памятью ядра операционной системы.

Приложения

**Код программы** **LAB3**.**asm**

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;Данные

AVAIL\_MEM db ' Amount of available memory: b',0dh,0ah,'$'

EXT\_MEM db ' Extended memory size: kB',0dh,0ah,'$'

TABLE\_HEAD db ' MSB Adress MSB Type PSP Address Size SC/SD ', 0dh, 0ah, '$'

TABLE\_DATA db ' ', 0dh, 0ah, '$'

NEW\_LINE 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 ;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, в AH - младшая

pop cx

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

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

WRD\_TO\_DEC PROC near

push cx ; перевод двух байтов в 10 с/с, SI - адрес поля младшей цифры

push dx

push ax

mov cx,10

wrd\_loop\_bd:

div cx

or dl,30h

mov [si],dl

dec si

xor dx,dx

cmp ax,10

jae wrd\_loop\_bd

cmp ax,00h

jbe wrd\_end\_l

or al,30h

mov [si],al

wrd\_end\_l:

pop ax

pop dx

pop cx

ret

WRD\_TO\_DEC ENDP

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

OUTPUT\_PROC PROC NEAR ;Вывод на экран сообщения

push ax

mov ah, 09h

int 21h

pop ax

ret

OUTPUT\_PROC ENDP

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

DET\_AVAIL\_MEM PROC NEAR ; Определение доступной памяти

push ax

push bx

push dx

push si

xor ax, ax

mov ah, 04Ah

mov bx, 0FFFFh

int 21h

mov ax, 10h

mul bx

lea si, AVAIL\_MEM

add si, 23h

call WRD\_TO\_DEC

pop si

pop dx

pop bx

pop ax

ret

DET\_AVAIL\_MEM ENDP

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

DET\_EXT\_MEM PROC near ; Определение расширенной памяти

push ax

push bx

push si

push dx

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

mov ah, al

mov al, bl ; Хранение в AX размера расширенной памяти

sub dx, dx

lea si, EXT\_MEM

add si, 28

call WRD\_TO\_DEC

pop dx

pop si

pop bx

pop ax

ret

DET\_EXT\_MEM ENDP

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

DET\_DATA PROC near ;Определяем данные МСВ

lea di, TABLE\_DATA ; Адрес МСВ

mov ax, es

add di, 05h

call WRD\_TO\_HEX

lea di, TABLE\_DATA ; Тип МСВ

add di, 0Fh

xor ah, ah

mov al, es:[00h]

call BYTE\_TO\_HEX

mov [di], al

inc di

mov [di], ah

lea di, TABLE\_DATA ; Адрес PSP

mov ax, es:[01h]

add di, 1Dh

call WRD\_TO\_HEX

lea di, TABLE\_DATA ; Размер

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 2Eh

push si

mov si, di

call WRD\_TO\_DEC

pop si

lea di, TABLE\_DATA ;SC/SD

add di, 35h

mov bx, 0h

print:

mov dl, es:[bx + 8]

mov [di], dl

inc di

inc bx

cmp bx, 8h

jne print

mov ax, es:[3h]

mov bl, es:[0h]

ret

DET\_DATA ENDP

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

OUTPUT\_DATA PROC NEAR ; Определяем и выводим цепочку блоков управления памятью

mov ah, 52h

int 21h

sub bx, 2h

mov es, es:[bx]

output:

call DET\_DATA

lea dx, TABLE\_DATA

call OUTPUT\_PROC

mov cx, es

add ax, cx

inc ax

mov es, ax

cmp bl, 4Dh

je output

ret

OUTPUT\_DATA ENDP

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

BEGIN:

call DET\_AVAIL\_MEM ;Вызываем функцию определения доступной памяти

lea dx,AVAIL\_MEM

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

call DET\_EXT\_MEM ;Вызываем функцию определения расширенной памяти

lea dx,EXT\_MEM

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

lea dx, TABLE\_HEAD

call OUTPUT\_PROC

call OUTPUT\_DATA

;Выход в DOS

xor al, al

mov ah, 4ch

int 21h

TESTPC ENDS

END START

**Код программы** **LAB3\_2**.**asm**

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;Данные

AVAIL\_MEM db ' Amount of available memory: b',0dh,0ah,'$'

EXT\_MEM db ' Extended memory size: kB',0dh,0ah,'$'

TABLE\_HEAD db ' MSB Adress MSB Type PSP Address Size SC/SD ', 0dh, 0ah, '$'

TABLE\_DATA db ' ', 0dh, 0ah, '$'

NEW\_LINE 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 ;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, в AH - младшая

pop cx

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

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

WRD\_TO\_DEC PROC near

push cx ; перевод двух байтов в 10 с/с, SI - адрес поля младшей цифры

push dx

push ax

mov cx,10

wrd\_loop\_bd:

div cx

or dl,30h

mov [si],dl

dec si

xor dx,dx

cmp ax,10

jae wrd\_loop\_bd

cmp ax,00h

jbe wrd\_end\_l

or al,30h

mov [si],al

wrd\_end\_l:

pop ax

pop dx

pop cx

ret

WRD\_TO\_DEC ENDP

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

OUTPUT\_PROC PROC NEAR ;Вывод на экран сообщения

push ax

mov ah, 09h

int 21h

pop ax

ret

OUTPUT\_PROC ENDP

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

DET\_AVAIL\_MEM PROC NEAR ; Определение доступной памяти

push ax

push bx

push dx

push si

xor ax, ax

mov ah, 04Ah

mov bx, 0FFFFh

int 21h

mov ax, 10h

mul bx

lea si, AVAIL\_MEM

add si, 23h

call WRD\_TO\_DEC

pop si

pop dx

pop bx

pop ax

ret

DET\_AVAIL\_MEM ENDP

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

DET\_EXT\_MEM PROC near ; Определение расширенной памяти

push ax

push bx

push si

push dx

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

mov ah, al

mov al, bl ; Хранение в AX размера расширенной памяти

sub dx, dx

lea si, EXT\_MEM

add si, 28

call WRD\_TO\_DEC

pop dx

pop si

pop bx

pop ax

ret

DET\_EXT\_MEM ENDP

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

DET\_DATA PROC near ;Определяем данные МСВ

lea di, TABLE\_DATA ; Адрес МСВ

mov ax, es

add di, 05h

call WRD\_TO\_HEX

lea di, TABLE\_DATA ; Тип МСВ

add di, 0Fh

xor ah, ah

mov al, es:[00h]

call BYTE\_TO\_HEX

mov [di], al

inc di

mov [di], ah

lea di, TABLE\_DATA ; Адрес PSP

mov ax, es:[01h]

add di, 1Dh

call WRD\_TO\_HEX

lea di, TABLE\_DATA ; Размер

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 2Eh

push si

mov si, di

call WRD\_TO\_DEC

pop si

lea di, TABLE\_DATA ;SC/SD

add di, 35h

mov bx, 0h

print:

mov dl, es:[bx + 8]

mov [di], dl

inc di

inc bx

cmp bx, 8h

jne print

mov ax, es:[3h]

mov bl, es:[0h]

ret

DET\_DATA ENDP

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

OUTPUT\_DATA PROC NEAR ; Определяем и выводим цепочку блоков управления памятью

mov ah, 52h

int 21h

sub bx, 2h

mov es, es:[bx]

output:

call DET\_DATA

lea dx, TABLE\_DATA

call OUTPUT\_PROC

mov cx, es

add ax, cx

inc ax

mov es, ax

cmp bl, 4Dh

je output

ret

OUTPUT\_DATA ENDP

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

BEGIN:

call DET\_AVAIL\_MEM ;Вызываем функцию определения доступной памяти

lea dx,AVAIL\_MEM

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

call DET\_EXT\_MEM ;Вызываем функцию определения расширенной памяти

lea dx,EXT\_MEM

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

mov ah, 4ah ;Освобождаем память

lea bx, END\_PROG

int 21h

lea dx, TABLE\_HEAD

call OUTPUT\_PROC

call OUTPUT\_DATA

;Выход в DOS

xor al, al

mov ah, 4ch

int 21h

END\_PROG db 0

TESTPC ENDS

END START

**Код программы** **LAB3\_3**.**asm**

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;Данные

AVAIL\_MEM db ' Amount of available memory: b',0dh,0ah,'$'

EXT\_MEM db ' Extended memory size: kB',0dh,0ah,'$'

TABLE\_HEAD db ' MSB Adress MSB Type PSP Address Size SC/SD ', 0dh, 0ah, '$'

TABLE\_DATA db ' ', 0dh, 0ah, '$'

NEW\_LINE 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 ;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, в AH - младшая

pop cx

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

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

WRD\_TO\_DEC PROC near

push cx ; перевод двух байтов в 10 с/с, SI - адрес поля младшей цифры

push dx

push ax

mov cx,10

wrd\_loop\_bd:

div cx

or dl,30h

mov [si],dl

dec si

xor dx,dx

cmp ax,10

jae wrd\_loop\_bd

cmp ax,00h

jbe wrd\_end\_l

or al,30h

mov [si],al

wrd\_end\_l:

pop ax

pop dx

pop cx

ret

WRD\_TO\_DEC ENDP

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

OUTPUT\_PROC PROC NEAR ;Вывод на экран сообщения

push ax

mov ah, 09h

int 21h

pop ax

ret

OUTPUT\_PROC ENDP

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

DET\_AVAIL\_MEM PROC NEAR ; Определение доступной памяти

push ax

push bx

push dx

push si

xor ax, ax

mov ah, 04Ah

mov bx, 0FFFFh

int 21h

mov ax, 10h

mul bx

lea si, AVAIL\_MEM

add si, 23h

call WRD\_TO\_DEC

pop si

pop dx

pop bx

pop ax

ret

DET\_AVAIL\_MEM ENDP

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

DET\_EXT\_MEM PROC near ; Определение расширенной памяти

push ax

push bx

push si

push dx

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

mov ah, al

mov al, bl ; Хранение в AX размера расширенной памяти

sub dx, dx

lea si, EXT\_MEM

add si, 28

call WRD\_TO\_DEC

pop dx

pop si

pop bx

pop ax

ret

DET\_EXT\_MEM ENDP

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

DET\_DATA PROC near ;Определяем данные МСВ

lea di, TABLE\_DATA ; Адрес МСВ

mov ax, es

add di, 05h

call WRD\_TO\_HEX

lea di, TABLE\_DATA ; Тип МСВ

add di, 0Fh

xor ah, ah

mov al, es:[00h]

call BYTE\_TO\_HEX

mov [di], al

inc di

mov [di], ah

lea di, TABLE\_DATA ; Адрес PSP

mov ax, es:[01h]

add di, 1Dh

call WRD\_TO\_HEX

lea di, TABLE\_DATA ; Размер

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 2Eh

push si

mov si, di

call WRD\_TO\_DEC

pop si

lea di, TABLE\_DATA ;SC/SD

add di, 35h

mov bx, 0h

print:

mov dl, es:[bx + 8]

mov [di], dl

inc di

inc bx

cmp bx, 8h

jne print

mov ax, es:[3h]

mov bl, es:[0h]

ret

DET\_DATA ENDP

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

OUTPUT\_DATA PROC NEAR ; Определяем и выводим цепочку блоков управления памятью

mov ah, 52h

int 21h

sub bx, 2h

mov es, es:[bx]

output:

call DET\_DATA

lea dx, TABLE\_DATA

call OUTPUT\_PROC

mov cx, es

add ax, cx

inc ax

mov es, ax

cmp bl, 4Dh

je output

ret

OUTPUT\_DATA ENDP

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

BEGIN:

call DET\_AVAIL\_MEM ;Вызываем функцию определения доступной памяти

lea dx,AVAIL\_MEM

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

call DET\_EXT\_MEM ;Вызываем функцию определения расширенной памяти

lea dx,EXT\_MEM

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

mov ah, 4ah ;Освобождаем память

lea bx, END\_PROG

int 21h

mov ah, 48h ;Запрашиваем 64Кб памяти

mov bx, 1000h

int 21h

lea dx, TABLE\_HEAD

call OUTPUT\_PROC

call OUTPUT\_DATA

;Выход в DOS

xor al, al

mov ah, 4ch

int 21h

END\_PROG db 0

TESTPC ENDS

END START

**Код программы** **LAB3\_4**.**asm**

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;Данные

AVAIL\_MEM db ' Amount of available memory: b',0dh,0ah,'$'

EXT\_MEM db ' Extended memory size: kB',0dh,0ah,'$'

TABLE\_HEAD db ' MSB Adress MSB Type PSP Address Size SC/SD ', 0dh, 0ah, '$'

TABLE\_DATA db ' ', 0dh, 0ah, '$'

NEW\_LINE db ' ',0dh,0ah,'$'

ERROR\_M db ' Memory error!', 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 ;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, в AH - младшая

pop cx

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

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

WRD\_TO\_DEC PROC near

push cx ; перевод двух байтов в 10 с/с, SI - адрес поля младшей цифры

push dx

push ax

mov cx,10

wrd\_loop\_bd:

div cx

or dl,30h

mov [si],dl

dec si

xor dx,dx

cmp ax,10

jae wrd\_loop\_bd

cmp ax,00h

jbe wrd\_end\_l

or al,30h

mov [si],al

wrd\_end\_l:

pop ax

pop dx

pop cx

ret

WRD\_TO\_DEC ENDP

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

OUTPUT\_PROC PROC NEAR ;Вывод на экран сообщения

push ax

mov ah, 09h

int 21h

pop ax

ret

OUTPUT\_PROC ENDP

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

DET\_AVAIL\_MEM PROC NEAR ; Определение доступной памяти

push ax

push bx

push dx

push si

xor ax, ax

mov ah, 04Ah

mov bx, 0FFFFh

int 21h

mov ax, 10h

mul bx

lea si, AVAIL\_MEM

add si, 23h

call WRD\_TO\_DEC

pop si

pop dx

pop bx

pop ax

ret

DET\_AVAIL\_MEM ENDP

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

DET\_EXT\_MEM PROC near ; Определение расширенной памяти

push ax

push bx

push si

push dx

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

mov ah, al

mov al, bl ; Хранение в AX размера расширенной памяти

sub dx, dx

lea si, EXT\_MEM

add si, 28

call WRD\_TO\_DEC

pop dx

pop si

pop bx

pop ax

ret

DET\_EXT\_MEM ENDP

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

DET\_DATA PROC near ;Определяем данные МСВ

lea di, TABLE\_DATA ; Адрес МСВ

mov ax, es

add di, 05h

call WRD\_TO\_HEX

lea di, TABLE\_DATA ; Тип МСВ

add di, 0Fh

xor ah, ah

mov al, es:[00h]

call BYTE\_TO\_HEX

mov [di], al

inc di

mov [di], ah

lea di, TABLE\_DATA ; Адрес PSP

mov ax, es:[01h]

add di, 1Dh

call WRD\_TO\_HEX

lea di, TABLE\_DATA ; Размер

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 2Eh

push si

mov si, di

call WRD\_TO\_DEC

pop si

lea di, TABLE\_DATA ;SC/SD

add di, 35h

mov bx, 0h

print:

mov dl, es:[bx + 8]

mov [di], dl

inc di

inc bx

cmp bx, 8h

jne print

mov ax, es:[3h]

mov bl, es:[0h]

ret

DET\_DATA ENDP

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

OUTPUT\_DATA PROC NEAR ; Определяем и выводим цепочку блоков управления памятью

mov ah, 52h

int 21h

sub bx, 2h

mov es, es:[bx]

output:

call DET\_DATA

lea dx, TABLE\_DATA

call OUTPUT\_PROC

mov cx, es

add ax, cx

inc ax

mov es, ax

cmp bl, 4Dh

je output

ret

OUTPUT\_DATA ENDP

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

BEGIN:

call DET\_AVAIL\_MEM ;Вызываем функцию определения доступной памяти

lea dx,AVAIL\_MEM

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

call DET\_EXT\_MEM ;Вызываем функцию определения расширенной памяти

lea dx,EXT\_MEM

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

mov ah, 48h ;Запрашиваем 64Кб

mov bx, 1000h

int 21h

jc error\_mem ;Проверяем, не произошла ли ошибка памяти

jmp all\_good

error\_mem: ; Выводим сообщение об ошибке

lea dx, ERROR\_M

call OUTPUT\_PROC

all\_good: ;Освобождаем память

mov ah, 4ah

lea bx, END\_PROG

int 21h

lea dx, TABLE\_HEAD

call OUTPUT\_PROC

call OUTPUT\_DATA

;Выход в DOS

xor al, al

mov ah, 4ch

int 21h

END\_PROG db 0

TESTPC ENDS

END START