**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра MOЭBM**

отчет

**по лабораторной работе №3**

**по дисциплине «Исследование организации управления основной памятью»**

**Тема: Исследование интерфейсов программных модулей**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Вергейчик Г. Л. |
| Преподаватель |  | Губкин А. Ф. |

Санкт-Петербург

2018

**Цель работы**

Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованный в компьютере и способ организации, принятый в ОС. В лабораторной работе рассматривается не страничная память и способ управления динамическими разделами. Для реализации управления памятью в этом случае строится список занятых и свободных участков памяти. Функции ядра, обеспечивающие управление основной памятью, просматривают и преобразуют этот список.

**Дополнительные сведения**

Учет занятой и свободной памяти ведется при помощи списка блоков управления памятью MCB и занимает 16 байт. Располагается всегда с адреса кратного 16 и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет.

Формат MCB:

* Смещение 00h – тип MCB
* Смещение 01h – сегментный адрес PSP владельца участка памяти
* Смещение 03h – размер участка памяти в параграфах
* Смещение 05h – зарезервирован
* Смещение 08h – SC или SD

**Ход работы**

1. Был написан файл исходного .COM модуля, определяющего количество доступной памяти, размер расширенной памяти. Модуль так же должен выводить цепочку блоков управления памятью.
2. Добавили в программу освобождения памяти, которую она не занимает.
3. Запросили 64 кбайт памяти и освободили их.
4. Освободили память и запросили 64 кбайт памяти.

**Функции программы**

* Функция TETR\_TO\_HEX переводит половину байта в символ.
* Функция BYTE\_TO\_HEX переводит байт в AL в два символа в шестн. числа в AX.
* Функция WRD\_TO\_HEX переводит в 16 с/с 16-ти разрядное число в AX - число, DI - адрес последнего символа.
* Функция BYTE\_TO\_DEC переводит в 10 с/с, SI - адрес поля младшей цифры.

Функция WRD\_TO\_DEC определяет сегментный адрес среды передаваемой программе

* GET\_MEMORY определяет и выводит на экран размер доступной памяти
* GET\_EXT\_MEMORY определяет и выводит на экран размер расширенной памяти
* GET\_TAIL определяет восьмибайтное окончание блока
* GET\_MCB определяет и выводит на экран характеристики блоков управления памятью

**Результаты выполнения и отладки программ:**

* **Lab3\_1(шаг 1)**

**Результаты отладки программы**(см. Рис1):

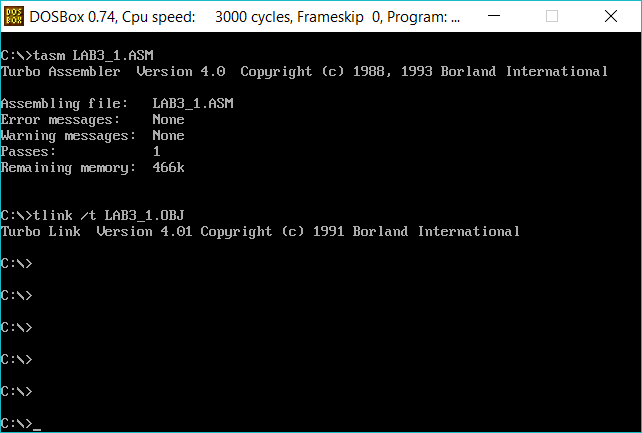


Рис1. Отладка lab3\_1

**Результат работы программы**(см. Рис2):

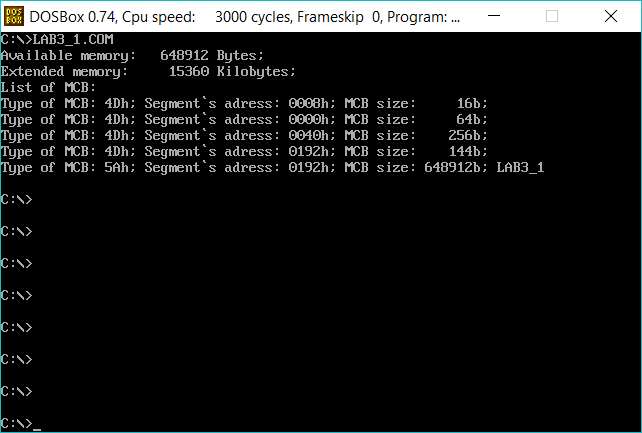


Рис2. Результат работы lab3\_1

* **Lab3\_2(шаг 2)**

**Результаты отладки программы**(см. Рис3):

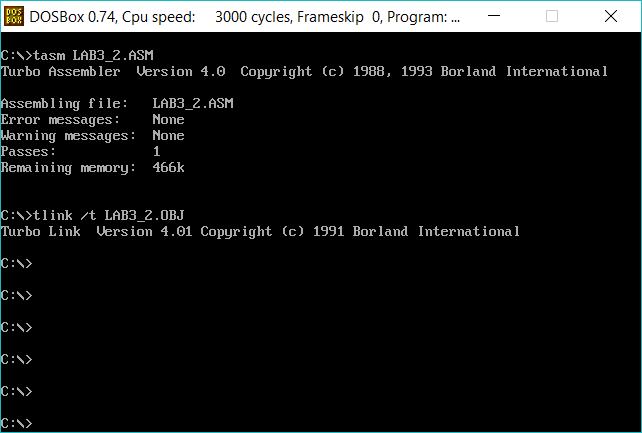


Рис3. Отладка lab3\_2

**Результат работы программы**(см. Рис4):

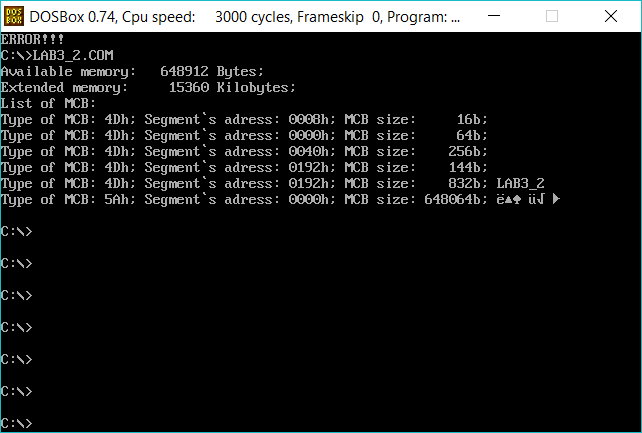


Рис4. Результат работы lab3\_2

* **Lab3\_3(шаг 3)**

**Результаты отладки программы**(см. Рис5):

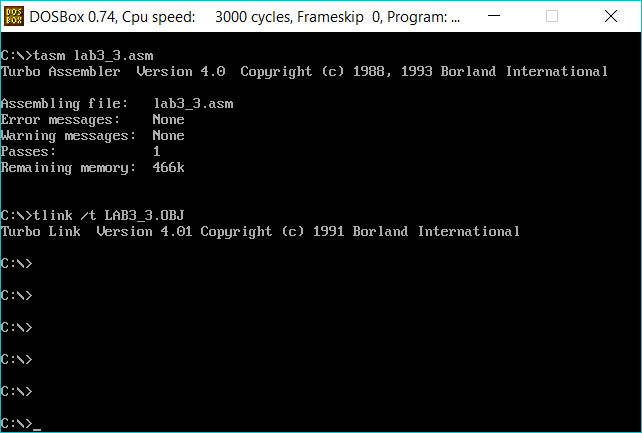


Рис5. Отладка lab3\_3

**Результат работы программы**(см. Рис6):

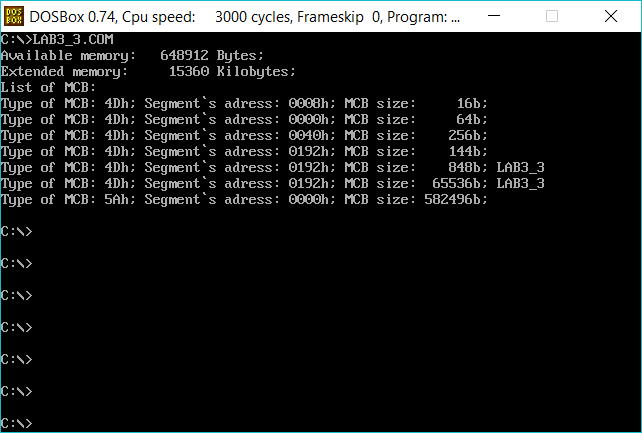


Рис6. Результат работы lab3\_3

* **Lab3\_4(шаг 4)**

**Результаты отладки программы**(см. Рис7):

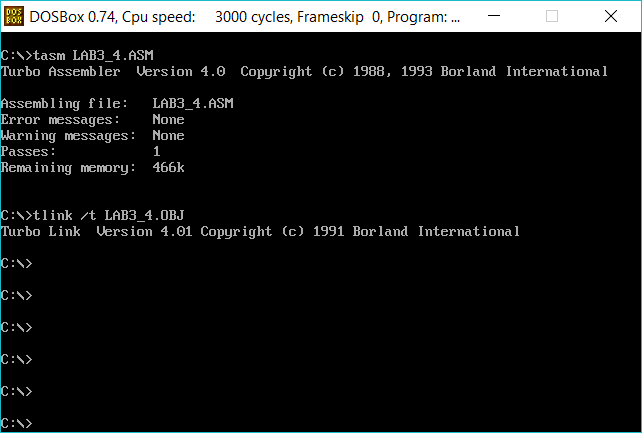


Рис7. Отладка lab3\_4

**Результат работы программы**(см. Рис8):

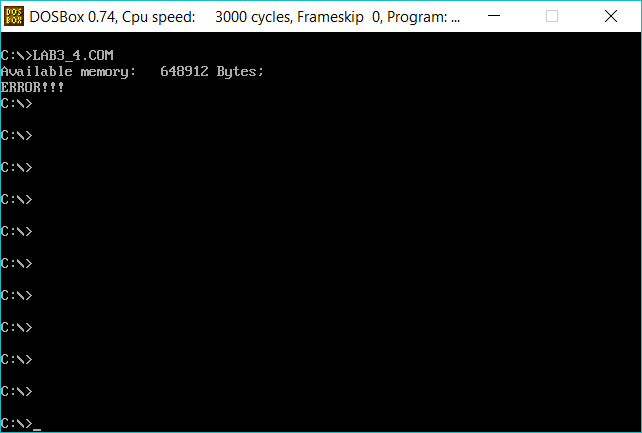


Рис8. Результат работы lab3\_4

**Ответы на вопросы**

**Сегментный адрес недоступной памяти**

1. **Что означает «доступный объем памяти»?**  
   Это часть оперативной памяти, занимаемой и используемой программой.
2. **Где MCB блок Вашей программы в списке?**

В первом случае блок программы последний и занимает всю доступную память.

Во втором случае блок программы второй снизу. Это связано с тем, что программа освобождает неиспользуемую память, и блок с неиспользуемой ей памятью оказывается последним.

В третьем случае блок программы третий снизу. Программа сначала освобождает неиспользуемую память, а затем запрашивает 64 Кб памяти.

В четвёртом случае программа запрашивает память до того, как освобождает неиспользуемую – и при обработке завершения функций ядра (проверке флага CF) мы узнаём, что возникает ошибка.

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

В первом случае – 648912 б.

Во втором – 832 б.

В третьем – 66400 б.

В четвертом - 648912 б, далее происходит ошибка при попытке выделить больше.

**Вывод**

В ходе выполнения данной лабораторной работы была исследована структура MCB, а также исследован процесс работы с памятью.

**Приложение**

1. Файл lab3\_1.asm

LR3\_1 SEGMENT

ASSUME CS:LR3\_1, DS:LR3\_1, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

av\_memory db 'Available memory: Bytes;',0dh,0ah,'$'

ext\_memory db 'Extended memory: Kilobytes;',0dh,0ah,'$'

MCB\_Capt db 'List of MCB:',0dh,0ah,'$'

MCB\_Type db 'Type of MCB: h; $'

MCB\_Seg db 'Segment`s adress: h; $'

MCB\_Size db 'MCB size: b$'

MCB\_Tail db '; ',0dh,0ah,'$'

Write\_msg PROC near

mov ah,09h

int 21h

ret

Write\_msg ENDP

; ‘ункци¤ перевода из 2 с/с в 16 половины байта

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 переводитс¤ 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 старша¤ цифра

pop cx ; в AH младша¤

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near

; первод в 16 с/с 16-ти разр¤дного 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 одного байта в 10 с/с, SI - адрес пол¤ младшей цифры

push cx

push dx

push ax

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 ax,00h

jbe end\_l

or al,30h

mov [si],al

end\_l:

pop ax

pop dx

pop cx

ret

BYTE\_TO\_DEC ENDP

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

WRD\_TO\_DEC PROC near

; перевод 2 байтов в 10 с/с, SI - адрес пол¤ младшей цифры

push cx

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

; ќпределение доступной пам¤ти

GET\_MEMORY PROC near

push ax

push bx

push cx

push dx

mov bx, 0ffffh

mov ah, 4Ah

int 21h

mov ax, bx

mov cx, 10h

mul cx

lea si, av\_memory+25

call WRD\_TO\_DEC

lea dx, av\_memory

call Write\_msg

pop dx

pop cx

pop bx

pop ax

ret

GET\_MEMORY ENDP

; ќпределение расширенной пам¤ти

GET\_EXT\_MEMORY PROC near

push ax

push bx

push si

push dx

mov al, 30h ; запись адреса ¤чейки CMOS

out 70h, al

in al, 71h ; чтение младшего байта

mov bl, al ; размера расширенной пам¤ти

mov al, 31h ; запись адреса ¤чейки CMOS

out 70h, al

in al, 71h ; чтение старшего байта размера расширенной пам¤ти

mov ah, al

mov al, bl ; теперь в AX размер расширенной пам¤ти

sub dx, dx

lea si, ext\_memory+25

call WRD\_TO\_DEC

lea dx, ext\_memory

call write\_msg

pop dx

pop si

pop bx

pop ax

ret

GET\_EXT\_MEMORY ENDP

; ѕолучаем восьмибайтное окончание блока

GET\_TAIL PROC near

push si

push cx

push bx

push ax

mov bx,0008h

mov cx,4

RE:

mov ax,es:[bx]

mov [si],ax

add bx,2h

add si,2h

loop RE

pop ax

pop bx

pop cx

pop si

ret

GET\_TAIL ENDP

; ќпредел¤ем и выводим цепочку блоков управлени¤ пам¤тью

GET\_MCB PROC near

push ax

push bx

push cx

push dx

lea dx, MCB\_Capt

call Write\_msg

mov ah,52h

int 21h

mov es,es:[bx-2]

mov bx,1

REPEAT:

sub ax,ax

sub cx,cx

sub di,di

sub si,si

mov al,es:[0000h]

call BYTE\_TO\_HEX

lea di,MCB\_Type+13

mov [di],ax

cmp ax,4135h

je MEN\_BX

STEP:

lea di,MCB\_Seg+21

mov ax,es:[0001h]

call WRD\_TO\_HEX

mov ax,es:[0003h]

mov cx,10h

mul cx

lea si,MCB\_Size+15

call WRD\_TO\_DEC

lea dx,MCB\_Type

call Write\_msg

lea dx,MCB\_Seg

call Write\_msg

lea dx,MCB\_Size

call Write\_msg

lea si,MCB\_Tail+2

call GET\_TAIl

lea dx,MCB\_Tail

call Write\_msg

cmp bx,0

jz END\_P

xor ax, ax

mov ax, es

add ax, es:[0003h]

inc ax

mov es, ax

jmp REPEAT

END\_P:

pop dx

pop cx

pop bx

pop ax

ret

MEN\_BX:

mov bx,0

jmp STEP

BEGIN:

; ¬ыполнение основного задани¤

call GET\_MEMORY

call GET\_EXT\_MEMORY

call GET\_MCB

; выход в DOS

xor al,al

mov ah,3Ch

int 21h

ret

LR3\_1 ENDS

END START

1. Файл lab3\_2.asm

;—≈√ћ≈Ќ“џ

LR3\_1 SEGMENT

ASSUME CS:LR3\_1, DS:LR3\_1, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

;ѕ≈–≈ћ≈ЌЌџ≈

av\_memory db 'Available memory: Bytes;',0dh,0ah,'$'

ext\_memory db 'Extended memory: Kilobytes;',0dh,0ah,'$'

MCB\_Capt db 'List of MCB:',0dh,0ah,'$'

MCB\_Type db 'Type of MCB: h; $'

MCB\_Seg db 'Segment`s adress: h; $'

MCB\_Size db 'MCB size: b$'

MCB\_Tail db '; ',0dh,0ah,'$'

;ѕ–ќ÷≈ƒ”–џ

; ¬ывод на экран

Write\_msg PROC near

mov ah,09h

int 21h

ret

Write\_msg ENDP

; ‘ункци¤ перевода из 2 с/с в 16 половины байта

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 переводитс¤ 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 старша¤ цифра

pop cx ; в AH младша¤

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near

; первод в 16 с/с 16-ти разр¤дного 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 одного байта в 10 с/с, SI - адрес пол¤ младшей цифры

push cx

push dx

push ax

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 ax,00h

jbe end\_l

or al,30h

mov [si],al

end\_l:

pop ax

pop dx

pop cx

ret

BYTE\_TO\_DEC ENDP

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

WRD\_TO\_DEC PROC near

; перевод 2 байтов в 10 с/с, SI - адрес пол¤ младшей цифры

push cx

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

; ќпределение доступной пам¤ти

GET\_MEMORY PROC near

push ax

push bx

push cx

push dx

mov bx, 0ffffh

mov ah, 4Ah

int 21h

mov ax, bx

mov cx, 10h

mul cx

lea si, av\_memory+25

call WRD\_TO\_DEC

lea dx, av\_memory

call Write\_msg

lea ax, END\_S

mov bx, 10h

sub dx, dx

div bx

inc ax

mov bx, ax

mov al, 0

mov ah, 4Ah

int 21h

pop dx

pop cx

pop bx

pop ax

ret

GET\_MEMORY ENDP

; ќпределение расширенной пам¤ти

GET\_EXT\_MEMORY PROC near

push ax

push bx

push si

push dx

mov al, 30h ; запись адреса ¤чейки CMOS

out 70h, al

in al, 71h ; чтение младшего байта

mov bl, al ; размера расширенной пам¤ти

mov al, 31h ; запись адреса ¤чейки CMOS

out 70h, al

in al, 71h ; чтение старшего байта размера расширенной пам¤ти

mov ah, al

mov al, bl ; теперь в AX размер расширенной пам¤ти

sub dx, dx

lea si, ext\_memory+25

call WRD\_TO\_DEC

lea dx, ext\_memory

call write\_msg

pop dx

pop si

pop bx

pop ax

ret

GET\_EXT\_MEMORY ENDP

; ѕолучаем восьмибайтное окончание блока

GET\_TAIL PROC near

push si

push cx

push bx

push ax

mov bx,0008h

mov cx,4

RE:

mov ax,es:[bx]

mov [si],ax

add bx,2h

add si,2h

loop RE

pop ax

pop bx

pop cx

pop si

ret

GET\_TAIL ENDP

; ќпредел¤ем и выводим цепочку блоков управлени¤ пам¤тью

GET\_MCB PROC near

push ax

push bx

push cx

push dx

lea dx, MCB\_Capt

call Write\_msg

mov ah,52h

int 21h

mov es,es:[bx-2]

mov bx,1

REPEAT:

sub ax,ax

sub cx,cx

sub di,di

sub si,si

mov al,es:[0000h]

call BYTE\_TO\_HEX

lea di,MCB\_Type+13

mov [di],ax

cmp ax,4135h

je MEN\_BX

STEP:

lea di,MCB\_Seg+21

mov ax,es:[0001h]

call WRD\_TO\_HEX

mov ax,es:[0003h]

mov cx,10h

mul cx

lea si,MCB\_Size+15

call WRD\_TO\_DEC

lea dx,MCB\_Type

call Write\_msg

lea dx,MCB\_Seg

call Write\_msg

lea dx,MCB\_Size

call Write\_msg

lea si,MCB\_Tail+2

call GET\_TAIl

lea dx,MCB\_Tail

call Write\_msg

cmp bx,0

jz END\_P

xor ax, ax

mov ax, es

add ax, es:[0003h]

inc ax

mov es, ax

jmp REPEAT

END\_P:

pop dx

pop cx

pop bx

pop ax

ret

MEN\_BX:

mov bx,0

jmp STEP

GET\_MCB ENDP

BEGIN:

; ¬ыполнение основного задани¤

call GET\_MEMORY

call GET\_EXT\_MEMORY

call GET\_MCB

; выход в DOS

xor al,al

mov ah,3Ch

int 21h

ret

END\_S:

LR3\_1 ENDS

END START

1. Файл lab3\_3.asm

;—≈√ћ≈Ќ“џ

LR3\_1 SEGMENT

ASSUME CS:LR3\_1, DS:LR3\_1, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

;ѕ≈–≈ћ≈ЌЌџ≈

av\_memory db 'Available memory: Bytes;',0dh,0ah,'$'

ext\_memory db 'Extended memory: Kilobytes;',0dh,0ah,'$'

MCB\_Capt db 'List of MCB:',0dh,0ah,'$'

MCB\_Type db 'Type of MCB: h; $'

MCB\_Seg db 'Segment`s adress: h; $'

MCB\_Size db 'MCB size: b$'

MCB\_Tail db '; ',0dh,0ah,'$'

;ѕ–ќ÷≈ƒ”–џ

; ¬ывод на экран

Write\_msg PROC near

mov ah,09h

int 21h

ret

Write\_msg ENDP

; ‘ункци¤ перевода из 2 с/с в 16 половины байта

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 переводитс¤ 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 старша¤ цифра

pop cx ; в AH младша¤

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near

; первод в 16 с/с 16-ти разр¤дного 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 одного байта в 10 с/с, SI - адрес пол¤ младшей цифры

push cx

push dx

push ax

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 ax,00h

jbe end\_l

or al,30h

mov [si],al

end\_l:

pop ax

pop dx

pop cx

ret

BYTE\_TO\_DEC ENDP

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

WRD\_TO\_DEC PROC near

; перевод 2 байтов в 10 с/с, SI - адрес пол¤ младшей цифры

push cx

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

; ќпределение доступной пам¤ти

GET\_MEMORY PROC near

push ax

push bx

push cx

push dx

mov bx, 0ffffh

mov ah, 4Ah

int 21h

mov ax, bx

mov cx, 10h

mul cx

lea si, av\_memory+25

call WRD\_TO\_DEC

lea dx, av\_memory

call Write\_msg

lea ax, END\_S

mov bx, 10h

xor dx, dx

div bx

inc ax

mov bx, ax

mov ah, 4Ah

int 21h

mov bx, 1000h

mov ah, 48h

int 21h

pop dx

pop cx

pop bx

pop ax

ret

GET\_MEMORY ENDP

; ќпределение расширенной пам¤ти

GET\_EXT\_MEMORY PROC near

push ax

push bx

push si

push dx

mov al, 30h ; запись адреса ¤чейки CMOS

out 70h, al

in al, 71h ; чтение младшего байта

mov bl, al ; размера расширенной пам¤ти

mov al, 31h ; запись адреса ¤чейки CMOS

out 70h, al

in al, 71h ; чтение старшего байта размера расширенной пам¤ти

mov ah, al

mov al, bl ; теперь в AX размер расширенной пам¤ти

sub dx, dx

lea si, ext\_memory+25

call WRD\_TO\_DEC

lea dx, ext\_memory

call write\_msg

pop dx

pop si

pop bx

pop ax

ret

GET\_EXT\_MEMORY ENDP

; ѕолучаем восьмибайтное окончание блока

GET\_TAIL PROC near

push si

push cx

push bx

push ax

mov bx,0008h

mov cx,4

RE:

mov ax,es:[bx]

mov [si],ax

add bx,2h

add si,2h

loop RE

pop ax

pop bx

pop cx

pop si

ret

GET\_TAIL ENDP

; ќпредел¤ем и выводим цепочку блоков управлени¤ пам¤тью

GET\_MCB PROC near

push ax

push bx

push cx

push dx

lea dx, MCB\_Capt

call Write\_msg

mov ah,52h

int 21h

mov es,es:[bx-2]

mov bx,1

REPEAT:

sub ax,ax

sub cx,cx

sub di,di

sub si,si

mov al,es:[0000h]

call BYTE\_TO\_HEX

lea di,MCB\_Type+13

mov [di],ax

cmp ax,4135h

je MEN\_BX

STEP:

lea di,MCB\_Seg+21

mov ax,es:[0001h]

call WRD\_TO\_HEX

mov ax,es:[0003h]

mov cx,10h

mul cx

lea si,MCB\_Size+15

call WRD\_TO\_DEC

lea dx,MCB\_Type

call Write\_msg

lea dx,MCB\_Seg

call Write\_msg

lea dx,MCB\_Size

call Write\_msg

lea si,MCB\_Tail+2

call GET\_TAIl

lea dx,MCB\_Tail

call Write\_msg

cmp bx,0

jz END\_P

xor ax, ax

mov ax, es

add ax, es:[0003h]

inc ax

mov es, ax

jmp REPEAT

END\_P:

pop dx

pop cx

pop bx

pop ax

ret

MEN\_BX:

mov bx,0

jmp STEP

GET\_MCB ENDP

BEGIN:

; ¬ыполнение основного задани¤

call GET\_MEMORY

call GET\_EXT\_MEMORY

call GET\_MCB

; выход в DOS

xor al,al

mov ah,3Ch

int 21h

ret

END\_S:

LR3\_1 ENDS

END START

1. Файл lab3\_4.asm

;—≈√ћ≈Ќ“џ

LR3\_1 SEGMENT

ASSUME CS:LR3\_1, DS:LR3\_1, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

;ѕ≈–≈ћ≈ЌЌџ≈

av\_memory db 'Available memory: Bytes;',0dh,0ah,'$'

ext\_memory db 'Extended memory: Kilobytes;',0dh,0ah,'$'

MCB\_Capt db 'List of MCB:',0dh,0ah,'$'

MCB\_Type db 'Type of MCB: h; $'

MCB\_Seg db 'Segment`s adress: h; $'

MCB\_Size db 'MCB size: b$'

MCB\_Tail db '; ',0dh,0ah,'$'

IF\_ERROR db 'ERROR!!!$'

;ѕ–ќ÷≈ƒ”–џ

; ¬ывод на экран

Write\_msg PROC near

mov ah,09h

int 21h

ret

Write\_msg ENDP

; ‘ункци¤ перевода из 2 с/с в 16 половины байта

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 переводитс¤ 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 старша¤ цифра

pop cx ; в AH младша¤

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near

; первод в 16 с/с 16-ти разр¤дного 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 одного байта в 10 с/с, SI - адрес пол¤ младшей цифры

push cx

push dx

push ax

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 ax,00h

jbe end\_l

or al,30h

mov [si],al

end\_l:

pop ax

pop dx

pop cx

ret

BYTE\_TO\_DEC ENDP

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

WRD\_TO\_DEC PROC near

; перевод 2 байтов в 10 с/с, SI - адрес пол¤ младшей цифры

push cx

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

; ќпределение доступной пам¤ти после BEGIN

; ќпределение расширенной пам¤ти

GET\_EXT\_MEMORY PROC near

push ax

push bx

push si

push dx

mov al, 30h ; запись адреса ¤чейки CMOS

out 70h, al

in al, 71h ; чтение младшего байта

mov bl, al ; размера расширенной пам¤ти

mov al, 31h ; запись адреса ¤чейки CMOS

out 70h, al

in al, 71h ; чтение старшего байта размера расширенной пам¤ти

mov ah, al

mov al, bl ; теперь в AX размер расширенной пам¤ти

sub dx, dx

lea si, ext\_memory+25

call WRD\_TO\_DEC

lea dx, ext\_memory

call write\_msg

pop dx

pop si

pop bx

pop ax

ret

GET\_EXT\_MEMORY ENDP

; ѕолучаем восьмибайтное окончание блока

GET\_TAIL PROC near

push si

push cx

push bx

push ax

mov bx,0008h

mov cx,4

RE:

mov ax,es:[bx]

mov [si],ax

add bx,2h

add si,2h

loop RE

pop ax

pop bx

pop cx

pop si

ret

GET\_TAIL ENDP

; ќпредел¤ем и выводим цепочку блоков управлени¤ пам¤тью

GET\_MCB PROC near

push ax

push bx

push cx

push dx

lea dx, MCB\_Capt

call Write\_msg

mov ah,52h

int 21h

mov es,es:[bx-2]

mov bx,1

REPEAT:

sub ax,ax

sub cx,cx

sub di,di

sub si,si

mov al,es:[0000h]

call BYTE\_TO\_HEX

lea di,MCB\_Type+13

mov [di],ax

cmp ax,4135h

je MEN\_BX

STEP:

lea di,MCB\_Seg+21

mov ax,es:[0001h]

call WRD\_TO\_HEX

mov ax,es:[0003h]

mov cx,10h

mul cx

lea si,MCB\_Size+15

call WRD\_TO\_DEC

lea dx,MCB\_Type

call Write\_msg

lea dx,MCB\_Seg

call Write\_msg

lea dx,MCB\_Size

call Write\_msg

lea si,MCB\_Tail+2

call GET\_TAIl

lea dx,MCB\_Tail

call Write\_msg

cmp bx,0

jz END\_P

xor ax, ax

mov ax, es

add ax, es:[0003h]

inc ax

mov es, ax

jmp REPEAT

END\_P:

pop dx

pop cx

pop bx

pop ax

ret

MEN\_BX:

mov bx,0

jmp STEP

GET\_MCB ENDP

BEGIN:

; ¬ыполнение основного задани¤

push ax

push bx

push cx

push dx

mov ah, 4Ah

mov bx, 0ffffh

int 21h

mov ax,bx

mov cx,10h

mul cx

lea si,av\_memory+25

call WRD\_TO\_DEC

lea dx,av\_memory

call Write\_msg

mov bx, 4096h

mov ah, 48h

int 21h

pop ax

pop bx

pop cx

pop dx

jc Y\_ERROR ;проверка флага CF

jmp N\_ERROR

Y\_ERROR:

lea dx,IF\_ERROR

call Write\_msg

jmp END3

N\_ERROR:

call GET\_EXT\_MEMORY

call GET\_MCB

; выход в DOS

END3:

xor al,al

mov ah,3Ch

int 21h

ret

LR3\_1 ENDS

END START