**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практической работе №3**

**по дисциплине «Операционные системы»**

**Тема**: **Исследование организации управления основной памятью**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2020

**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.

**Основные теоретические положения.**

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:

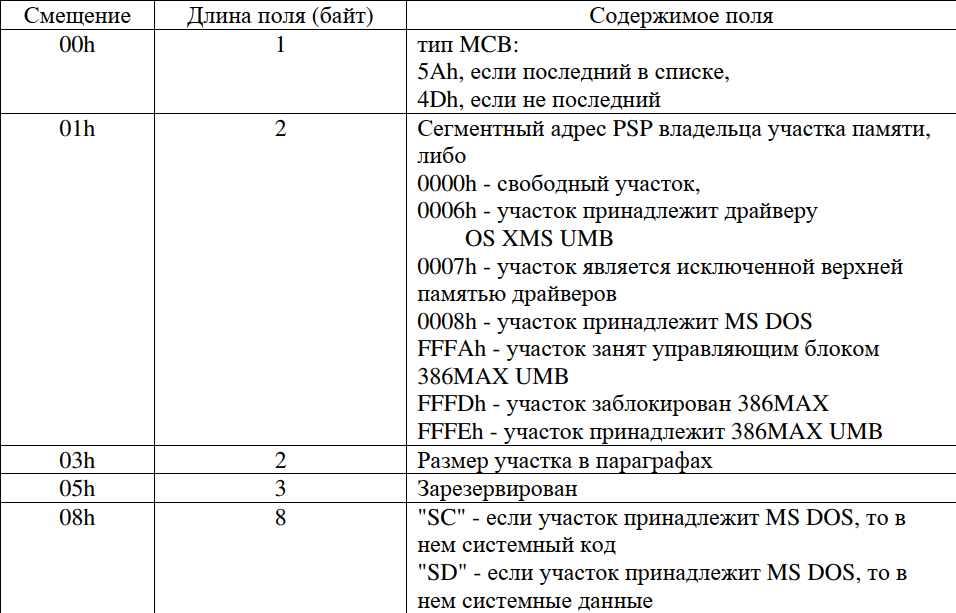
****

Рисунок 1 – Структура MCB

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 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 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

**Выполнение работы.**

Написан текст исходного .COM модуля, который выбирает и распечатывает следующую информацию:

* Количество доступной памяти
* Размер расширенной памяти
* Выводит цепочку блоков управления памятью

Полученный исходный модуль был отлажен. Результаты выполнения программы представлены на рис. 2.

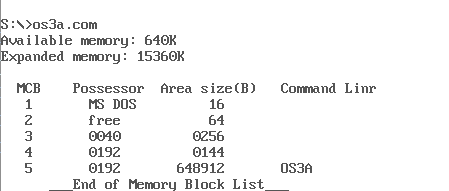


Рисунок 2 – Вывод программы os3a.com

Программа занимает всю доступную память.

Далее программа была изменена таким образом, чтобы она освобождала память, которую она не занимает. Была использована функция 4Ah прерывания 21h. Результат выполнения программы представлен на рис. 3.

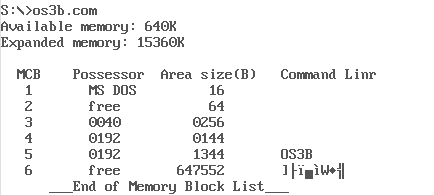


Рисунок 3 – Вывод программы os3b.com

В данном случае мы освобождаем память. Как видно из рисунка, освобожденная память относится к шестому блоку управления памятью, который является свободным.

Далее программа была изменена таким образом, чтобы после освобождения памяти, она запрашивала 64Кб памяти функцией 48h прерывания 21h. Результат выполнения программы представлен на рис. 4.

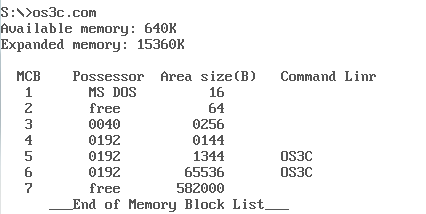


Рисунок 4 – Вывод программы os3c.com

В данном случае мы сначала выделяем всю доступную память, потом освобождаем то, что не нужно. Затем запрашиваем блок памяти 64 Кб, в итоге система выделяет нам ещё 65536 б памяти. Сегментный адрес PSP владельца участка памяти 5 и 6 блока совпадают.

Далее программа была изменена таким образом, чтобы сначала она запрашивала дополнительно 64Кб, а затем освобождала память. Результат выполнения программы представлен на рис. 5.

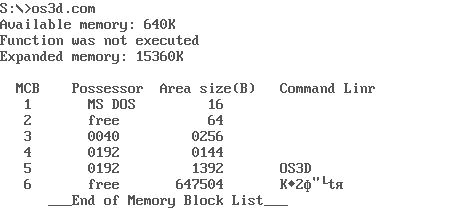


Рисунок 5 – Вывод программы os3d.com

В данном случае мы выделяем всё доступную память, а затем ещё запрашиваем 64 кб. В результате возникает ошибка. Она возникает из-за того, что в первый раз уже была выделена вся доступная память.

**Контрольные вопросы**

* **Что означает «доступный объём памяти»?**

Максимальный объем памяти, который может использовать программа.

* **Где MCB блок Вашей программы в списке?**

Принадлежность MCB можно определить по сегментной компоненте адреса владельца блока.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 6 – Смещение 1 байт в MCB

При загрузке для программы выделяются блоки памяти, располагающиеся в следующей последовательности:

* MCB для блока памяти переменных среды;
* блок памяти переменных среды;
* MCB программного блока памяти;
* префикс программного сегмента PSP;
* программный модуль.

Во всех случаях программа имеет два блока управления памятью. 4 блок - MCB для блока памяти переменных среды и 5 блок - MCB программного блока памяти. В третьем случае также появляется 6 блок для управления выделенной памятью размером 64 Кб.

* **Какой размер памяти занимает программа в каждом случае?**
* OS3A.ASM

Программа занимает (648912Б + 144Б) 649056 байт.

* OS3B.ASM

Программа занимает (1344Б + 144Б) 1488 байт после освобождения памяти.

* OS3C.ASM

Программа занимает (1344Б + 144Б) 1488 байт после освобождения памяти и дополнительно 65536 байт после выделения.

* OS3D.ASM

Программа занимает (1392Б + 144Б) 1536 байт.

**Вывод.**

В результате выполнения данной лабораторной работы был изучен список блоков управления памятью, а также методы выделения и освобождения памяти для программы.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ. OS3A.ASM**

lab segment

assume cs:lab, ds:lab, es:nothing, ss:nothing

org 100h

main: jmp processing

LastMCB EQU 5Ah ;тип последнего MCB

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 16 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 цифра

ret

byte\_to\_hex endp

word\_to\_hex proc near

;перевод в 16 сс 16 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

xor ah, ah

call byte\_to\_hex

mov [di], ah

dec di

mov [di], al

pop bx

ret

word\_to\_hex endp

word\_to\_dec proc near

;перевод числа в 16 сс

;в ax - число, si - адрес последнего символа

push cx

push dx

mov cx, 10

pr: div cx

or dl, 30h

mov [si], dl

dec si

xor dx, dx

cmp ax, 10

jae pr

cmp al, 0

je end\_pr

or al, 30h

mov [si], al

end\_pr: pop dx

pop cx

ret

word\_to\_dec endp

word\_to\_str proc near

;на входе ax число 16 бит

;si указатель на строку

;bx разрядность результата

push ax

push bx

push cx

push dx

push di

push si

cmp bx, 16

ja end\_wts

cmp ax, 7FFFh

jna plus

mov byte ptr [si], '-'

inc si

not ax

inc ax

plus:

xor cx, cx

jmp manipulation

manipulation:

xor dx, dx

div bx

mov di, ax

mov al, dl

cmp al, 10

sbb al, 69h

das

push di

lea di, mesto

add di, cx

mov byte ptr [di], al

pop di

mov ax, di

inc cx

test ax, ax

jz endrep

jmp manipulation

endrep:

lea di, mesto

add di, cx

copyrep:

dec di

mov dl, byte ptr [di]

mov byte ptr [si], dl

inc si

loop copyrep

end\_wts:

pop si

pop di

pop dx

pop cx

pop bx

pop ax

ret

word\_to\_str endp

print proc near

push ax

mov ah, 09h

int 21h

pop ax

ret

print endp

MemoryInfo proc near

push ax

push bx

push cx

push dx

push di

push si

mov si, offset avlmem

add si, 18

int 12h

;Reports the number of contiguous 1K memory blocks in the system (up to 640K)

;This is the amount of memory available to the entire system

;This is not the amount of memory available to the user's program

mov bx, 10

call word\_to\_str

lea dx, avlmem

call print

mov al, 30h

out 70h, al

in al, 71h ;чтение младшего байта

mov bl, al

mov al, 31h

out 70h, al

in al, 71h ;чтение старшего байта

mov ah, al

mov al, bl

lea si, expmem

add si, 17

mov bx, 10

call word\_to\_str

lea dx, expmem

call print

pop si

pop di

pop dx

pop cx

pop bx

pop ax

ret

MemoryInfo endp

MCB\_processing PROC near

push ax

push bx

push cx

push dx

mov ah, 52h

int 21h

sub bx, 2

mov ax, word ptr es:[bx]

mov es, ax ;адрес первого блока

xor di, di

mov cx, 1

lea dx, tit ;таблица

call print

manipulations:

mov ax, cx

inc cx

lea si, count

add si, 5

mov bx, 10

call word\_to\_str

lea dx, count

call PRINT ;номер MCB блока

push cx

xor ax, ax

mov al, es:[0h] ;тип MCB

push ax

mov ax, es:[1h] ;владелец

irpc case, 0678

cmp ax, 000&case&h

je MCB\_label\_&case&

endm

irpc case, ADE

cmp ax, 0FFF&case&h

je MCB\_label\_&case&

endm

lea di, space

add di, 5

call word\_to\_hex

lea dx, space

call print

jmp MCB\_size

irpc met, 0678ADE

MCB\_label\_&met&:

lea dx, owner\_&met&

call print

jmp MCB\_size

endm

MCB\_size: ;размер

mov ax, es:[3h]

mov bx, 16

mul bx

lea si, space

add si, 5

call word\_to\_dec

lea dx, space

call print

mov cx, 8

xor si, si

Linr: mov dl, es:[si+8h]

mov ah, 02h

int 21h

inc si

loop Linr

mov ax, es:[3h]

mov bx, es

add bx, ax

inc bx

mov es, bx ;адрес следующего блока

pop ax

pop cx

cmp al, LastMCB

je ending

jmp manipulations

ending:

lea dx, endstr

call print

pop dx

pop cx

pop bx

pop ax

ret

MCB\_processing endp

processing:

call MemoryInfo

call MCB\_processing

mov ah, 4ch

int 21h

ret

avlmem db 'Available memory: K', 13, 10, '$'

expmem db 'Expanded memory: K', 13, 10, '$'

mesto db 16 dup (0)

tit db 13, 10, ' MCB Possessor Area size(B) Command Linr $'

endstr db 13, 10, ' \_\_\_End of Memory Block List\_\_\_$'

space db 13 dup (?), '$'

count db 13, 10, 9 dup (?), '$'

owner\_0 db ' free $'

owner\_6 db 'OS XMS UMB$'

owner\_7 db 'Excluded top memory of driver$'

owner\_8 db ' MS DOS $'

owner\_A db '386MAX UMB$'

owner\_D db '386MAX$'

owner\_E db '386MAX UMB$'

lab ends

end main

**ИСХОДНЫЙ КОД ПРОГРАММЫ. OS3B.ASM**

lab segment

assume cs:lab, ds:lab, es:nothing, ss:nothing

org 100h

main: jmp processing

LastMCB EQU 5Ah ;тип последнего MCB

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 16 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 цифра

ret

byte\_to\_hex endp

word\_to\_hex proc near

;перевод в 16 сс 16 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

xor ah, ah

call byte\_to\_hex

mov [di], ah

dec di

mov [di], al

pop bx

ret

word\_to\_hex endp

word\_to\_dec proc near

;перевод числа в 16 сс

;в ax - число, si - адрес последнего символа

push cx

push dx

mov cx, 10

pr: div cx

or dl, 30h

mov [si], dl

dec si

xor dx, dx

cmp ax, 10

jae pr

cmp al, 0

je end\_pr

or al, 30h

mov [si], al

end\_pr: pop dx

pop cx

ret

word\_to\_dec endp

word\_to\_str proc near

;на входе ax число 16 бит

;si указатель на строку

;bx разрядность результата

push ax

push bx

push cx

push dx

push di

push si

cmp bx, 16

ja end\_wts

cmp ax, 7FFFh

jna plus

mov byte ptr [si], '-'

inc si

not ax

inc ax

plus:

xor cx, cx

jmp manipulation

manipulation:

xor dx, dx

div bx

mov di, ax

mov al, dl

cmp al, 10

sbb al, 69h

das

push di

lea di, mesto

add di, cx

mov byte ptr [di], al

pop di

mov ax, di

inc cx

test ax, ax

jz endrep

jmp manipulation

endrep:

lea di, mesto

add di, cx

copyrep:

dec di

mov dl, byte ptr [di]

mov byte ptr [si], dl

inc si

loop copyrep

end\_wts:

pop si

pop di

pop dx

pop cx

pop bx

pop ax

ret

word\_to\_str endp

print proc near

push ax

mov ah, 09h

int 21h

pop ax

ret

print endp

MemoryInfo proc near

push ax

push bx

push cx

push dx

push di

push si

mov si, offset avlmem

add si, 18

int 12h

;Reports the number of contiguous 1K memory blocks in the system (up to 640K)

;This is the amount of memory available to the entire system

;This is not the amount of memory available to the user's program

mov bx, 10

call word\_to\_str

lea dx, avlmem

call print

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

lea bx, last ;смещение конца программы

mov cl, 4 ;вычисляем длину в параграфах

shr bx, cl

add bx, 17 ;добавляем 1 параграф для выравнивания

mov ah, 4Ah ;изменяем размер выделенного блока памяти

int 21h

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

mov al, 30h

out 70h, al

in al, 71h ;чтение младшего байта

mov bl, al

mov al, 31h

out 70h, al

in al, 71h ;чтение старшего байта

mov ah, al

mov al, bl

lea si, expmem

add si, 17

mov bx, 10

call word\_to\_str

lea dx, expmem

call print

pop si

pop di

pop dx

pop cx

pop bx

pop ax

ret

MemoryInfo endp

MCB\_processing PROC near

push ax

push bx

push cx

push dx

mov ah, 52h

int 21h

sub bx, 2

mov ax, word ptr es:[bx]

mov es, ax ;адрес первого блока

xor di, di

mov cx, 1

lea dx, tit ;таблица

call print

manipulations:

mov ax, cx

inc cx

lea si, count

add si, 5

mov bx, 10

call word\_to\_str

lea dx, count

call PRINT ;номер MCB блока

push cx

xor ax, ax

mov al, es:[0h] ;тип MCB

push ax

mov ax, es:[1h] ;владелец

irpc case, 0678

cmp ax, 000&case&h

je MCB\_label\_&case&

endm

irpc case, ADE

cmp ax, 0FFF&case&h

je MCB\_label\_&case&

endm

lea di, space

add di, 5

call word\_to\_hex

lea dx, space

call print

jmp MCB\_size

irpc met, 0678ADE

MCB\_label\_&met&:

lea dx, owner\_&met&

call print

jmp MCB\_size

endm

MCB\_size: ;размер

mov ax, es:[3h]

mov bx, 16

mul bx

lea si, space

add si, 5

call word\_to\_dec

lea dx, space

call print

mov cx, 8

xor si, si

Linr: mov dl, es:[si+8h]

mov ah, 02h

int 21h

inc si

loop Linr

mov ax, es:[3h]

mov bx, es

add bx, ax

inc bx

mov es, bx ;адрес следующего блока

pop ax

pop cx

cmp al, LastMCB

je ending

jmp manipulations

ending:

lea dx, endstr

call print

pop dx

pop cx

pop bx

pop ax

ret

MCB\_processing endp

processing:

call MemoryInfo

call MCB\_processing

mov ah, 4ch

int 21h

ret

avlmem db 'Available memory: K', 13, 10, '$'

expmem db 'Expanded memory: K', 13, 10, '$'

mesto db 16 dup (0)

tit db 13, 10, ' MCB Possessor Area size(B) Command Linr $'

endstr db 13, 10, ' \_\_\_End of Memory Block List\_\_\_$'

space db 13 dup (?), '$'

count db 13, 10, 9 dup (?), '$'

owner\_0 db ' free $'

owner\_6 db 'OS XMS UMB$'

owner\_7 db 'Excluded top memory of driver$'

owner\_8 db ' MS DOS $'

owner\_A db '386MAX UMB$'

owner\_D db '386MAX$'

owner\_E db '386MAX UMB$'

last db ?

lab ends

end main

**ИСХОДНЫЙ КОД ПРОГРАММЫ. OS3C.ASM**

lab segment

assume cs:lab, ds:lab, es:nothing, ss:nothing

org 100h

main: jmp processing

LastMCB EQU 5Ah ;тип последнего MCB

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 16 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 цифра

ret

byte\_to\_hex endp

word\_to\_hex proc near

;перевод в 16 сс 16 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

xor ah, ah

call byte\_to\_hex

mov [di], ah

dec di

mov [di], al

pop bx

ret

word\_to\_hex endp

word\_to\_dec proc near

;перевод числа в 16 сс

;в ax - число, si - адрес последнего символа

push cx

push dx

mov cx, 10

pr: div cx

or dl, 30h

mov [si], dl

dec si

xor dx, dx

cmp ax, 10

jae pr

cmp al, 0

je end\_pr

or al, 30h

mov [si], al

end\_pr: pop dx

pop cx

ret

word\_to\_dec endp

word\_to\_str proc near

;на входе ax число 16 бит

;si указатель на строку

;bx разрядность результата

push ax

push bx

push cx

push dx

push di

push si

cmp bx, 16

ja end\_wts

cmp ax, 7FFFh

jna plus

mov byte ptr [si], '-'

inc si

not ax

inc ax

plus:

xor cx, cx

jmp manipulation

manipulation:

xor dx, dx

div bx

mov di, ax

mov al, dl

cmp al, 10

sbb al, 69h

das

push di

lea di, mesto

add di, cx

mov byte ptr [di], al

pop di

mov ax, di

inc cx

test ax, ax

jz endrep

jmp manipulation

endrep:

lea di, mesto

add di, cx

copyrep:

dec di

mov dl, byte ptr [di]

mov byte ptr [si], dl

inc si

loop copyrep

end\_wts:

pop si

pop di

pop dx

pop cx

pop bx

pop ax

ret

word\_to\_str endp

print proc near

push ax

mov ah, 09h

int 21h

pop ax

ret

print endp

MemoryInfo proc near

push ax

push bx

push cx

push dx

push di

push si

mov si, offset avlmem

add si, 18

int 12h

;Reports the number of contiguous 1K memory blocks in the system (up to 640K)

;This is the amount of memory available to the entire system

;This is not the amount of memory available to the user's program

mov bx, 10

call word\_to\_str

lea dx, avlmem

call print

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

lea bx, last ;смещение конца программы

mov cl, 4 ;вычисляем длину в параграфах

shr bx, cl

add bx, 17 ;добавляем 1 параграф для выравнивания

mov ah, 4Ah ;изменяем размер выделенного блока памяти

int 21h

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

mov ah, 48h

mov bx, 1000h ;запрошенное количество памяти в 16-байтовых параграфах

int 21h

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

mov al, 30h

out 70h, al

in al, 71h ;чтение младшего байта

mov bl, al

mov al, 31h

out 70h, al

in al, 71h ;чтение старшего байта

mov ah, al

mov al, bl

lea si, expmem

add si, 17

mov bx, 10

call word\_to\_str

lea dx, expmem

call print

pop si

pop di

pop dx

pop cx

pop bx

pop ax

ret

MemoryInfo endp

MCB\_processing PROC near

push ax

push bx

push cx

push dx

mov ah, 52h

int 21h

sub bx, 2

mov ax, word ptr es:[bx]

mov es, ax ;адрес первого блока

xor di, di

mov cx, 1

lea dx, tit ;таблица

call print

manipulations:

mov ax, cx

inc cx

lea si, count

add si, 5

mov bx, 10

call word\_to\_str

lea dx, count

call PRINT ;номер MCB блока

push cx

xor ax, ax

mov al, es:[0h] ;тип MCB

push ax

mov ax, es:[1h] ;владелец

irpc case, 0678

cmp ax, 000&case&h

je MCB\_label\_&case&

endm

irpc case, ADE

cmp ax, 0FFF&case&h

je MCB\_label\_&case&

endm

lea di, space

add di, 5

call word\_to\_hex

lea dx, space

call print

jmp MCB\_size

irpc met, 0678ADE

MCB\_label\_&met&:

lea dx, owner\_&met&

call print

jmp MCB\_size

endm

MCB\_size: ;размер

mov ax, es:[3h]

mov bx, 16

mul bx

lea si, space

add si, 5

call word\_to\_dec

lea dx, space

call print

mov cx, 8

xor si, si

Linr: mov dl, es:[si+8h]

mov ah, 02h

int 21h

inc si

loop Linr

mov ax, es:[3h]

mov bx, es

add bx, ax

inc bx

mov es, bx ;адрес следующего блока

pop ax

pop cx

cmp al, LastMCB

je ending

jmp manipulations

ending:

lea dx, endstr

call print

pop dx

pop cx

pop bx

pop ax

ret

MCB\_processing endp

processing:

call MemoryInfo

call MCB\_processing

mov ah, 4ch

int 21h

ret

avlmem db 'Available memory: K', 13, 10, '$'

expmem db 'Expanded memory: K', 13, 10, '$'

mesto db 16 dup (0)

tit db 13, 10, ' MCB Possessor Area size(B) Command Linr $'

endstr db 13, 10, ' \_\_\_End of Memory Block List\_\_\_$'

space db 13 dup (?), '$'

count db 13, 10, 9 dup (?), '$'

owner\_0 db ' free $'

owner\_6 db 'OS XMS UMB$'

owner\_7 db 'Excluded top memory of driver$'

owner\_8 db ' MS DOS $'

owner\_A db '386MAX UMB$'

owner\_D db '386MAX$'

owner\_E db '386MAX UMB$'

last db ?

lab ends

end main

**ИСХОДНЫЙ КОД ПРОГРАММЫ. OS3D.ASM**

lab segment

assume cs:lab, ds:lab, es:nothing, ss:nothing

org 100h

main: jmp processing

LastMCB EQU 5Ah ;тип последнего MCB

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 16 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 цифра

ret

byte\_to\_hex endp

word\_to\_hex proc near

;перевод в 16 сс 16 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

xor ah, ah

call byte\_to\_hex

mov [di], ah

dec di

mov [di], al

pop bx

ret

word\_to\_hex endp

word\_to\_dec proc near

;перевод числа в 16 сс

;в ax - число, si - адрес последнего символа

push cx

push dx

mov cx, 10

pr: div cx

or dl, 30h

mov [si], dl

dec si

xor dx, dx

cmp ax, 10

jae pr

cmp al, 0

je end\_pr

or al, 30h

mov [si], al

end\_pr: pop dx

pop cx

ret

word\_to\_dec endp

word\_to\_str proc near

;на входе ax число 16 бит

;si указатель на строку

;bx разрядность результата

push ax

push bx

push cx

push dx

push di

push si

cmp bx, 16

ja end\_wts

cmp ax, 7FFFh

jna plus

mov byte ptr [si], '-'

inc si

not ax

inc ax

plus:

xor cx, cx

jmp manipulation

manipulation:

xor dx, dx

div bx

mov di, ax

mov al, dl

cmp al, 10

sbb al, 69h

das

push di

lea di, mesto

add di, cx

mov byte ptr [di], al

pop di

mov ax, di

inc cx

test ax, ax

jz endrep

jmp manipulation

endrep:

lea di, mesto

add di, cx

copyrep:

dec di

mov dl, byte ptr [di]

mov byte ptr [si], dl

inc si

loop copyrep

end\_wts:

pop si

pop di

pop dx

pop cx

pop bx

pop ax

ret

word\_to\_str endp

print proc near

push ax

mov ah, 09h

int 21h

pop ax

ret

print endp

MemoryInfo proc near

push ax

push bx

push cx

push dx

push di

push si

mov si, offset avlmem

add si, 18

int 12h

;Reports the number of contiguous 1K memory blocks in the system (up to 640K)

;This is the amount of memory available to the entire system

;This is not the amount of memory available to the user's program

mov bx, 10

call word\_to\_str

lea dx, avlmem

call print

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

mov ah, 48h

mov bx, 1000h ;запрошенное количество памяти в 16-байтовых параграфах

int 21h

jnc no\_error ;переход, если перенос не установлен

lea dx, error\_msg

call print

no\_error:

lea bx, last ;смещение конца программы

mov cl, 4 ;вычисляем длину в параграфах

shr bx, cl

add bx, 17 ;добавляем 1 параграф для выравнивания

mov ah, 4Ah ;изменяем размер выделенного блока памяти

int 21h

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

mov al, 30h

out 70h, al

in al, 71h ;чтение младшего байта

mov bl, al

mov al, 31h

out 70h, al

in al, 71h ;чтение старшего байта

mov ah, al

mov al, bl

lea si, expmem

add si, 17

mov bx, 10

call word\_to\_str

lea dx, expmem

call print

pop si

pop di

pop dx

pop cx

pop bx

pop ax

ret

MemoryInfo endp

MCB\_processing PROC near

push ax

push bx

push cx

push dx

mov ah, 52h

int 21h

sub bx, 2

mov ax, word ptr es:[bx]

mov es, ax ;адрес первого блока

xor di, di

mov cx, 1

lea dx, tit ;таблица

call print

manipulations:

mov ax, cx

inc cx

lea si, count

add si, 5

mov bx, 10

call word\_to\_str

lea dx, count

call PRINT ;номер MCB блока

push cx

xor ax, ax

mov al, es:[0h] ;тип MCB

push ax

mov ax, es:[1h] ;владелец

irpc case, 0678

cmp ax, 000&case&h

je MCB\_label\_&case&

endm

irpc case, ADE

cmp ax, 0FFF&case&h

je MCB\_label\_&case&

endm

lea di, space

add di, 5

call word\_to\_hex

lea dx, space

call print

jmp MCB\_size

irpc met, 0678ADE

MCB\_label\_&met&:

lea dx, owner\_&met&

call print

jmp MCB\_size

endm

MCB\_size: ;размер

mov ax, es:[3h]

mov bx, 16

mul bx

lea si, space

add si, 5

call word\_to\_dec

lea dx, space

call print

mov cx, 8

xor si, si

Linr: mov dl, es:[si+8h]

mov ah, 02h

int 21h

inc si

loop Linr

mov ax, es:[3h]

mov bx, es

add bx, ax

inc bx

mov es, bx ;адрес следующего блока

pop ax

pop cx

cmp al, LastMCB

je ending

jmp manipulations

ending:

lea dx, endstr

call print

pop dx

pop cx

pop bx

pop ax

ret

MCB\_processing endp

processing:

call MemoryInfo

call MCB\_processing

mov ah, 4ch

int 21h

ret

avlmem db 'Available memory: K', 13, 10, '$'

expmem db 'Expanded memory: K', 13, 10, '$'

mesto db 16 dup (0)

tit db 13, 10, ' MCB Possessor Area size(B) Command Linr $'

endstr db 13, 10, ' \_\_\_End of Memory Block List\_\_\_$'

space db 13 dup (?), '$'

count db 13, 10, 9 dup (?), '$'

owner\_0 db ' free $'

owner\_6 db 'OS XMS UMB$'

owner\_7 db 'Excluded top memory of driver$'

owner\_8 db ' MS DOS $'

owner\_A db '386MAX UMB$'

owner\_D db '386MAX$'

owner\_E db '386MAX UMB$'

error\_msg db 'Function was not executed', 13, 10, '$'

last db ?

lab ends

end main