**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. 6383 |  | Медведев Г.О. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы.**

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

**Описание функций и структур данных.**

|  |  |
| --- | --- |
| **Название функции** | **Назначение** |
| \_AVAILABLEMEMORY | определяет размер доступной памяти |
| \_EXTENDEDMEMORY | определяет размер расширенной памяти |
| \_DATA | определяет цепочку блоков управления памятью |
| OUTPUT | выводит цепочку блоков управления памятью |
| BYTE\_TO\_HEX | переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah |
| TETR\_TO\_HEX | вспомогательная функция для работы функции BYTE\_TO\_HEX |
| WRD\_TO\_HEX | переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры |
| BYTE\_TO\_DEC | переводит байт из AL в десятичную с/с и записывает получившееся число по адресу si, начиная с младшей цифры |
| \_TO\_DEC | переводит два байта в 10-ую с/с |
| PRINT | вызывает функцию печати строки |

**Последовательность действий, выполняемых утилитой.**

1. вывод количества доступной памяти;
2. вывод размера расширенной памяти;
3. вывод цепочки блоков управления памятью.

**Результаты выполнения программ.**

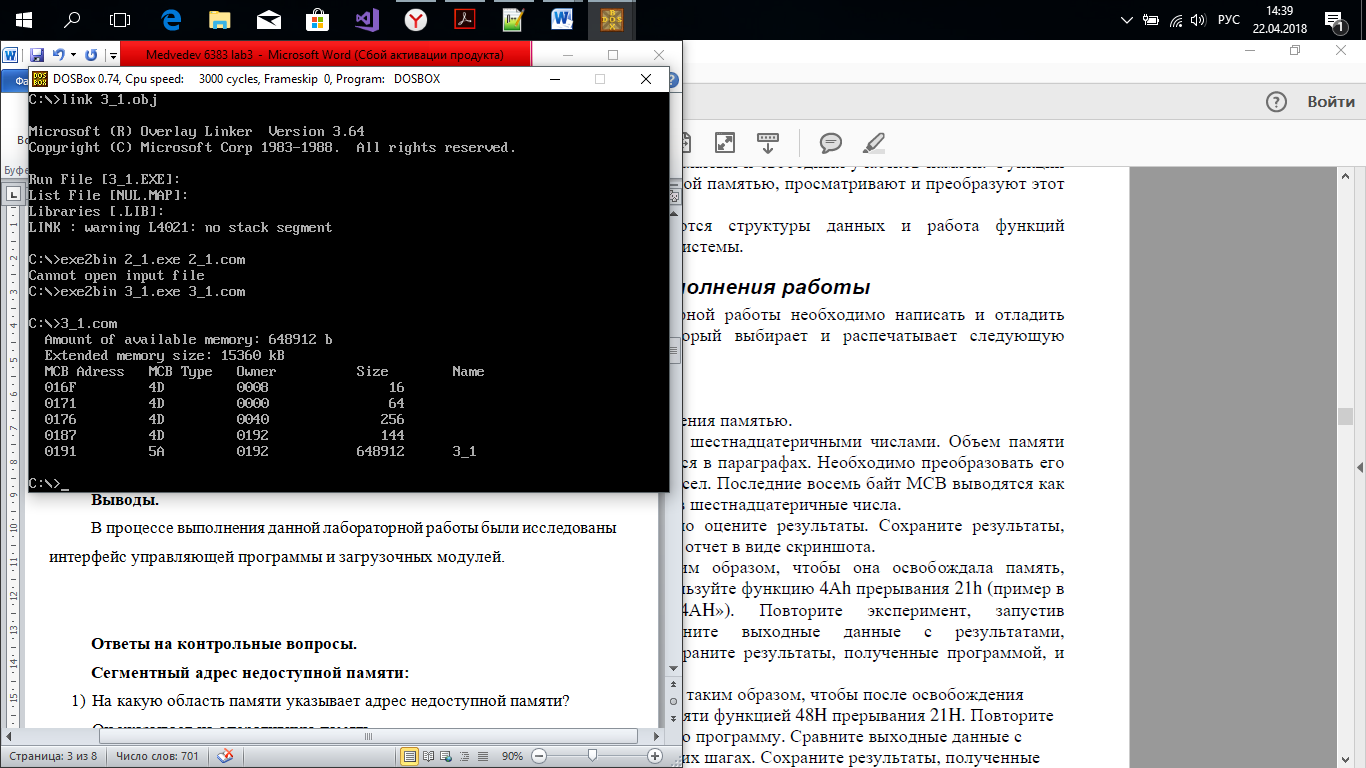


Рисунок 1 – Результат выполнения программы 3\_1.com

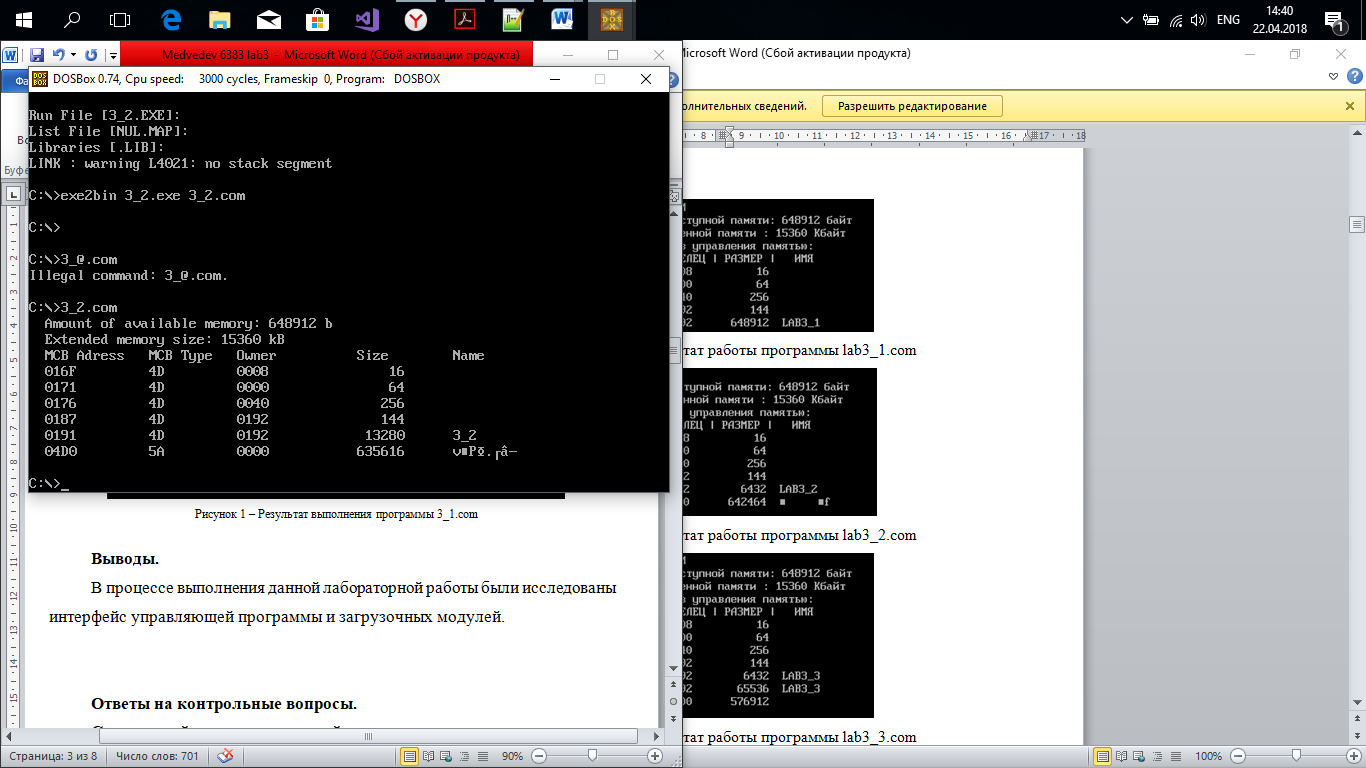


Рисунок 2 – Результат выполнения программы 3\_2.com

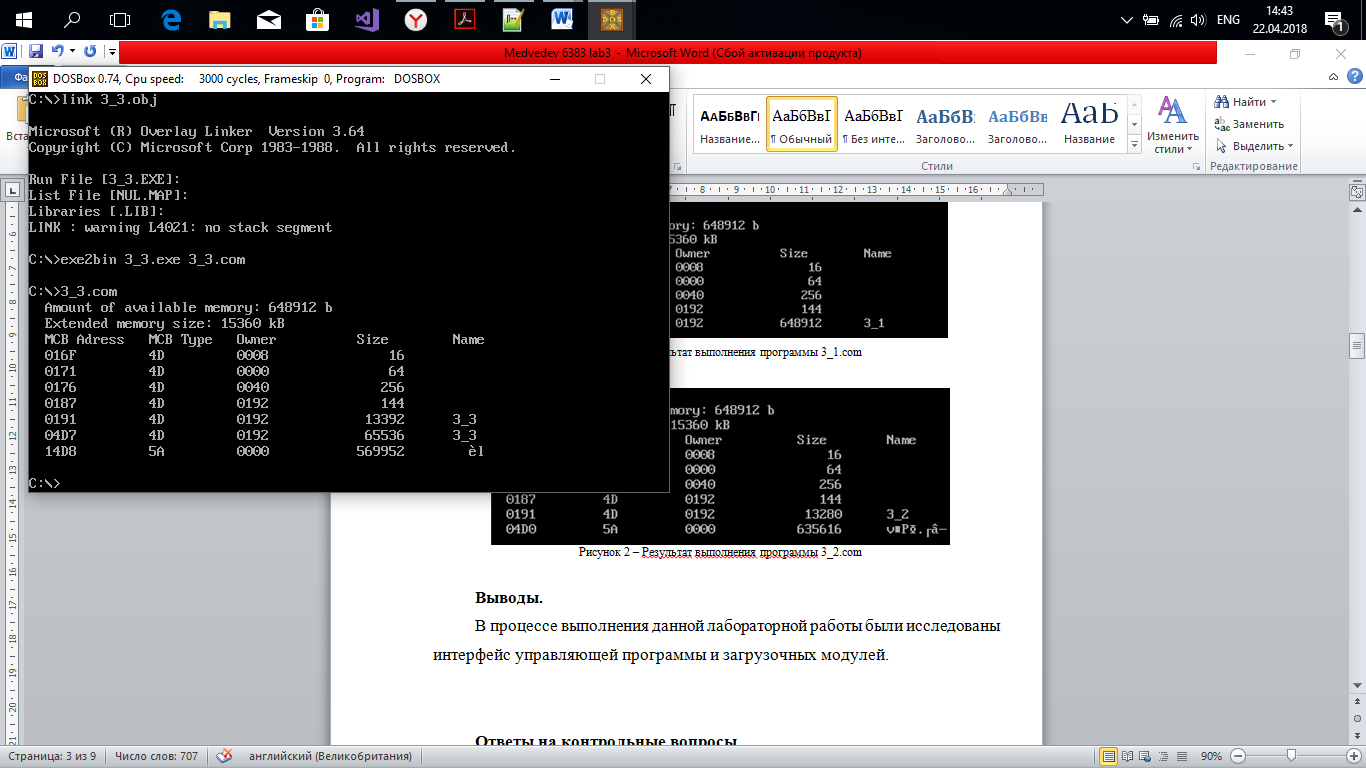


Рисунок 3 – Результат выполнения программы 3\_3.com

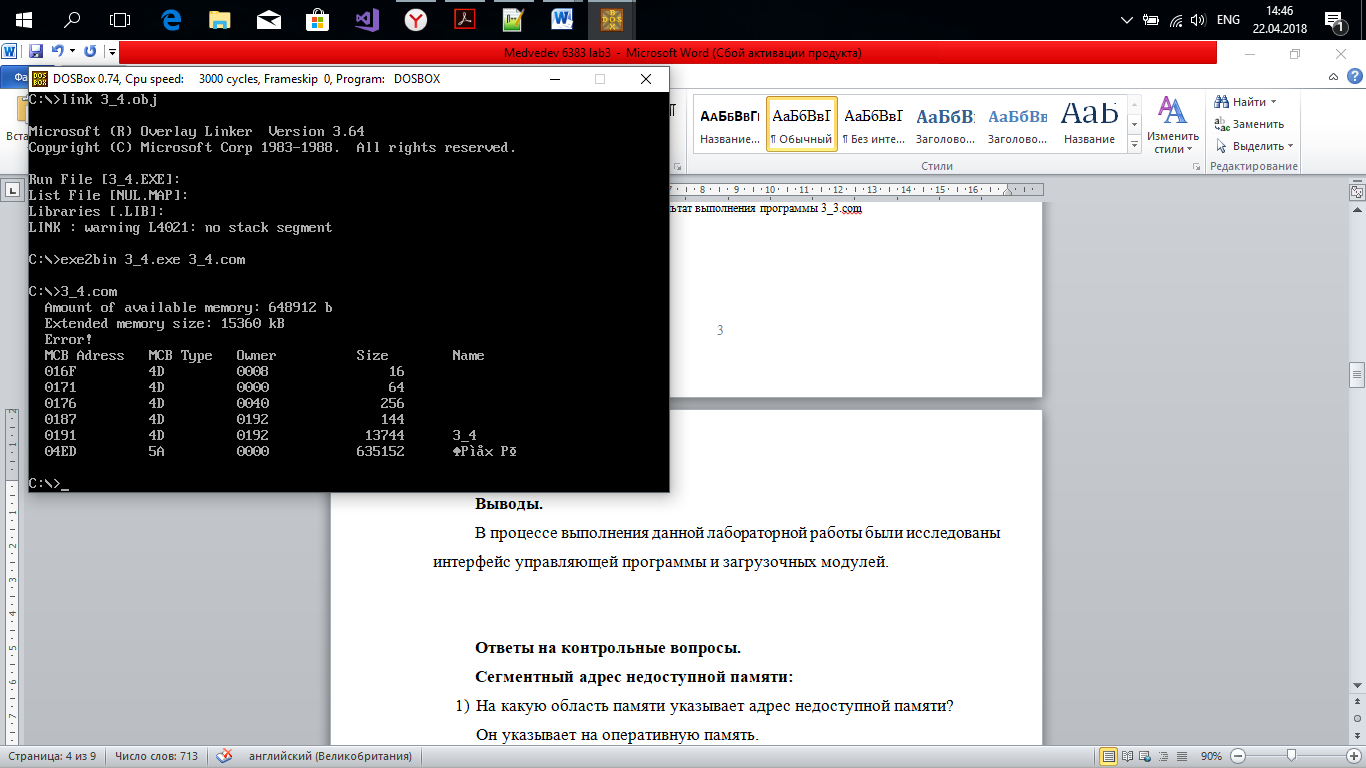


Рисунок 4 – Результат выполнения программы 3\_4.com

**Выводы.**

В процессе выполнения данной лабораторной работы были исследованы структуры данных и работы функций управления памятью ядра операционной системы.

**Ответы на контрольные вопросы.**

1. Что означает «доступный объем памяти»?

Доступный объём памяти – такой объём памяти, который может быть использован для загрузки программ.

1. Где MCB блок вашей программы в списке?

В программах 3\_1.com, 3\_2.com и 3\_4.com MCB блок имеет адрес 0191h, а в программе 3\_3.com присутствует два MCB блока, первый блок имеет адрес 0191h, второй – 04D7h. В каждой программе присутствует блок MCB размером 144б, который имеет адрес 0187h. Данный блок служит для управления памятью для области среды программы.

1. Какой размер памяти занимает программа в каждом случае?

В первом случае программа занимает 648912 байт.

Во втором случае: 648912-635616-16=13280 байт.

В третьем случае: 648912-569952-65536-32=13392 байт.

В четвёртом случае: 648912-635152-16=13744 байт.

ПРИЛОЖЕНИЕ А

3\_1.ASM

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;data

AVAILABLEMEMORY db ' Amount of available memory: b',0dh,0ah,'$'

EXTENDEDMEMORY db ' Extended memory size: kB',0dh,0ah,'$'

HEAD db ' MCB Adress MCB Type Owner Size Name ', 0dh, 0ah, '$'

DATA db ' ', 0dh, 0ah, '$'

;procedures

TETR\_TO\_HEX PROC near

and al,0fh

cmp al,09

jbe NEXT

add al,07

NEXT: add al,30h

ret

TETR\_TO\_HEX ENDP

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

BYTE\_TO\_HEX PROC near

push cx

mov ah,al

call TETR\_TO\_HEX

xchg al,ah

mov cl,4

shr al,cl

call TETR\_TO\_HEX

pop cx

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near

push bx

mov bh,ah

call BYTE\_TO\_HEX

mov [di],ah

dec di

mov [di],al

dec di

mov al,bh

call BYTE\_TO\_HEX

mov [di],ah

dec di

mov [di],al

pop bx

ret

WRD\_TO\_HEX ENDP

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

BYTE\_TO\_DEC PROC near

push cx

push dx

xor ah,ah

xor dx,dx

mov cx,10

loop\_bd:div cx

or dl,30h

mov [si],dl

dec si

xor dx,dx

cmp ax,10

jae loop\_bd

cmp al,00h

je end\_l

or al,30h

mov [si],al

end\_l: pop dx

pop cx

ret

BYTE\_TO\_DEC ENDP

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

\_TO\_DEC PROC near

push cx

push dx

push ax

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

\_TO\_DEC ENDP

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

PRINT PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

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

\_AVAILABLEMEMORY PROC NEAR ; Search for available memory

push ax

push bx

push dx

push si

xor ax, ax

mov ah, 04Ah

mov bx, 0FFFFh

int 21h

mov ax, 10h

mul bx

mov si, offset AVAILABLEMEMORY

add si, 23h

call \_TO\_DEC

pop si

pop dx

pop bx

pop ax

ret

\_AVAILABLEMEMORY ENDP

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

\_EXTENDEDMEMORY PROC near ; Search for extended memory

push ax

push bx

push si

push dx

mov al, 30h

out 70h, al

in al, 71h

mov bl, al

mov al, 31h

out 70h, al

in al, 71h

mov ah, al

mov al, bl

sub dx, dx

mov si, offset EXTENDEDMEMORY

add si, 28

call \_TO\_DEC

pop dx

pop si

pop bx

pop ax

ret

\_EXTENDEDMEMORY ENDP

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

\_DATA PROC near ; Search for MCB

mov di, offset DATA ; Address of MCB

mov ax, es

add di, 05h

call WRD\_TO\_HEX

mov di, offset DATA ; Type of MCB

add di, 0Fh

xor ah, ah

mov al, es:[00h]

call BYTE\_TO\_HEX

mov [di], al

inc di

mov [di], ah

mov di, offset DATA ; Owner

mov ax, es:[01h]

add di, 1Dh

call WRD\_TO\_HEX

mov di, offset DATA ; Size

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 2Eh

push si

mov si, di

call \_TO\_DEC

pop si

mov di, offset DATA ; Name

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

\_DATA ENDP

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

OUTPUT PROC NEAR ; Search for a chain of memory management units

mov ah, 52h

int 21h

sub bx, 2h

mov es, es:[bx]

output\_:

call \_DATA

mov dx, offset DATA

call PRINT

mov cx, es

add ax, cx

inc ax

mov es, ax

cmp bl, 4Dh

je output\_

ret

OUTPUT ENDP

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

BEGIN:

call \_AVAILABLEMEMORY

mov dx, offset AVAILABLEMEMORY

call PRINT

call \_EXTENDEDMEMORY

mov dx, offset EXTENDEDMEMORY

call PRINT

lea dx, HEAD

call PRINT

call OUTPUT

xor al, al

mov ah, 4ch

int 21h

TESTPC ENDS

END START

ПРИЛОЖЕНИЕ Б

3\_2.ASM

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;data

AVAILABLEMEMORY db ' Amount of available memory: b',0dh,0ah,'$'

EXTENDEDMEMORY db ' Extended memory size: kB',0dh,0ah,'$'

HEAD db ' MCB Adress MCB Type Owner Size Name ', 0dh, 0ah, '$'

DATA db ' ', 0dh, 0ah, '$'

;procedurs

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

TETR\_TO\_HEX PROC near

and al,0fh

cmp al,09

jbe NEXT

add al,07

NEXT: add al,30h

ret

TETR\_TO\_HEX ENDP

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

BYTE\_TO\_HEX PROC near

push cx

mov ah,al

call TETR\_TO\_HEX

xchg al,ah

mov cl,4

shr al,cl

call TETR\_TO\_HEX

pop cx

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near

push bx

mov bh,ah

call BYTE\_TO\_HEX

mov [di],ah

dec di

mov [di],al

dec di

mov al,bh

call BYTE\_TO\_HEX

mov [di],ah

dec di

mov [di],al

pop bx

ret

WRD\_TO\_HEX ENDP

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

BYTE\_TO\_DEC PROC near

push cx

push dx

xor ah,ah

xor dx,dx

mov cx,10

loop\_bd:div cx

or dl,30h

mov [si],dl

dec si

xor dx,dx

cmp ax,10

jae loop\_bd

cmp al,00h

je end\_l

or al,30h

mov [si],al

end\_l: pop dx

pop cx

ret

BYTE\_TO\_DEC ENDP

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

\_TO\_DEC PROC near

push cx

push dx

push ax

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

\_TO\_DEC ENDP

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

PRINT PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

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

\_AVAILABLEMEMORY PROC NEAR ; Search for available memory

push ax

push bx

push dx

push si

xor ax, ax

mov ah, 04Ah

mov bx, 0FFFFh

int 21h

mov ax, 10h

mul bx

mov si, offset AVAILABLEMEMORY

add si, 23h

call \_TO\_DEC

pop si

pop dx

pop bx

pop ax

ret

\_AVAILABLEMEMORY ENDP

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

\_EXTENDEDMEMORY PROC near ; Search for extended memory

push ax

push bx

push si

push dx

mov al, 30h

out 70h, al

in al, 71h

mov bl, al

mov al, 31h

out 70h, al

in al, 71h

mov ah, al

mov al, bl

sub dx, dx

mov si, offset EXTENDEDMEMORY

add si, 28

call \_TO\_DEC

pop dx

pop si

pop bx

pop ax

ret

\_EXTENDEDMEMORY ENDP

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

\_DATA PROC near ; Search for MCB

mov di, offset DATA ; Address of MCB

mov ax, es

add di, 05h

call WRD\_TO\_HEX

mov di, offset DATA ; Type of MCB

add di, 0Fh

xor ah, ah

mov al, es:[00h]

call BYTE\_TO\_HEX

mov [di], al

inc di

mov [di], ah

mov di, offset DATA ; Owner

mov ax, es:[01h]

add di, 1Dh

call WRD\_TO\_HEX

mov di, offset DATA ; Size

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 2Eh

push si

mov si, di

call \_TO\_DEC

pop si

mov di, offset DATA ; Name

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

\_DATA ENDP

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

OUTPUT PROC NEAR ; Search for a chain of memory management units

mov ah, 52h

int 21h

sub bx, 2h

mov es, es:[bx]

output\_:

call \_DATA

mov dx, offset DATA

call PRINT

mov cx, es

add ax, cx

inc ax

mov es, ax

cmp bl, 4Dh

je output\_

ret

OUTPUT ENDP

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

BEGIN:

call \_AVAILABLEMEMORY

mov dx, offset AVAILABLEMEMORY

call PRINT

call \_EXTENDEDMEMORY

mov dx, offset EXTENDEDMEMORY

call PRINT

mov ah, 4ah ; Freeing of memory

mov bx, offset END\_PROG

int 21h

mov dx, offset HEAD

call PRINT

call OUTPUT

xor al, al

mov ah, 4ch

int 21h

END\_PROG db 0

TESTPC ENDS

END START

ПРИЛОЖЕНИЕ В

3\_3.ASM

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;data

AVAILABLEMEMORY db ' Amount of available memory: b',0dh,0ah,'$'

EXTENDEDMEMORY db ' Extended memory size: kB',0dh,0ah,'$'

HEAD db ' MCB Adress MCB Type Owner Size Name ', 0dh, 0ah, '$'

DATA db ' ', 0dh, 0ah, '$'

;procedurs

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

TETR\_TO\_HEX PROC near

and al,0fh

cmp al,09

jbe NEXT

add al,07

NEXT: add al,30h

ret

TETR\_TO\_HEX ENDP

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

BYTE\_TO\_HEX PROC near

push cx

mov ah,al

call TETR\_TO\_HEX

xchg al,ah

mov cl,4

shr al,cl

call TETR\_TO\_HEX

pop cx

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near

push bx

mov bh,ah

call BYTE\_TO\_HEX

mov [di],ah

dec di

mov [di],al

dec di

mov al,bh

call BYTE\_TO\_HEX

mov [di],ah

dec di

mov [di],al

pop bx

ret

WRD\_TO\_HEX ENDP

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

BYTE\_TO\_DEC PROC near

push cx

push dx

xor ah,ah

xor dx,dx

mov cx,10

loop\_bd:div cx

or dl,30h

mov [si],dl

dec si

xor dx,dx

cmp ax,10

jae loop\_bd

cmp al,00h

je end\_l

or al,30h

mov [si],al

end\_l: pop dx

pop cx

ret

BYTE\_TO\_DEC ENDP

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

\_TO\_DEC PROC near

push cx

push dx

push ax

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

\_TO\_DEC ENDP

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

PRINT PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

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

\_AVAILABLEMEMORY PROC NEAR ; Search for available memory

push ax

push bx

push dx

push si

xor ax, ax

mov ah, 04Ah

mov bx, 0FFFFh

int 21h

mov ax, 10h

mul bx

mov si, offset AVAILABLEMEMORY

add si, 23h

call \_TO\_DEC

pop si

pop dx

pop bx

pop ax

ret

\_AVAILABLEMEMORY ENDP

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

\_EXTENDEDMEMORY PROC near ; Search for extended memory

push ax

push bx

push si

push dx

mov al, 30h

out 70h, al

in al, 71h

mov bl, al

mov al, 31h

out 70h, al

in al, 71h

mov ah, al

mov al, bl

sub dx, dx

mov si, offset EXTENDEDMEMORY

add si, 28

call \_TO\_DEC

pop dx

pop si

pop bx

pop ax

ret

\_EXTENDEDMEMORY ENDP

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

\_DATA PROC near ; Search for MCB

mov di, offset DATA ; Address of MCB

mov ax, es

add di, 05h

call WRD\_TO\_HEX

mov di, offset DATA ; Type of MCB

add di, 0Fh

xor ah, ah

mov al, es:[00h]

call BYTE\_TO\_HEX

mov [di], al

inc di

mov [di], ah

mov di, offset DATA ; Owner

mov ax, es:[01h]

add di, 1Dh

call WRD\_TO\_HEX

mov di, offset DATA ; Size

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 2Eh

push si

mov si, di

call \_TO\_DEC

pop si

mov di, offset DATA ; Name

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

\_DATA ENDP

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

OUTPUT PROC NEAR ; Search for a chain of memory management units

mov ah, 52h

int 21h

sub bx, 2h

mov es, es:[bx]

output\_:

call \_DATA

mov dx, offset DATA

call PRINT

mov cx, es

add ax, cx

inc ax

mov es, ax

cmp bl, 4Dh

je output\_

ret

OUTPUT ENDP

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

BEGIN:

call \_AVAILABLEMEMORY

mov dx, offset AVAILABLEMEMORY

call PRINT

call \_EXTENDEDMEMORY

mov dx, offset EXTENDEDMEMORY

call PRINT

mov ah, 4ah ; Freeing of memory

mov bx, offset END\_PROG

int 21h

mov ah, 48h ; Request 64 KB of memory

mov bx, 1000h

int 21h

mov dx, offset HEAD

call PRINT

call OUTPUT

xor al, al

mov ah, 4ch

int 21h

END\_PROG db 0

TESTPC ENDS

END START

ПРИЛОЖЕНИЕ Г

3\_4.ASM

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;data

AVAILABLEMEMORY db ' Amount of available memory: b',0dh,0ah,'$'

EXTENDEDMEMORY db ' Extended memory size: kB',0dh,0ah,'$'

HEAD db ' MCB Adress MCB Type Owner Size Name ', 0dh, 0ah, '$'

DATA db ' ', 0dh, 0ah, '$'

ERRORM db ' Error!', 0dh, 0ah, '$'

;procedurs

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

TETR\_TO\_HEX PROC near

and al,0fh

cmp al,09

jbe NEXT

add al,07

NEXT: add al,30h

ret

TETR\_TO\_HEX ENDP

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

BYTE\_TO\_HEX PROC near

push cx

mov ah,al

call TETR\_TO\_HEX

xchg al,ah

mov cl,4

shr al,cl

call TETR\_TO\_HEX

pop cx

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near

push bx

mov bh,ah

call BYTE\_TO\_HEX

mov [di],ah

dec di

mov [di],al

dec di

mov al,bh

call BYTE\_TO\_HEX

mov [di],ah

dec di

mov [di],al

pop bx

ret

WRD\_TO\_HEX ENDP

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

BYTE\_TO\_DEC PROC near

push cx

push dx

xor ah,ah

xor dx,dx

mov cx,10

loop\_bd:div cx

or dl,30h

mov [si],dl

dec si

xor dx,dx

cmp ax,10

jae loop\_bd

cmp al,00h

je end\_l

or al,30h

mov [si],al

end\_l: pop dx

pop cx

ret

BYTE\_TO\_DEC ENDP

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

\_TO\_DEC PROC near

push cx

push dx

push ax

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

\_TO\_DEC ENDP

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

PRINT PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

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

\_AVAILABLEMEMORY PROC NEAR ; Search for available memory

push ax

push bx

push dx

push si

xor ax, ax

mov ah, 04Ah

mov bx, 0FFFFh

int 21h

mov ax, 10h

mul bx

mov si, offset AVAILABLEMEMORY

add si, 23h

call \_TO\_DEC

pop si

pop dx

pop bx

pop ax

ret

\_AVAILABLEMEMORY ENDP

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

\_EXTENDEDMEMORY PROC near ; Search for extended memory

push ax

push bx

push si

push dx

mov al, 30h

out 70h, al

in al, 71h

mov bl, al

mov al, 31h

out 70h, al

in al, 71h

mov ah, al

mov al, bl

sub dx, dx

mov si, offset EXTENDEDMEMORY

add si, 28

call \_TO\_DEC

pop dx

pop si

pop bx

pop ax

ret

\_EXTENDEDMEMORY ENDP

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

\_DATA PROC near ; Search for MCB

mov di, offset DATA ; Address of MCB

mov ax, es

add di, 05h

call WRD\_TO\_HEX

mov di, offset DATA ; Type of MCB

add di, 0Fh

xor ah, ah

mov al, es:[00h]

call BYTE\_TO\_HEX

mov [di], al

inc di

mov [di], ah

mov di, offset DATA ; Owner

mov ax, es:[01h]

add di, 1Dh

call WRD\_TO\_HEX

mov di, offset DATA ; Size

mov ax, es:[03h]

mov bx, 10h

mul bx

add di, 2Eh

push si

mov si, di

call \_TO\_DEC

pop si

mov di, offset DATA ; Name

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

\_DATA ENDP

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

OUTPUT PROC NEAR ; Search for a chain of memory management units

mov ah, 52h

int 21h

sub bx, 2h

mov es, es:[bx]

output\_:

call \_DATA

mov dx, offset DATA

call PRINT

mov cx, es

add ax, cx

inc ax

mov es, ax

cmp bl, 4Dh

je output\_

ret

OUTPUT ENDP

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

BEGIN:

call \_AVAILABLEMEMORY

mov dx, offset AVAILABLEMEMORY

call PRINT

call \_EXTENDEDMEMORY

mov dx, offset EXTENDEDMEMORY

call PRINT

mov ah, 48h ; Request 64 KB of memory

mov bx, 1000h

int 21h

jc memoryErr ; Error check

jmp next\_

memoryErr:

mov dx, offset ErrorM

call PRINT

next\_: ; Freeing of memory

mov ah, 4ah

mov bx, offset PROGRAMM\_ENDS

int 21h

mov dx, offset HEAD

call PRINT

call OUTPUT

xor al, al

mov ah, 4ch

int 21h

PROGRAMM\_ENDS db 0

TESTPC ENDS

END START