**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Операционные системы»**

Тема: Исследование организации управления основной памятью

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Никитин К.В. |
| Преподаватель |  | Губкин А. Ф. |

Санкт-Петербург

2018

**Постановка задачи.**

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

**Описание функций и структур данных:**

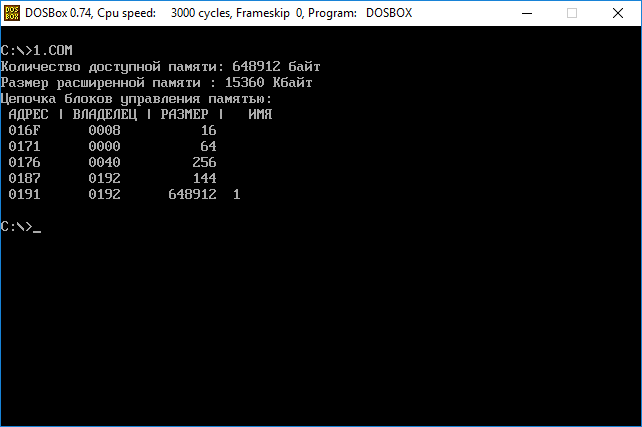
|  |  |
| --- | --- |
| **Название** | **Назначение** |
| WRITEMSG | Вывод сообщения на экран |
| AVV\_MEM | Распечатывает количество доступной памяти |
| EXP\_MEM | Распечатывает размер расширенной памяти |
| MCB | Выводит цепочку блоков управления памятью |
| TETR\_TO\_HEX | Десятичная цифра переводится в код символа |
| BYTE\_TO\_HEX | Байт в AL переводится в два символа шестн. числа в AX |
| WRD\_TO\_HEX | Перевод в 16 с/с 16-ти разрядного числа |
| BYTE\_TO\_DEC | Перевод байта в 16-ной с/с в символьный код в 10-ной с/с |
| TO\_DEC | Переводит dx:ax в 10-ную с/с, SI – адрес поля младшей цифры |

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Назначение** |
| COUNT\_AVV\_MEM\_MSG | db | Выводит количество доступной памяти |
| COUNT\_AVV\_MEM | db | Хранение количества доступной памяти |
| SIZE\_EXP\_MEM\_MSG | db | Выводит размер расширенной памяти |
| SIZE\_EXP\_MEM | db | Хранение размера расширенной памяти |
| BLOCK\_UPR\_MEM\_MSG | db | Выводит цепочку блоков управления памятью |
| BLOCK\_UPR\_MEM | db | Хранение цепочки блоков управления памятью |

Последовательность действий, выполняемых утилитой:

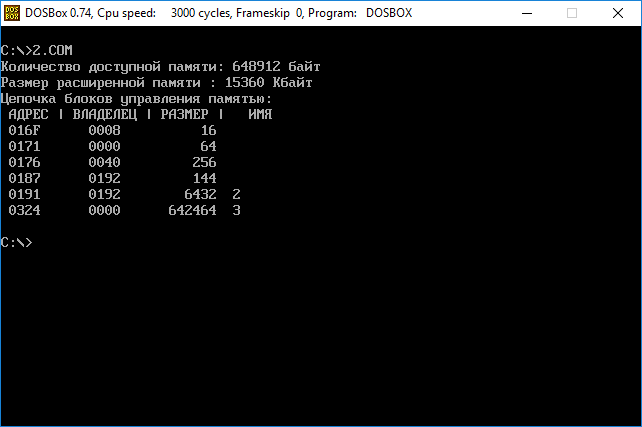
1. Вывод количества доступной памяти.
2. Вывод размера расширенной памяти.
3. Вывод цепочки блоков управления памятью.

Шаг 1:



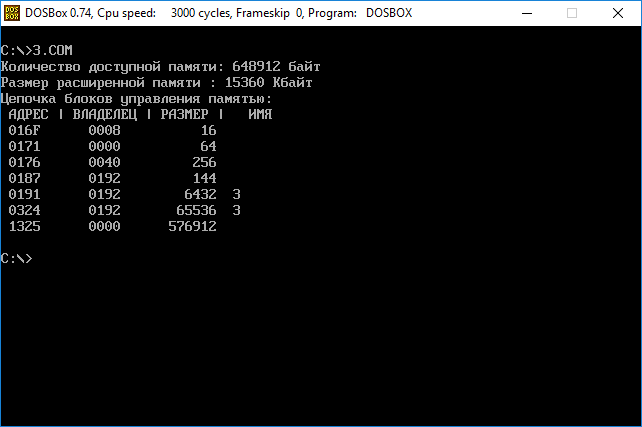
*Рисунок 1. Результат выполнения программы 1.com*

Шаг 2:



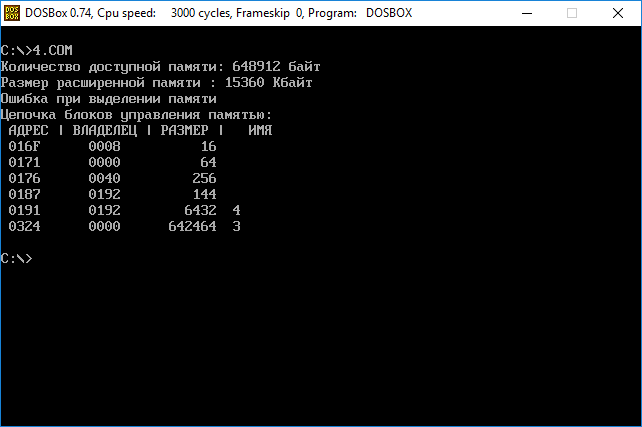
*Рисунок 2. Результат выполнения программы 2.com*

Шаг 3:



*Рисунок 3. Результат выполнения программы 3.com*

Шаг 4:



*Рисунок 4. Результат выполнения программы 4.com*

**Заключение.**

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

**Ответы на контрольные вопросы.**

1. Что означает «доступный объем памяти?»

Доступный объем памяти – это объем памяти, который можно использовать для загрузки пользовательских программ.

1. Где MCB блок вашей программы в списке?

Для первой программы MCB находится на строках 4 и 5, для второй на строках 4 и 5, для третьей на строках 4, 5 и 6 и для четвёртой программы на строках 4 и 5.

1. Какой размер памяти занимает программа в каждом случае?

В первом случае – 648912 Б, во втором – 6432 Б, в третьем – 6432+65536 Б, в четвёртом – 6432 Б.

**ПРИЛОЖЕНИЕ A**

**1.ASM**

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

COUNT\_AVV\_MEM\_MSG db 'Количество доступной памяти: '

COUNT\_AVV\_MEM db ' байт',0DH,0AH,'$'

SIZE\_EXP\_MEM\_MSG db 'Размер расширенной памяти : '

SIZE\_EXP\_MEM db ' Кбайт',0DH,0AH,'$'

BLOCK\_UPR\_MEM\_MSG db 'Цепочка блоков управления памятью: ',0DH,0AH

db ' АДРЕС | ВЛАДЕЛЕЦ | РАЗМЕР | ИМЯ',0DH,0AH,'$'

BLOCK\_UPR\_MEM db ' $'

STRENDL db 0DH,0AH,'$'

WRITEMSG PROC

push ax

mov ah,09h

int 21h

pop ax

ret

WRITEMSG ENDP

AVV\_MEM PROC

mov ax,0

mov ah,4Ah

mov bx,0FFFFh

int 21h

mov ax,bx

mov bx,16

mul bx

mov si,offset COUNT\_AVV\_MEM+5

call TO\_DEC

mov dx,offset COUNT\_AVV\_MEM\_MSG

call WRITEMSG

ret

AVV\_MEM ENDP

EXP\_MEM PROC

mov AL,30h

out 70h,AL

in AL,71h

mov BL,AL

mov AL,31h

out 70h,AL

in AL,71h

mov bh,al

mov ax,bx

mov dx,0

mov si,offset SIZE\_EXP\_MEM+4

call TO\_DEC

mov dx,offset SIZE\_EXP\_MEM\_MSG

call WRITEMSG

ret

EXP\_MEM ENDP

MCB PROC

mov dx,offset BLOCK\_UPR\_MEM\_MSG

call WRITEMSG

push es

mov ah,52h

int 21h

mov bx,es:[bx-2]

mov es,bx

CYCLE:

mov ax,es

mov di,offset BLOCK\_UPR\_MEM+4

call WRD\_TO\_HEX

mov ax,es:[01h]

mov di,offset BLOCK\_UPR\_MEM+14

call WRD\_TO\_HEX

mov ax,es:[03h]

mov si,offset BLOCK\_UPR\_MEM+26

mov dx, 0

mov bx, 10h

mul bx

call TO\_DEC

mov dx,offset BLOCK\_UPR\_MEM

call WRITEMSG

mov cx,8

mov bx,8

mov ah,02h

CYCLE2:

mov dl,es:[bx]

add bx,1

int 21h

loop CYCLE2

mov dx,offset STRENDL

call WRITEMSG

mov ax,es

add ax,1

add ax,es:[03h]

mov bl,es:[00h]

mov es,ax

push bx

mov ax,' '

mov bx,offset BLOCK\_UPR\_MEM

mov [bx+19],ax

mov [bx+21],ax

mov [bx+23],ax

pop bx

cmp bl,4Dh

je CYCLE

pop es

ret

MCB ENDP

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

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

ret

TO\_DEC ENDP

BEGIN:

call AVV\_MEM

call EXP\_MEM

call MCB

xor AL,AL

mov AH,4Ch

int 21H

TESTPC ENDS

END START

**ПРИЛОЖЕНИЕ Б**

**2.ASM**

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

COUNT\_AVV\_MEM\_MSG db 'Количество доступной памяти: '

COUNT\_AVV\_MEM db ' байт',0DH,0AH,'$'

SIZE\_EXP\_MEM\_MSG db 'Размер расширенной памяти : '

SIZE\_EXP\_MEM db ' Кбайт',0DH,0AH,'$'

BLOCK\_UPR\_MEM\_MSG db 'Цепочка блоков управления памятью: ',0DH,0AH

db ' АДРЕС | ВЛАДЕЛЕЦ | РАЗМЕР | ИМЯ',0DH,0AH,'$'

BLOCK\_UPR\_MEM db ' $'

ERRORR\_STR db 'Ошибка',0DH,0AH,'$'

STRENDL db 0DH,0AH,'$'

WRITEMSG PROC

push ax

mov ah,09h

int 21h

pop ax

ret

WRITEMSG ENDP

AVV\_MEM PROC

mov ax,0

mov ah,4Ah

mov bx,0FFFFh

int 21h

mov ax,bx

mov bx,16

mul bx

mov si,offset COUNT\_AVV\_MEM+5

call TO\_DEC

mov dx,offset COUNT\_AVV\_MEM\_MSG

call WRITEMSG

ret

AVV\_MEM ENDP

EXP\_MEM PROC

mov AL,30h

out 70h,AL

in AL,71h

mov BL,AL

mov AL,31h

out 70h,AL

in AL,71h

mov bh,al

mov ax,bx

mov dx,0

mov si,offset SIZE\_EXP\_MEM+4

call TO\_DEC

mov dx,offset SIZE\_EXP\_MEM\_MSG

call WRITEMSG

ret

EXP\_MEM ENDP

MCB PROC

mov bx,0A000h

mov ax,offset ENDING

mov bl,10h

div bl

xor ah,ah

add ax,1

mov bx,cs

add ax,bx

mov bx,es

sub ax,bx

mov al,0

mov ah,4Ah

int 21h

jnc ERRORR

mov dx,offset ERRORR\_STR

call WRITEMSG

ERRORR:

mov dx,offset BLOCK\_UPR\_MEM\_MSG

call WRITEMSG

push es

mov ah,52h

int 21h

mov bx,es:[bx-2]

mov es,bx

CYCLE:

mov ax,es

mov di,offset BLOCK\_UPR\_MEM+4

call WRD\_TO\_HEX

mov ax,es:[01h]

mov di,offset BLOCK\_UPR\_MEM+14

call WRD\_TO\_HEX

mov ax,es:[03h]

mov si,offset BLOCK\_UPR\_MEM+26

mov dx, 0

mov bx, 10h

mul bx

call TO\_DEC

mov dx,offset BLOCK\_UPR\_MEM

call WRITEMSG

mov cx,8

mov bx,8

mov ah,02h

CYCLE2:

mov dl,es:[bx]

add bx,1

int 21h

loop CYCLE2

mov dx,offset STRENDL

call WRITEMSG

mov ax,es

add ax,1

add ax,es:[03h]

mov bl,es:[00h]

mov es,ax

push bx

mov ax,' '

mov bx,offset BLOCK\_UPR\_MEM

mov [bx+19],ax

mov [bx+21],ax

mov [bx+23],ax

pop bx

cmp bl,4Dh

je CYCLE

pop es

ret

MCB ENDP

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

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

ret

TO\_DEC ENDP

BEGIN:

call AVV\_MEM

call EXP\_MEM

call MCB

xor AL,AL

mov AH,4Ch

int 21H

ENDING:

TESTPC ENDS

END START

**ПРИЛОЖЕНИЕ В**

**3.ASM**

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

COUNT\_AVV\_MEM\_MSG db 'Количество доступной памяти: '

COUNT\_AVV\_MEM db ' байт',0DH,0AH,'$'

SIZE\_EXP\_MEM\_MSG db 'Размер расширенной памяти : '

SIZE\_EXP\_MEM db ' Кбайт',0DH,0AH,'$'

BLOCK\_UPR\_MEM\_MSG db 'Цепочка блоков управления памятью: ',0DH,0AH

db ' АДРЕС | ВЛАДЕЛЕЦ | РАЗМЕР | ИМЯ',0DH,0AH,'$'

BLOCK\_UPR\_MEM db ' $'

ERRORR\_STR db 'Ошибка',0DH,0AH,'$'

STRENDL db 0DH,0AH,'$'

WRITEMSG PROC

push ax

mov ah,09h

int 21h

pop ax

ret

WRITEMSG ENDP

AVV\_MEM PROC

mov ax,0

mov ah,4Ah

mov bx,0FFFFh

int 21h

mov ax,bx

mov bx,16

mul bx

mov si,offset COUNT\_AVV\_MEM+5

call TO\_DEC

mov dx,offset COUNT\_AVV\_MEM\_MSG

call WRITEMSG

ret

AVV\_MEM ENDP

EXP\_MEM PROC

mov AL,30h

out 70h,AL

in AL,71h

mov BL,AL

mov AL,31h

out 70h,AL

in AL,71h

mov bh,al

mov ax,bx

mov dx,0

mov si,offset SIZE\_EXP\_MEM+4

call TO\_DEC

mov dx,offset SIZE\_EXP\_MEM\_MSG

call WRITEMSG

ret

EXP\_MEM ENDP

MCB PROC

mov bx,0A000h

mov ax,offset ENDING

mov bl,10h

div bl

xor ah,ah

add ax,1

mov bx,cs

add ax,bx

mov bx,es

sub ax,bx

mov al,0

mov ah,4Ah

int 21h

jnc ERRORR

mov dx,offset ERRORR\_STR

call WRITEMSG

ERRORR:

mov bx,1000h

mov ah,48h

int 21h

mov dx,offset BLOCK\_UPR\_MEM\_MSG

call WRITEMSG

push es

mov ah,52h

int 21h

mov bx,es:[bx-2]

mov es,bx

CYCLE:

mov ax,es

mov di,offset BLOCK\_UPR\_MEM+4

call WRD\_TO\_HEX

mov ax,es:[01h]

mov di,offset BLOCK\_UPR\_MEM+14

call WRD\_TO\_HEX

mov ax,es:[03h]

mov si,offset BLOCK\_UPR\_MEM+26

mov dx, 0

mov bx, 10h

mul bx

call TO\_DEC

mov dx,offset BLOCK\_UPR\_MEM

call WRITEMSG

mov cx,8

mov bx,8

mov ah,02h

CYCLE2:

mov dl,es:[bx]

add bx,1

int 21h

loop CYCLE2

mov dx,offset STRENDL

call WRITEMSG

mov ax,es

add ax,1

add ax,es:[03h]

mov bl,es:[00h]

mov es,ax

push bx

mov ax,' '

mov bx,offset BLOCK\_UPR\_MEM

mov [bx+19],ax

mov [bx+21],ax

mov [bx+23],ax

pop bx

cmp bl,4Dh

je CYCLE

pop es

ret

MCB ENDP

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

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

ret

TO\_DEC ENDP

BEGIN:

call AVV\_MEM

call EXP\_MEM

call MCB

xor AL,AL

mov AH,4Ch

int 21H

ENDING:

TESTPC ENDS

END START

**ПРИЛОЖЕНИЕ Г**

**4.ASM**

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

COUNT\_AVV\_MEM\_MSG db 'Количество доступной памяти: '

COUNT\_AVV\_MEM db ' байт',0DH,0AH,'$'

SIZE\_EXP\_MEM\_MSG db 'Размер расширенной памяти : '

SIZE\_EXP\_MEM db ' Кбайт',0DH,0AH,'$'

BLOCK\_UPR\_MEM\_MSG db 'Цепочка блоков управления памятью: ',0DH,0AH

db ' АДРЕС | ВЛАДЕЛЕЦ | РАЗМЕР | ИМЯ',0DH,0AH,'$'

BLOCK\_UPR\_MEM db ' $'

ERROR\_STR db 'Ошибка',0DH,0AH,'$'

ERROR\_MEM db 'Ошибка при выделении памяти',0DH,0AH,'$'

STRENDL db 0DH,0AH,'$'

WRITEMSG PROC

push ax

mov ah,09h

int 21h

pop ax

ret

WRITEMSG ENDP

AVV\_MEM PROC

mov ax,0

mov ah,4Ah

mov bx,0FFFFh

int 21h

mov ax,bx

mov bx,16

mul bx

mov si,offset COUNT\_AVV\_MEM+5

call TO\_DEC

mov dx,offset COUNT\_AVV\_MEM\_MSG

call WRITEMSG

ret

AVV\_MEM ENDP

EXP\_MEM PROC

mov AL,30h

out 70h,AL

in AL,71h

mov BL,AL

mov AL,31h

out 70h,AL

in AL,71h

mov bh,al

mov ax,bx

mov dx,0

mov si,offset SIZE\_EXP\_MEM+4

call TO\_DEC

mov dx,offset SIZE\_EXP\_MEM\_MSG

call WRITEMSG

ret

EXP\_MEM ENDP

MCB PROC

mov bx,1000h

mov ah,48h

int 21h

jnc ERRORR

mov dx,offset ERROR\_MEM

call WRITEMSG

ERRORR:

mov bx,0A000h

mov ax,offset ENDING

mov bl,10h

div bl

xor ah,ah

add ax,1

mov bx,cs

add ax,bx

mov bx,es

sub ax,bx

mov al,0

mov ah,4Ah

int 21h

jnc ERRORR2

mov dx,offset ERROR\_STR

call WRITEMSG

ERRORR2:

mov dx,offset BLOCK\_UPR\_MEM\_MSG

call WRITEMSG

push es

mov ah,52h

int 21h

mov bx,es:[bx-2]

mov es,bx

CYCLE:

mov ax,es

mov di,offset BLOCK\_UPR\_MEM+4

call WRD\_TO\_HEX

mov ax,es:[01h]

mov di,offset BLOCK\_UPR\_MEM+14

call WRD\_TO\_HEX

mov ax,es:[03h]

mov si,offset BLOCK\_UPR\_MEM+26

mov dx, 0

mov bx, 10h

mul bx

call TO\_DEC

mov dx,offset BLOCK\_UPR\_MEM

call WRITEMSG

mov cx,8

mov bx,8

mov ah,02h

CYCLE2:

mov dl,es:[bx]

add bx,1

int 21h

loop CYCLE2

mov dx,offset STRENDL

call WRITEMSG

mov ax,es

add ax,1

add ax,es:[03h]

mov bl,es:[00h]

mov es,ax

push bx

mov ax,' '

mov bx,offset BLOCK\_UPR\_MEM

mov [bx+19],ax

mov [bx+21],ax

mov [bx+23],ax

pop bx

cmp bl,4Dh

je CYCLE

pop es

ret

MCB ENDP

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

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

ret

TO\_DEC ENDP

BEGIN:

call AVV\_MEM

call EXP\_MEM

call MCB

xor AL,AL

mov AH,4Ch

int 21H

ENDING:

TESTPC ENDS

END START