

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. 8383

Кормщикова А. О.

Преподаватель

Ефремов М. А.

Санкт-Петербург

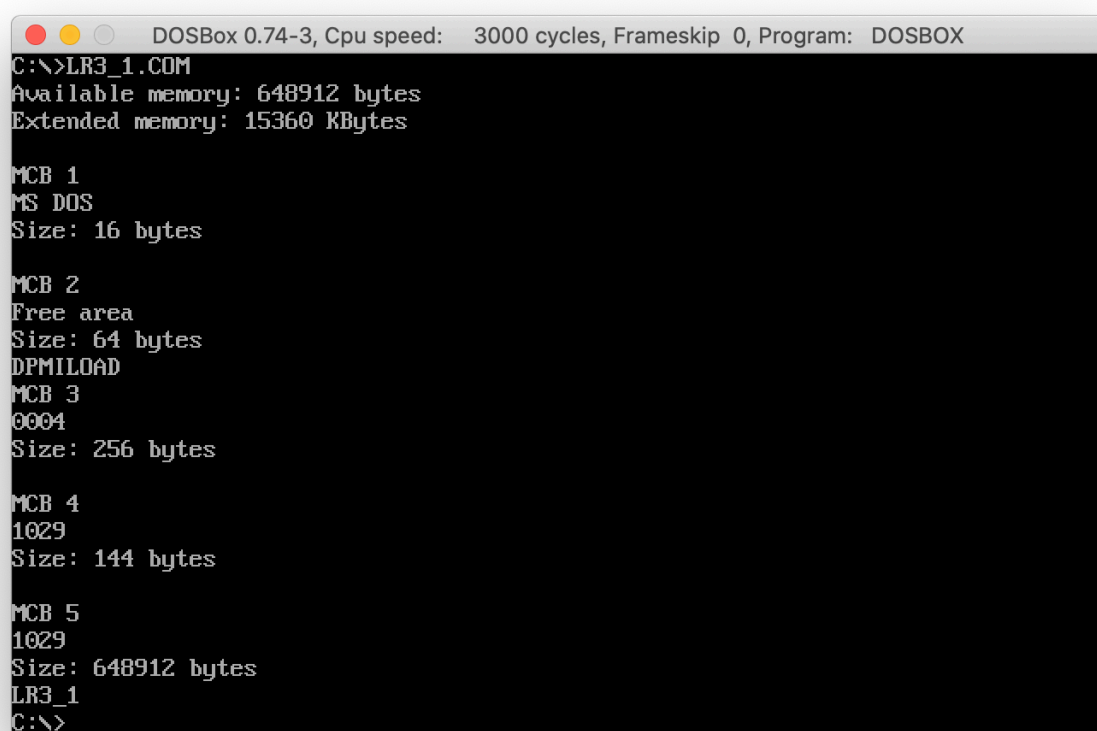
2020

Цель работы.

Исследование структур данных и работы функций управления памятью ядра ОС.

Ход выполнения.

Был написан код исходного .COM модуля (LR3_1.ASM представлен в приложении А), который выбирает и распечатывает следующую информацию: количество доступной памяти, размер расширенной памяти, цепочку блогов управления памятью. Результат работы программы показан на рис. 1.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>LR3_1.COM
Available memory: 648912 bytes
Extended memory: 15360 KBytes

MCB 1
MS DOS
Size: 16 bytes

MCB 2
Free area
Size: 64 bytes
DPMILOAD
MCB 3
0004
Size: 256 bytes

MCB 4
1029
Size: 144 bytes

MCB 5
1029
Size: 648912 bytes
LR3_1
C:\>
```

Рисунок 1 - Результат выполнения исходной программы

Программа была изменена таким образом, что теперь она освобождает память, которую не занимает. Код представлен в приложении Б. Результат выполнения модифицированной программы представлен на рис.2.

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

C:\>LR3_2.COM
Available memory: 648912 bytes
Extended memory: 15360 KBytes

MCB 1
MS DOS
Size: 16 bytes. End:
MCB 2
Free area
Size: 64 bytes. End: DPMILOAD
MCB 3
0004
Size: 256 bytes. End:
MCB 4
1029
Size: 144 bytes. End:
MCB 5
1029
Size: 13696 bytes. End: LR3_2
MCB 6
Free area
Size: 635200 bytes. End:
C:\>
```

Рисунок 2 - Результат выполнения программы

Освобожденная память относится к новому шестому блоку управления памяти.

Далее в программу был добавлен запрос 64Кб памяти после освобождения памяти. Результат работы представлен на рис. 3, код находится в приложении В.

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

C:\>tlink LR3_3.OBJ /t
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

C:\>LR3_3.COM
Available memory: 648912 bytes
Successful memory free
Successful 64KB memory request
Extended memory: 15360 KBytes

MCB1
MS DOS Size: 16 bytes. End:
MCB2
Free area Size: 64 bytes. End: DPMILOAD
MCB3
0004 Size: 256 bytes. End:
MCB4
1029 Size: 144 bytes. End:
MCB5
1029 Size: 15984 bytes. End: LR3_3
MCB6
1029 Size: 65536 bytes. End: LR3_3
MCB7
Free area Size: 567360 bytes. End:
C:\>
```

Рисунок 3 - Результат выполнения программы

Выделенная память относится к 6 блоку, а освобожденная - к 7.

Был изменен порядок запроса памяти и его освобождения. Теперь запрос производится перед освобождением памяти. Результат работы программы представлен на рисунке 4, код расположен в приложении Г.

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Remaining memory: 470k

C:\>tlink LR3_4.OBJ /t
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

C:\>LR3_4.COM
Available memory: 648912 bytes
ERROR 64KB memory request
Successful memory free
Extended memory: 15360 KBytes

MCB1
MS DOS Size: 16 bytes. End:
MCB2
Free area Size: 64 bytes. End: DPMILOAD
MCB3
0004 Size: 256 bytes. End:
MCB4
1029 Size: 144 bytes. End:
MCB5
1029 Size: 15968 bytes. End: LR3_4
MCB6
Free area Size: 632928 bytes. End: ->δ-Borl
C:\>_
```

Рисунок 4 - Результат выполнения программы

Видно, что память не была выделена.

Контрольные вопросы:

1) Что означает "доступный объем памяти"?

Размер памяти, который доступен для выполнения и запуска программ.

2) Где MCB блок Вашей программы в списке?

В 1, 2 и 4 версиях программы это пятый и четвертый блоки. В 3й версии добавляется еще 6 блок, к которому относится выделенная память.

3) Какой размер памяти занимает программа в каждом случае?

1. 649056 байт
2. 13840 байт
3. 16128 + 65536 байт, выделенные дополнительно

4. 16112 байт

Выводы.

В ходе выполнения лабораторной работы были исследованы структуры данных и работа функций управления памятью ядра ОС.

ПРИЛОЖЕНИЕ А

```

LR SEGMENT
    ASSUME  CS:LR, DS:LR, ES:NOTHING, SS:NOTHING
    ORG     100H
START:     JMP     BEGIN

;DATA

    AV_MEM db 'Available memory: $'
    BYTES db ' bytes', 0DH, 0AH, '$'
    NEW_LINE db 0DH, 0AH, '$'
    EX_MEM db 'Extended memory: $'
    KB db ' KBytes ', 0DH, 0AH, '$'
    MCB db 0DH, 0AH, 'MCB $'
    FRA db 0DH, 0AH, 'Free area $'
    OSXMS db 0DH, 0AH, 'OS XMS UMB $'
    TOPMEM db 0DH, 0AH, 'Top driver memory $'
    MSD db 0DH, 0AH, 'MS DOS $'
    UMB_block db 0DH, 0AH, 'Control block 386MAX UMB $'
    UMB_blocked db 0DH, 0AH, 'Blocked 386MAX $'
    UMB_belongs db 0DH, 0AH, 'Belongs 386MAX UMB $'
    SIZE__ db 0DH, 0AH, 'Size: $'

;-----

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 в 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 цифра
    pop     CX           ;в AH младшая
    ret
BYTE_TO_HEX ENDP
;-----

PRINT_DEC PROC near
    push ax
    push bx
    push cx
    push dx

    xor cx, cx
    mov bx, 10

loop:
    div bx
    push dx
    xor dx, dx
    inc cx
    cmp ax, 00h

```

```

        jne loop

        mov ah, 02h

print:
        pop dx
        or dl, 30h
        int 21h
        loop print

        pop dx
        pop cx
        pop bx
        pop ax
        ret

PRINT_DEC ENDP

PRINT_LINE PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_LINE ENDP

PRINT_SYMBOL PROC near
        push ax
        mov ah, 02h
        int 21h
        pop ax
        ret
PRINT_SYMBOL ENDP

PRINT_HEX PROC near

        push ax
        push ax
        mov al, ah
        call BYTE_TO_HEX
        mov dl, ah
        call PRINT_SYMBOL
        mov dl, al
        call PRINT_SYMBOL
        pop ax
        call BYTE_TO_HEX
        mov dl, ah
        call PRINT_SYMBOL
        mov dl, al
        call PRINT_SYMBOL
        pop ax
        ret
PRINT_HEX ENDP

BEGIN:

;AVAILABLE MEM
        mov dx, offset AV_MEM
        call PRINT_LINE
        mov ah, 4Ah
        mov bx, 0ffffh

```

```

        int 21h
        mov ax, bx
        mov bx, 10h
        mul bx
        call PRINT_DEC
        mov dx, offset BYTES
        call PRINT_LINE

;EXTENDED MEM
        mov dx, offset EX_MEM
        call PRINT_LINE

        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
        xor dx, dx
        call PRINT_DEC
        mov dx, offset KB
        call PRINT_LINE

;MEMORY CONTROL BLOCK

        mov ah, 52h
        int 21h

        mov ax, es:[bx-2]
        mov es, ax
        xor cx, cx
listMCB:

        inc cx
        mov dx, offset MCB
        call PRINT_LINE
        mov ax, cx
        xor dx, dx
        call PRINT_DEC
        mov ax, es:[01h]

        cmp ax, 00h
        je FREE_AREA
        cmp ax, 06h
        je OS_XMS_UMB
        cmp ax, 07h
        je TOP_DRIVER_MEMORY
        cmp ax, 08h
        je MS_DOS
        cmp ax, 0FFFAh
        je UMB_CONTROL
        cmp ax, 0FFFDh
        je UMB_BLOCKED
        cmp ax, 0FFFEh
        je UMB_BELNGS

        mov dx, offset NEW_LINE
        call PRINT_LINE
        call PRINT_HEX
        jmp SIZE_

```



```

FREE_AREA:
    mov dx, offset FRA
    jmp PRINT_OWNER

OS_XMS_UMB:
    mov dx, offset OSXMS
    jmp PRINT_OWNER

TOP_DRIVER_MEMORY:
    mov dx, offset TOPMEM
    jmp PRINT_OWNER

MS_DOS:
    mov dx, offset MSD
    jmp PRINT_OWNER

UMB_CONTROL:
    mov dx, offset UMB_block
    jmp PRINT_OWNER

UMB_BLOCKED:
    mov dx, offset UMB_blocked
    jmp PRINT_OWNER

UMB_BELNGS:
    mov dx, offset UMB_belongs
    jmp PRINT_OWNER

PRINT_OWNER:
    call PRINT_LINE

SIZE_:
    mov dx, offset SIZE__
    call PRINT_LINE
    mov ax, es:[3h]
    mov bx, 10h
    mul bx
    call PRINT_DEC
    mov dx, offset BYTES
    call PRINT_LINE

    mov ah, 02h
    xor si, si
    push cx
    mov cx, 8

LAST_LOOP:
    mov dl, es:[si+8h]
    int 21h
    inc si
    loop LAST_LOOP
    pop cx

    mov ax, es:[00h]
    cmp al, 5Ah
    je EXIT

    mov ax, es:[03h]
    mov bx, es
    add bx, ax
    inc bx
    mov es, bx
    jmp listMCB

```

```
;exit to dos
EXIT:
    xor AL,AL
    mov AH,4Ch
    int 21H

LR      ENDS
END START
```

ПРИЛОЖЕНИЕ Б

```

LR SEGMENT
    ASSUME  CS:LR, DS:LR, ES:NOTHING, SS:NOTHING
    ORG     100H
START:     JMP     BEGIN

;DATA

    AV_MEM db 'Available memory: $'
    BYTES db ' bytes $'
    BTEND db ' bytes. End: $'

    NEW_LINE db 0DH, 0AH, '$'
    EX_MEM db 0DH, 0AH, 'Extended memory: $'
    KB db ' KBytes ', 0DH, 0AH, '$'
    MCB db 0DH, 0AH, 'MCB $'
    FRA db 0DH, 0AH, 'Free area $'
    OSXMS db 0DH, 0AH, 'OS XMS UMB $'
    TOPMEM db 0DH, 0AH, 'Top driver memory $'
    MSD db 0DH, 0AH, 'MS DOS $'
    UMB_block db 0DH, 0AH, 'Control block 386MAX UMB $'
    UMB_blocked db 0DH, 0AH, 'Blocked 386MAX $'
    UMB_belongs db 0DH, 0AH, 'Belongs 386MAX UMB $'
    SIZE__ db 0DH, 0AH, 'Size: $'

;-----

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 в 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 цифра
    pop     CX           ;в AH младшая
    ret
BYTE_TO_HEX ENDP
;-----

PRINT_DEC PROC near
    push ax
    push bx
    push cx
    push dx

    xor cx, cx
    mov bx, 10

loop:
    div bx
    push dx
    xor dx, dx

```

```

        inc cx
        cmp ax, 00h
        jne loop

        mov ah, 02h

print:
        pop dx
        or dl, 30h
        int 21h
        loop print

        pop dx
        pop cx
        pop bx
        pop ax
        ret

PRINT_DEC ENDP

PRINT_LINE PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_LINE ENDP

PRINT_SYMBOL PROC near
        push ax
        mov ah, 02h
        int 21h
        pop ax
        ret
PRINT_SYMBOL ENDP

PRINT_HEX PROC near
        push ax
        push ax
        mov al, ah
        call BYTE_TO_HEX
        mov dl, ah
        call PRINT_SYMBOL
        mov dl, al
        call PRINT_SYMBOL
        pop ax
        call BYTE_TO_HEX
        mov dl, ah
        call PRINT_SYMBOL
        mov dl, al
        call PRINT_SYMBOL
        pop ax
        ret
PRINT_HEX ENDP

BEGIN:

;AVAILABLE MEM
        mov dx, offset AV_MEM
        call PRINT_LINE

```

```

    mov ah, 4Ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov bx, 10h
    mul bx
    call PRINT_DEC
    mov dx, offset BYTES
    call PRINT_LINE

;FREE MEM
    mov ah, 4Ah
    mov bx, offset LREND
    int 21h

;EXTENDED MEM
    mov dx, offset EX_MEM
    call PRINT_LINE

    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
    xor dx, dx
    call PRINT_DEC
    mov dx, offset KB
    call PRINT_LINE

;MEMORY CONTROL BLOCK

    mov ah, 52h
    int 21h

    mov ax, es:[bx-2]
    mov es, ax
    xor cx, cx
listMCB:

    inc cx
    mov dx, offset MCB
    call PRINT_LINE
    mov ax, cx
    xor dx, dx
    call PRINT_DEC
    mov ax, es:[01h]

    cmp ax, 00h
    je FREE_AREA
    cmp ax, 06h
    je OS_XMS_UMB
    cmp ax, 07h
    je TOP_DRIVER_MEMORY
    cmp ax, 08h
    je MS_DOS
    cmp ax, 0FFFAh
    je UMB_CONTROL
    cmp ax, 0FFFDh
    je UMB_BLOCKED
    cmp ax, 0FFFEh

```

```

        je UMB_BELNGS

        mov dx, offset NEW_LINE
        call PRINT_LINE
        call PRINT_HEX
        jmp SIZE_

FREE_AREA:
        mov dx, offset FRA
        jmp PRINT_OWNER

OS_XMS_UMB:
        mov dx, offset OSXMS
        jmp PRINT_OWNER

TOP_DRIVER_MEMORY:
        mov dx, offset TOPMEM
        jmp PRINT_OWNER

MS_DOS:
        mov dx, offset MSD
        jmp PRINT_OWNER

UMB_CONTROL:
        mov dx, offset UMB_block
        jmp PRINT_OWNER

UMB_BLOCKED:
        mov dx, offset UMB_blocked
        jmp PRINT_OWNER

UMB_BELNGS:
        mov dx, offset UMB_belongs
        jmp PRINT_OWNER

PRINT_OWNER:
        call PRINT_LINE

SIZE_:
        mov dx, offset SIZE__
        call PRINT_LINE
        mov ax, es:[3h]
        mov bx, 10h
        mul bx
        call PRINT_DEC
        mov dx, offset BTEND
        call PRINT_LINE

        mov ah, 02h
        xor si, si
        push cx
        mov cx, 8

LAST_LOOP:
        mov dl, es:[si+8h]
        int 21h
        inc si
        loop LAST_LOOP
        pop cx

        mov ax, es:[00h]
        cmp al, 5Ah
        je EXIT

```

```
        mov ax, es:[03h]
        mov bx, es
        add bx, ax
        inc bx
        mov es, bx
        jmp listMCB

;exit to dos
EXIT:
        xor AL,AL
        mov AH,4Ch
        int 21H

LREND:
LR      ENDS
END START
```

ПРИЛОЖЕНИЕ В

```

LR SEGMENT
    ASSUME  CS:LR, DS:LR, ES:NOTHING, SS:NOTHING
    ORG     100H
START:     JMP     BEGIN

;DATA

    AV_MEM db 'Available memory: $'
    BYTES db ' bytes $'
    BTEND db ' bytes. End: $'

    NEW_LINE db 0DH, 0AH, '$'
    EX_MEM db 0DH, 0AH, 'Extended memory: $'
    KB db ' KBytes ', 0DH, 0AH, '$'
    MCB db 0DH, 0AH, 'MCB$'
    FRA db 0DH, 0AH, 'Free area $'
    OSXMS db 0DH, 0AH, 'OS XMS UMB $'
    TOPMEM db 0DH, 0AH, 'Top driver memory $'
    MSD db 0DH, 0AH, 'MS DOS $'
    UMB_block db 0DH, 0AH, 'Control block 386MAX UMB $'
    UMB_blocked db 0DH, 0AH, 'Blocked 386MAX $'
    UMB_belongs db 0DH, 0AH, 'Belongs 386MAX UMB $'
    SIZE__ db ' Size: $'

    FREE_S db 0DH, 0AH, 'Successful memory free $'
    FREE_E db 0DH, 0AH, 'Memory free ERROR $'
    REQUEST_S db 0DH, 0AH, 'Successful 64KB memory request $'
    REQUEST_E db 0DH, 0AH, 'ERROR 64KB memory request $'

;-----

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 в 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 цифра
    pop     CX          ;в AH младшая
    ret
BYTE_TO_HEX ENDP

;-----

PRINT_DEC PROC near
    push ax
    push bx
    push cx
    push dx

    xor cx, cx
    mov bx, 10

```



```

loop:
    div bx
    push dx
    xor dx, dx
    inc cx
    cmp ax, 00h
    jne loop

    mov ah, 02h

print:
    pop dx
    or dl, 30h
    int 21h
    loop print

    pop dx
    pop cx
    pop bx
    pop ax
    ret

PRINT_DEC ENDP

PRINT_LINE PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT_LINE ENDP

PRINT_SYMBOL PROC near
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
PRINT_SYMBOL ENDP

PRINT_HEX PROC near

    push ax
    push ax
    mov al, ah
    call BYTE_TO_HEX
    mov dl, ah
    call PRINT_SYMBOL
    mov dl, al
    call PRINT_SYMBOL
    pop ax
    call BYTE_TO_HEX
    mov dl, ah
    call PRINT_SYMBOL
    mov dl, al
    call PRINT_SYMBOL
    pop ax
    ret
PRINT_HEX ENDP

BEGIN:

```

```

;AVAILABLE MEM
    mov dx, offset AV_MEM
    call PRINT_LINE
    mov ah, 4Ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov bx, 10h
    mul bx
    call PRINT_DEC
    mov dx, offset BYTES
    call PRINT_LINE

;FREE MEM
    mov ah, 4Ah
    mov bx, offset LREND
    int 21h

    jnc F_S
    mov dx, offset FREE_E
    call PRINT_LINE
    jmp END_F
F_S:    mov dx, offset FREE_S
        call PRINT_LINE
END_F:

;REQUEST
    mov ah, 48h
    mov bx, 1000h
    int 21h
    jnc R_S
    mov dx, offset REQUEST_E
    call PRINT_LINE
    jmp END_R
R_S:    mov dx, offset REQUEST_S
        call PRINT_LINE
END_R:

;EXTENDED MEM
    mov dx, offset EX_MEM
    call PRINT_LINE

    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
    xor dx, dx
    call PRINT_DEC
    mov dx, offset KB
    call PRINT_LINE

;MEMORY CONTROL BLOCK

```

```

        mov ah, 52h
        int 21h

        mov ax, es:[bx-2]
        mov es, ax
        xor cx, cx
listMCB:

        inc cx
        mov dx, offset MCB
        call PRINT_LINE
        mov ax, cx
        xor dx, dx
        call PRINT_DEC
        mov ax, es:[01h]

        cmp ax, 00h
        je FREE_AREA
        cmp ax, 06h
        je OS_XMS_UMB
        cmp ax, 07h
        je TOP_DRIVER_MEMORY
        cmp ax, 08h
        je MS_DOS
        cmp ax, 0FFFAh
        je UMB_CONTROL
        cmp ax, 0FFFDh
        je UMB_BLOCKED
        cmp ax, 0FFFEh
        je UMB_BELNGS

        mov dx, offset NEW_LINE
        call PRINT_LINE
        call PRINT_HEX
        jmp SIZE_

FREE_AREA:
        mov dx, offset FRA
        jmp PRINT_OWNER

OS_XMS_UMB:
        mov dx, offset OSXMS
        jmp PRINT_OWNER

TOP_DRIVER_MEMORY:
        mov dx, offset TOPMEM
        jmp PRINT_OWNER

MS_DOS:
        mov dx, offset MSD
        jmp PRINT_OWNER

UMB_CONTROL:
        mov dx, offset UMB_block
        jmp PRINT_OWNER

UMB_BLOCKED:
        mov dx, offset UMB_blocked
        jmp PRINT_OWNER

UMB_BELNGS:
        mov dx, offset UMB_belongs
        jmp PRINT_OWNER

```

```

PRINT_OWNER:
    call PRINT_LINE

SIZE_:
    mov dx, offset SIZE__
    call PRINT_LINE
    mov ax, es:[3h]
    mov bx, 10h
    mul bx
    call PRINT_DEC
    mov dx, offset BTEND
    call PRINT_LINE

    mov ah, 02h
    xor si, si
    push cx
    mov cx, 8

LAST_LOOP:
    mov dl, es:[si+8h]
    int 21h
    inc si
    loop LAST_LOOP
    pop cx

    mov ax, es:[00h]
    cmp al, 5Ah
    je EXIT

    mov ax, es:[03h]
    mov bx, es
    add bx, ax
    inc bx
    mov es, bx
    jmp listMCB

;exit to dos
EXIT:
    xor AL,AL
    mov AH,4Ch
    int 21H
LREND:
LR      ENDS
END START

```

ПРИЛОЖЕНИЕ Г

```

LR SEGMENT
    ASSUME CS:LR, DS:LR, ES:NOTHING, SS:NOTHING
    ORG 100H
START:    JMP BEGIN

;DATA

    AV_MEM db 'Available memory: $'
    BYTES db ' bytes $'
    BTEND db ' bytes. End: $'

    NEW_LINE db 0DH, 0AH, '$'
    EX_MEM db 0DH, 0AH, 'Extended memory: $'
    KB db ' KBytes ', 0DH, 0AH, '$'
    MCB db 0DH, 0AH, 'MCB$'
    FRA db 0DH, 0AH, 'Free area $'
    OSXMS db 0DH, 0AH, 'OS XMS UMB $'
    TOPMEM db 0DH, 0AH, 'Top driver memory $'
    MSD db 0DH, 0AH, 'MS DOS $'
    UMB_block db 0DH, 0AH, 'Control block 386MAX UMB $'
    UMB_blocked db 0DH, 0AH, 'Blocked 386MAX $'
    UMB_belongs db 0DH, 0AH, 'Belongs 386MAX UMB $'
    SIZE__ db ' Size: $'

    FREE_S db 0DH, 0AH, 'Successful memory free $'
    FREE_E db 0DH, 0AH, 'Memory free ERROR $'
    REQUEST_S db 0DH, 0AH, 'Successful 64KB memory request $'
    REQUEST_E db 0DH, 0AH, 'ERROR 64KB memory request $'
;-----

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 в 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 цифра
    pop CX ;в AH младшая
    ret
BYTE_TO_HEX ENDP
;-----

PRINT_DEC PROC near
    push ax
    push bx
    push cx
    push dx

    xor cx, cx
    mov bx, 10

```

```

loop:
    div bx
    push dx
    xor dx, dx
    inc cx
    cmp ax, 00h
    jne loop

    mov ah, 02h

print:
    pop dx
    or dl, 30h
    int 21h
    loop print

    pop dx
    pop cx
    pop bx
    pop ax
    ret

PRINT_DEC ENDP

PRINT_LINE PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT_LINE ENDP

PRINT_SYMBOL PROC near
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
PRINT_SYMBOL ENDP

PRINT_HEX PROC near

    push ax
    push ax
    mov al, ah
    call BYTE_TO_HEX
    mov dl, ah
    call PRINT_SYMBOL
    mov dl, al
    call PRINT_SYMBOL
    pop ax
    call BYTE_TO_HEX
    mov dl, ah
    call PRINT_SYMBOL
    mov dl, al
    call PRINT_SYMBOL
    pop ax
    ret
PRINT_HEX ENDP

BEGIN:

```

```

;AVAILABLE MEM
    mov dx, offset AV_MEM
    call PRINT_LINE
    mov ah, 4Ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov bx, 10h
    mul bx
    call PRINT_DEC
    mov dx, offset BYTES
    call PRINT_LINE

;REQUEST
    mov ah, 48h
    mov bx, 1000h
    int 21h
    jnc R_S
    mov dx, offset REQUEST_E
    call PRINT_LINE
    jmp END_R
R_S:
    mov dx, offset REQUEST_S
    call PRINT_LINE
END_R:

;FREE MEM
    mov ah, 4Ah
    mov bx, offset LREND
    int 21h

    jnc F_S
    mov dx, offset FREE_E
    call PRINT_LINE
    jmp END_F
F_S:
    mov dx, offset FREE_S
    call PRINT_LINE
END_F:

;EXTENDED MEM
    mov dx, offset EX_MEM
    call PRINT_LINE

    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
    xor dx, dx
    call PRINT_DEC
    mov dx, offset KB
    call PRINT_LINE

;MEMORY CONTROL BLOCK

    mov ah, 52h
    int 21h

```

```

        mov ax, es:[bx-2]
        mov es, ax
        xor cx, cx
listMCB:

        inc cx
        mov dx, offset MCB
        call PRINT_LINE
        mov ax, cx
        xor dx, dx
        call PRINT_DEC
        mov ax, es:[01h]

        cmp ax, 00h
        je FREE_AREA
        cmp ax, 06h
        je OS_XMS_UMB
        cmp ax, 07h
        je TOP_DRIVER_MEMORY
        cmp ax, 08h
        je MS_DOS
        cmp ax, 0FFFAh
        je UMB_CONTROL
        cmp ax, 0FFFDh
        je UMB_BLOCKED
        cmp ax, 0FFFEh
        je UMB_BELNGS

        mov dx, offset NEW_LINE
        call PRINT_LINE
        call PRINT_HEX
        jmp SIZE_

FREE_AREA:
        mov dx, offset FRA
        jmp PRINT_OWNER

OS_XMS_UMB:
        mov dx, offset OSXMS
        jmp PRINT_OWNER

TOP_DRIVER_MEMORY:
        mov dx, offset TOPMEM
        jmp PRINT_OWNER

MS_DOS:
        mov dx, offset MSD
        jmp PRINT_OWNER

UMB_CONTROL:
        mov dx, offset UMB_block
        jmp PRINT_OWNER

UMB_BLOCKED:
        mov dx, offset UMB_blocked
        jmp PRINT_OWNER

UMB_BELNGS:
        mov dx, offset UMB_belongs
        jmp PRINT_OWNER

PRINT_OWNER:
        call PRINT_LINE

```



```

SIZE_:
    mov dx, offset SIZE__
    call PRINT_LINE
    mov ax, es:[3h]
    mov bx, 10h
    mul bx
    call PRINT_DEC
    mov dx, offset BTEND
    call PRINT_LINE

    mov ah, 02h
    xor si, si
    push cx
    mov cx, 8

LAST_LOOP:
    mov dl, es:[si+8h]
    int 21h
    inc si
    loop LAST_LOOP
    pop cx

    mov ax, es:[00h]
    cmp al, 5Ah
    je EXIT

    mov ax, es:[03h]
    mov bx, es
    add bx, ax
    inc bx
    mov es, bx
    jmp listMCB

;exit to dos
EXIT:
    xor AL,AL
    mov AH,4Ch
    int 21H

LREND:
LR      ENDS
END START

```