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. 8382	_____	Колногоров Д.Г.
Преподаватель	_____	Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

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

Выполнение работы.

Был написан программный модуль типа **.COM** (представлен в приложении А), который определяет и распечатывает следующую информацию:

1) Количество доступной памяти.

2) Размер расширенной памяти.

3) Выводит цепочку блоков управления памятью.

Результат исполнения COM модуля представлен на рисунке 1.

```
available memory:648912 bytes
extended memory: 15360 bytes

01
owner: MS DOS
size:     16 bytes
last bytes:

02
owner: free
size:     64 bytes
last bytes:

03
owner: 0040
size:    256 bytes
last bytes:

04
owner: 0192
size:    144 bytes
last bytes:

05
owner: 0192
size: 648912 bytes
last bytes: LR3 1
```

Рисунок 1 — результат исполнения первого COM модуля

Далее программа была изменена таким образом, чтобы она освобождала память, которую она не занимает. Исходный код представлен в приложении Б, а результат исполнения — на рисунке 2.

```
available memory:648912 bytes
extended memory: 15360 bytes

successful free

01
owner: MS DOS
size:      16 bytes
last bytes:

02
owner: free
size:      64 bytes
last bytes:

03
owner: 0040
size:     256 bytes
last bytes:

04
owner: 0192
size:     144 bytes
last bytes:

05
owner: 0192
size:     928 bytes
last bytes: LR3 2

06
owner: free
size: 647968 bytes
last bytes: ÀuÛt9
```

Рисунок 2 — результат исполнения второго COM модуля

Далее программа была изменена таким образом, чтобы после освобождения памяти программа запрашивала 64Кб памяти. Исходный код представлен в приложении В, а результат исполнения — на рисунке 3.

```
available memory:648912 bytes
extended memory: 15360 bytes

successful free
successful allocation

01
owner: MS DOS
size:    16 bytes
last bytes:

02
owner: free
size:    64 bytes
last bytes:

03
owner: 0040
size:   256 bytes
last bytes:

04
owner: 0192
size:   144 bytes
last bytes:

05
owner: 0192
size:  1008 bytes
last bytes: LR3_3

06
owner: 0192
size: 65536 bytes
last bytes: LR3_3

07
owner: free
size:582336 bytes
last bytes: 1error
```

Рисунок 3 — результат исполнения третьего COM модуля

Далее программа была изменена таким образом, чтобы память запрашивалась до освобождения. Исходный код представлен в приложении Г, а результат исполнения — на рисунке 4.

```
available memory:648912 bytes
extended memory: 15360 bytes

unsuccessful allocation
successful free

01
owner: MS DOS
size:     16   bytes
last bytes:

02
owner: free
size:     64   bytes
last bytes:

03
owner: 0040
size:    256   bytes
last bytes:

04
owner: 0192
size:    144   bytes
last bytes:

05
owner: 0192
size:   1008   bytes
last bytes: LR3 4

06
owner: free
size: 647888   bytes
last bytes: %°&8%u
```

Рисунок 4 — результат исполнения четвертого COM модуля

Контрольные вопросы.

1) Что означает «доступный объём памяти»?

Это максимальный размер памяти, который может быть выделен программе.

2) Где MCB блок Вашей программы в списке?

В первой, второй и четвёртой программах блоком программы является пятый блок. В третьей программе — это пятый и шестой.

3) Какой размер занимает программа в каждом случае?

Первая — занимает все доступные 648912 байта.

Вторая — занимает 928 байт, остальное было освобождено.

Третья — занимает 1008 байт + выделенные 64Кб.

Четвёртая — занимает 1008 байт.

Вывод.

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

СОДЕРЖИМОЕ ФАЙЛА LAB3_1.ASM

```
TESTPC      SEGMENT
            ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
            ORG     100H


START:      JMP     BEGIN
; data
NEW_LINE            db 10,13,'$'
STR_MEM_AVAILABLE db 'available memory:        bytes',10,13,'$'
STR_MEM_EXTENDED  db 'extended memory:        bytes',10,13,'$'
STR_MCB_SIZE      db 'size:          bytes',10,13,'$'
STR_OWNER         db 'owner: $'
STR_LAST_BYTES    db 'last bytes: $'
STR_FREE          db 'free$'
STR_OSXMSUBM      db 'OS XMS UMB$'
STR_TOP_MEM       db "driver's top memory$"
STR_MSDOS         db 'MS DOS$'
STR_TAKEN386      db "386MAX UMB's block$"
STR_BLOCKED386    db 'blocked by 386MAX$'
STR_OWNED386      db '386MAX UMB$'


PRINT_NEW_LINE  PROC near
    push DX
    push AX

    mov DX, offset NEW_LINE
    mov AH, 09h
    int 21h

    pop AX
    pop DX
    ret
PRINT_NEW_LINE  ENDP


PRINT_BYTE      PROC near
; prints AL as two hex digits
    push BX
    push DX
```

```
        call BYTE_TO_HEX
        mov BH, AH

        mov DL, AL
        mov AH, 02h
        int 21h

        mov DL, BH
        mov AH, 02h
        int 21h

        pop DX
        pop BX
        ret
PRINT_BYTE     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
; AL --> two hex symbols in AX
        push     CX
        mov      AH,AL
        call     TETR_TO_HEX
        xchg     AL,AH
        mov      CL,4
        shr      AL,CL
        call     TETR_TO_HEX ; AL - high digit
        pop      CX          ; AH - low digit
        ret
BYTE_TO_HEX   ENDP
```

```
WORD_TO_DEC PROC near
; convert AX to dec, SI - adress of low digit
            push      AX
            push      BX
            push      CX
            push      DX
            mov       CX,10
loop_bd2:   div       CX
            or        DL,30h
            mov       [SI],DL
            dec       SI
            xor       DX,DX
            cmp       AX,0
            jnz       loop_bd2
end_l2:     pop       DX
            pop       CX
            pop       BX
            pop       AX
            ret
WORD_TO_DEC ENDP


; CODE
BEGIN:


PRINT_AVAILABLE_MEMORY:
     mov AH, 4Ah
     mov BX, 0FFFFh
     int 21h
     mov AX, BX
     mov BX, 10h
     mul BX

     mov SI, offset STR_MEM_AVAILABLE
     add SI, 22
     call WORD_TO_DEC
     mov DX, offset STR_MEM_AVAILABLE
     mov AH, 09h
     int 21h


PRINT_EXTENDED_MEMORY:
```

```asm
        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


        mov SI, offset STR_MEM_EXTENDED
        add SI, 21
        xor DX, DX
        call WORD_TO_DEC
        mov DX, offset STR_MEM_EXTENDED
        mov AH, 09h
        int 21h


PRINT_MCBS:
        ; get first mcb's address
        mov AH, 52h
        int 21h
        mov AX, ES:[BX-2]
        mov ES, AX
        mov CX, 0


        NEXT_MCB:
            call PRINT_NEW_LINE
            inc CX
            mov AL, CL
            call PRINT_BYTE
            call PRINT_NEW_LINE


            mov DX, offset STR_OWNER
            mov AH, 09h
            int 21h


            ; get owner
            mov BX, ES:[1h]
```

```
                  ; match owner
                  mov DX, offset STR_FREE
                  cmp BX, 0000h
                  je MCB_MATCHED
                  mov DX, offset STR_OSXMSUBM
                  cmp BX, 0006h
                  je MCB_MATCHED
                  mov DX, offset STR_TOP_MEM
                  cmp BX, 0007h
                  je MCB_MATCHED
                  mov DX, offset STR_MSDOS
                  cmp BX, 0008h
                  je MCB_MATCHED
                  mov DX, offset STR_TAKEN386
                  cmp BX, 0FFFAh
                  je MCB_MATCHED
                  mov DX, offset STR_BLOCKED386
                  cmp BX, 0FFFDh
                  je MCB_MATCHED
                  mov DX, offset STR_OWNED386
                  cmp BX, 0FFFEh
                  je MCB_MATCHED


                  jmp MCB_NOT_MATCHED


                  ; print owner
                  MCB_MATCHED:
                        mov AH, 09h
                        int 21h
                        jmp MCB_MATCH_END
                  MCB_NOT_MATCHED:
                        mov AL, BH
                        call PRINT_BYTE
                        mov AL, BL
                        call PRINT_BYTE


                  MCB_MATCH_END:
                  call PRINT_NEW_LINE


                  ; get size
                  mov AX, ES:[3h]
```

```
mov BX, 10h
mul BX


; print size
mov SI, offset STR_MCB_SIZE
add SI, 11
call WORD_TO_DEC
mov DX, offset STR_MCB_SIZE
mov AH, 09h
int 21h


; print last 8 bytes
mov DX, offset STR_LAST_BYTES
mov AH, 09h
int 21h


push CX
mov CX, 8
mov BX, 0
mov AH, 02h
PRINT_LAST_BYTES:
      mov DL, ES:[BX+8h]
      int 21h
      inc BX
      loop PRINT_LAST_BYTES
      call PRINT_NEW_LINE
pop CX


; check if last block
mov AL, ES:[0h]
cmp AL, 5Ah
je PRINT_MCBS_END


; get next block's address
mov AX, ES:[3h]
mov BX, ES
add BX, AX
inc BX
mov ES, BX
```

```
        jmp NEXT_MCB


    PRINT_MCBS_END:




; return to DOS
        xor     AL,AL
        mov     AH,4Ch
        int     21H
TESTPC   ENDS
        END     START      ; module end START - entry point
```

ПРИЛОЖЕНИЕ Б

СОДЕРЖИМОЕ ФАЙЛА LAB3_2.ASM

```
        TESTPC     SEGMENT
               ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
               ORG     100H


START:     JMP     BEGIN
; data
NEW_LINE          db 10,13,'$'
STR_MEM_AVAILABLE db 'available memory:       bytes',10,13,'$'
STR_MEM_EXTENDED  db 'extended memory:        bytes',10,13,'$'
STR_FREE_SUCCESS  db 'successful free',10,13,'$'
STR_FREE_ERROR    db 'unsuccessful free',10,13,'$'
STR_MCB_SIZE      db 'size:          bytes',10,13,'$'
STR_OWNER         db 'owner: $'
STR_LAST_BYTES    db 'last bytes: $'
STR_FREE          db 'free$'
STR_OSXMSUBM      db 'OS XMS UMB$'
STR_TOP_MEM       db "driver's top memory$"
STR_MSDOS         db 'MS DOS$'
STR_TAKEN386      db "386MAX UMB's block$"
STR_BLOCKED386    db 'blocked by 386MAX$'
STR_OWNED386      db '386MAX UMB$'


PRINT_NEW_LINE  PROC near
     push DX
     push AX

     mov DX, offset NEW_LINE
     mov AH, 09h
     int 21h

     pop AX
     pop DX
     ret
PRINT_NEW_LINE  ENDP


PRINT_BYTE      PROC near
; prints AL as two hex digits
```

```
        push BX
        push DX


        call BYTE_TO_HEX
        mov BH, AH


        mov DL, AL
        mov AH, 02h
        int 21h


        mov DL, BH
        mov AH, 02h
        int 21h


        pop DX
        pop BX
        ret
PRINT_BYTE    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
; AL --> two hex symbols in AX
        push     CX
        mov      AH,AL
        call     TETR_TO_HEX
        xchg     AL,AH
        mov      CL,4
        shr      AL,CL
        call     TETR_TO_HEX ; AL - high digit
        pop      CX          ; AH - low digit
        ret
```

```
            BYTE_TO_HEX  ENDP


            BYTE_TO_DEC    PROC   near
            ; convert AL to dec, SI - adress of low digit
                    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


            WORD_TO_DEC PROC near
            ; convert AX to dec, SI - adress of low digit
                    push     CX
                    push     DX
                    mov      CX,10
            loop_bd2:  div      CX
                    or       DL,30h
                    mov      [SI],DL
                    dec      SI
                    xor      DX,DX
                    cmp      AX,0
                    jnz      loop_bd2
            end_l2:    pop      DX
                    pop      CX
                    ret
            WORD_TO_DEC ENDP
```

```
; CODE
BEGIN:


PRINT_AVAILABLE_MEMORY:
    mov AH, 4Ah
    mov BX, 0FFFFh
    int 21h
    mov AX, BX
    mov BX, 10h
    mul BX


    mov SI, offset STR_MEM_AVAILABLE
    add SI, 22
    call WORD_TO_DEC
    mov DX, offset STR_MEM_AVAILABLE
    mov AH, 09h
    int 21h


PRINT_EXTENDED_MEMORY:
    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


    mov SI, offset STR_MEM_EXTENDED
    add SI, 21
    xor DX, DX
    call WORD_TO_DEC
    mov DX, offset STR_MEM_EXTENDED
    mov AH, 09h
    int 21h


    call PRINT_NEW_LINE
```

```
FREE_MEMORY:
      mov BX, offset END_OF_PROGRAM
      add BX, 10h
      shr BX, 1
      shr BX, 1
      shr BX, 1
      shr BX, 1


      mov AH, 4Ah
      int 21h


      jnc FREE_SUCCESS
      jmp FREE_ERROR


      FREE_SUCCESS:
            mov DX, offset STR_FREE_SUCCESS
            jmp FREE_MEMORY_END
      FREE_ERROR:
            mov DX, offset STR_FREE_ERROR


      FREE_MEMORY_END:
      mov AH, 09h
      int 21h


PRINT_MCBS:
      ; get first mcb's address
      mov AH, 52h
      int 21h
      mov AX, ES:[BX-2]
      mov ES, AX
      mov CX, 0


      NEXT_MCB:
            call PRINT_NEW_LINE
            inc CX
            mov AL, CL
            call PRINT_BYTE
            call PRINT_NEW_LINE


            mov DX, offset STR_OWNER
```

```asm
        mov AH, 09h
        int 21h


        ; get owner
        mov BX, ES:[1h]


        ; match owner
        mov DX, offset STR_FREE
        cmp BX, 0000h
        je MCB_MATCHED
        mov DX, offset STR_OSXMSUBM
        cmp BX, 0006h
        je MCB_MATCHED
        mov DX, offset STR_TOP_MEM
        cmp BX, 0007h
        je MCB_MATCHED
        mov DX, offset STR_MSDOS
        cmp BX, 0008h
        je MCB_MATCHED
        mov DX, offset STR_TAKEN386
        cmp BX, 0FFFAh
        je MCB_MATCHED
        mov DX, offset STR_BLOCKED386
        cmp BX, 0FFFDh
        je MCB_MATCHED
        mov DX, offset STR_OWNED386
        cmp BX, 0FFFEh
        je MCB_MATCHED


        jmp MCB_NOT_MATCHED


        ; print owner
MCB_MATCHED:
        mov AH, 09h
        int 21h
        jmp MCB_MATCH_END
MCB_NOT_MATCHED:
        mov AL, BH
        call PRINT_BYTE
        mov AL, BL
```

```
        call PRINT_BYTE

MCB_MATCH_END:
call PRINT_NEW_LINE

; get size
mov AX, ES:[3h]
mov BX, 10h
mul BX

; print size
mov SI, offset STR_MCB_SIZE
add SI, 11
call WORD_TO_DEC
mov DX, offset STR_MCB_SIZE
mov AH, 09h
int 21h

; print last 8 bytes
mov DX, offset STR_LAST_BYTES
mov AH, 09h
int 21h

push CX
mov CX, 8
mov BX, 0
mov AH, 02h
PRINT_LAST_BYTES:
        mov DL, ES:[BX+8h]
        int 21h
        inc BX
        loop PRINT_LAST_BYTES
        call PRINT_NEW_LINE
pop CX

; check if last block
mov AL, ES:[0h]
cmp AL, 5Ah
je PRINT_MCBS_END
```

```asm
                ; get next block's address
                mov AX, ES:[3h]
                mov BX, ES
                add BX, AX
                inc BX
                mov ES, BX


                jmp NEXT_MCB


        PRINT_MCBS_END:




; return to DOS
                xor     AL,AL
                mov     AH,4Ch
                int     21H


    END_OF_PROGRAM:
TESTPC      ENDS
                END     START       ; module end START - entry point
```

СОДЕРЖИМОЕ ФАЙЛА LAB3_3.ASM

```
TESTPC      SEGMENT
        ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG     100H


START:      JMP     BEGIN
; DATA
NEW_LINE             DB 10,13,'$'
STR_MEM_AVAILABLE    DB 'AVAILABLE MEMORY:        BYTES',10,13,'$'
STR_MEM_EXTENDED     DB 'EXTENDED MEMORY:        BYTES',10,13,'$'
STR_FREE_SUCCESS     DB 'SUCCESSFUL FREE',10,13,'$'
STR_FREE_ERROR       DB 'UNSUCCESSFUL FREE',10,13,'$'
STR_ALLOCATE_SUCCESS DB 'SUCCESSFUL ALLOCATION',10,13,'$'
STR_ALLOCATE_ERROR   DB 'UNSUCCESSFUL ALLOCATION',10,13,'$'
STR_MCB_SIZE         DB 'SIZE:        BYTES',10,13,'$'
STR_OWNER            DB 'OWNER: $'
STR_LAST_BYTES       DB 'LAST BYTES: $'
STR_FREE             DB 'FREE$'
STR_OSXMSUBM         DB 'OS XMS UMB$'
STR_TOP_MEM          DB "DRIVER'S TOP MEMORY$"
STR_MSDOS            DB 'MS DOS$'
STR_TAKEN386         DB "386MAX UMB'S BLOCK$"
STR_BLOCKED386       DB 'BLOCKED BY 386MAX$'
STR_OWNED386         DB '386MAX UMB$'


PRINT_NEW_LINE  PROC NEAR
     PUSH DX
     PUSH AX


     MOV DX, OFFSET NEW_LINE
     MOV AH, 09H
     INT 21H


     POP AX
     POP DX
     RET
PRINT_NEW_LINE  ENDP
```

```
PRINT_BYTE        PROC NEAR
; PRINTS AL AS TWO HEX DIGITS
      PUSH BX
      PUSH DX

      CALL BYTE_TO_HEX
      MOV BH, AH

      MOV DL, AL
      MOV AH, 02H
      INT 21H

      MOV DL, BH
      MOV AH, 02H
      INT 21H

      POP DX
      POP BX
      RET
PRINT_BYTE     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
; AL --> TWO HEX SYMBOLS IN AX
      PUSH      CX
      MOV       AH,AL
      CALL      TETR_TO_HEX
      XCHG      AL,AH
      MOV       CL,4
      SHR       AL,CL
      CALL      TETR_TO_HEX ; AL - HIGH DIGIT
```

23

```asm
        POP     CX              ; AH - LOW DIGIT
        RET
BYTE_TO_HEX  ENDP


BYTE_TO_DEC    PROC  NEAR
; CONVERT AL TO DEC, SI - ADRESS OF LOW DIGIT
            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


WORD_TO_DEC PROC NEAR
; CONVERT AX TO DEC, SI - ADRESS OF LOW DIGIT
            PUSH    AX
            PUSH    BX
            PUSH    CX
            PUSH    DX
            XOR             CX,CX
            MOV     BX,10
LOOP_BD2:
            DIV     BX
            OR      DL,30H
            MOV     [SI],DL
            DEC     SI
            XOR     DX,DX
```

```
                CMP       AX,0H
                JNZ       LOOP_BD2
END_L2:     POP       DX
                POP       CX
                    POP       BX
                    POP       AX
                RET


WORD_TO_DEC ENDP


; CODE
BEGIN:


PRINT_AVAILABLE_MEMORY:
        MOV AH, 4AH
        MOV BX, 0FFFFH
        INT 21H
        MOV AX, BX
        MOV BX, 10H
        MUL BX


        MOV SI, OFFSET STR_MEM_AVAILABLE
        ADD SI, 22
        CALL WORD_TO_DEC
        MOV DX, OFFSET STR_MEM_AVAILABLE
        MOV AH, 09H
        INT 21H


PRINT_EXTENDED_MEMORY:
        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


        MOV SI, OFFSET STR_MEM_EXTENDED
```

```asm
        ADD SI, 21
        XOR DX, DX
        CALL WORD_TO_DEC
        MOV DX, OFFSET STR_MEM_EXTENDED
        MOV AH, 09H
        INT 21H


        CALL PRINT_NEW_LINE


FREE_MEMORY:
        MOV BX, OFFSET END_OF_PROGRAM
        ADD BX, 10H
        SHR BX, 1
        SHR BX, 1
        SHR BX, 1
        SHR BX, 1


        MOV AH, 4AH
        INT 21H


        JNC FREE_SUCCESS
        JMP FREE_ERROR


        FREE_SUCCESS:
                MOV DX, OFFSET STR_FREE_SUCCESS
                JMP FREE_MEMORY_END
        FREE_ERROR:
                MOV DX, OFFSET STR_FREE_ERROR


        FREE_MEMORY_END:
        MOV AH, 09H
        INT 21H


ALLOCATE_MEMORY:
        MOV BX, 1000H
        MOV AH, 48H
        INT 21H


        JNC ALLOCATE_SUCCESS
        JMP ALLOCATE_ERROR
```

```
ALLOCATE_SUCCESS:
        MOV DX, OFFSET STR_ALLOCATE_SUCCESS
        JMP ALLOCATE_MEMORY_END
ALLOCATE_ERROR:
        MOV DX, OFFSET STR_ALLOCATE_ERROR


ALLOCATE_MEMORY_END:
MOV AH, 09H
INT 21H


PRINT_MCBS:
        ; GET FIRST MCB'S ADDRESS
        MOV AH, 52H
        INT 21H
        MOV AX, ES:[BX-2]
        MOV ES, AX
        MOV CX, 0


        NEXT_MCB:
                CALL PRINT_NEW_LINE
                INC CX
                MOV AL, CL
                CALL PRINT_BYTE
                CALL PRINT_NEW_LINE


                MOV DX, OFFSET STR_OWNER
                MOV AH, 09H
                INT 21H


                ; GET OWNER
                MOV BX, ES:[1H]


                ; MATCH OWNER
                MOV DX, OFFSET STR_FREE
                CMP BX, 0000H
                JE MCB_MATCHED
                MOV DX, OFFSET STR_OSXMSUBM
                CMP BX, 0006H
                JE MCB_MATCHED
```

27

```asm
        MOV DX, OFFSET STR_TOP_MEM
        CMP BX, 0007H
        JE MCB_MATCHED
        MOV DX, OFFSET STR_MSDOS
        CMP BX, 0008H
        JE MCB_MATCHED
        MOV DX, OFFSET STR_TAKEN386
        CMP BX, 0FFFAH
        JE MCB_MATCHED
        MOV DX, OFFSET STR_BLOCKED386
        CMP BX, 0FFFDH
        JE MCB_MATCHED
        MOV DX, OFFSET STR_OWNED386
        CMP BX, 0FFFEH
        JE MCB_MATCHED


        JMP MCB_NOT_MATCHED


; PRINT OWNER
MCB_MATCHED:
        MOV AH, 09H
        INT 21H
        JMP MCB_MATCH_END
MCB_NOT_MATCHED:
        MOV AL, BH
        CALL PRINT_BYTE
        MOV AL, BL
        CALL PRINT_BYTE


MCB_MATCH_END:
CALL PRINT_NEW_LINE


; GET SIZE
MOV AX, ES:[3H]
MOV BX, 10H
MUL BX


; PRINT SIZE
MOV SI, OFFSET STR_MCB_SIZE
ADD SI, 10
```

```asm
        CALL WORD_TO_DEC
        MOV DX, OFFSET STR_MCB_SIZE
        MOV AH, 09H
        INT 21H


        ; PRINT LAST 8 BYTES
        MOV DX, OFFSET STR_LAST_BYTES
        MOV AH, 09H
        INT 21H


        PUSH CX
        MOV CX, 8
        MOV BX, 0
        MOV AH, 02H
        PRINT_LAST_BYTES:
                MOV DL, ES:[BX+8H]
                INT 21H
                INC BX
                LOOP PRINT_LAST_BYTES
                CALL PRINT_NEW_LINE
        POP CX


        ; CHECK IF LAST BLOCK
        MOV AL, ES:[0H]
        CMP AL, 5AH
        JE PRINT_MCBS_END


        ; GET NEXT BLOCK'S ADDRESS
        MOV AX, ES:[3H]
        MOV BX, ES
        ADD BX, AX
        INC BX
        MOV ES, BX


        JMP NEXT_MCB


PRINT_MCBS_END:
```

```
; RETURN TO DOS
          XOR      AL,AL
          MOV      AH,4CH
          INT      21H


    END_OF_PROGRAM:
TESTPC    ENDS
          END      START      ; MODULE END START - ENTRY POINT
```

ПРИЛОЖЕНИЕ Г

СОДЕРЖИМОЕ ФАЙЛА LAB3_4.ASM

```asm
TESTPC      SEGMENT
        ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG     100H


START:      JMP     BEGIN
; DATA
NEW_LINE            DB 10,13,'$'
STR_MEM_AVAILABLE DB 'AVAILABLE MEMORY:        BYTES',10,13,'$'
STR_MEM_EXTENDED  DB 'EXTENDED MEMORY:        BYTES',10,13,'$'
STR_FREE_SUCCESS  DB 'SUCCESSFUL FREE',10,13,'$'
STR_FREE_ERROR    DB 'UNSUCCESSFUL FREE',10,13,'$'
STR_ALLOCATE_SUCCESS  DB 'SUCCESSFUL ALLOCATION',10,13,'$'
STR_ALLOCATE_ERROR    DB 'UNSUCCESSFUL ALLOCATION',10,13,'$'
STR_MCB_SIZE      DB 'SIZE:            BYTES',10,13,'$'
STR_OWNER         DB 'OWNER: $'
STR_LAST_BYTES    DB 'LAST BYTES: $'
STR_FREE          DB 'FREE$'
STR_OSXMSUBM      DB 'OS XMS UMB$'
STR_TOP_MEM       DB "DRIVER'S TOP MEMORY$"
STR_MSDOS         DB 'MS DOS$'
STR_TAKEN386      DB "386MAX UMB'S BLOCK$"
STR_BLOCKED386    DB 'BLOCKED BY 386MAX$'
STR_OWNED386      DB '386MAX UMB$'


PRINT_NEW_LINE  PROC NEAR
     PUSH DX
     PUSH AX


     MOV DX, OFFSET NEW_LINE
     MOV AH, 09H
     INT 21H


     POP AX
     POP DX
     RET
PRINT_NEW_LINE  ENDP
```

```asm
PRINT_BYTE          PROC NEAR
; PRINTS AL AS TWO HEX DIGITS
        PUSH BX
        PUSH DX

        CALL BYTE_TO_HEX
        MOV BH, AH

        MOV DL, AL
        MOV AH, 02H
        INT 21H

        MOV DL, BH
        MOV AH, 02H
        INT 21H

        POP DX
        POP BX
        RET
PRINT_BYTE      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
; AL --> TWO HEX SYMBOLS IN AX
        PUSH       CX
        MOV        AH,AL
        CALL       TETR_TO_HEX
        XCHG       AL,AH
        MOV        CL,4
        SHR        AL,CL
        CALL       TETR_TO_HEX ; AL - HIGH DIGIT
```

32

```
        POP       CX            ; AH - LOW DIGIT
        RET
BYTE_TO_HEX   ENDP


BYTE_TO_DEC    PROC   NEAR
; CONVERT AL TO DEC, SI - ADRESS OF LOW DIGIT
              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


WORD_TO_DEC PROC NEAR
; CONVERT AX TO DEC, SI - ADRESS OF LOW DIGIT
              PUSH       CX
              PUSH       DX
              MOV        CX,10
LOOP_BD2:     DIV        CX
              OR         DL,30H
              MOV        [SI],DL
              DEC        SI
              XOR        DX,DX
              CMP        AX,0
              JNZ        LOOP_BD2
END_L2:       POP        DX
              POP        CX
```

```
              RET
WORD_TO_DEC ENDP


; CODE
BEGIN:


PRINT_AVAILABLE_MEMORY:
        MOV AH, 4AH
        MOV BX, 0FFFFH
        INT 21H
        MOV AX, BX
        MOV BX, 10H
        MUL BX


        MOV SI, OFFSET STR_MEM_AVAILABLE
        ADD SI, 22
        CALL WORD_TO_DEC
        MOV DX, OFFSET STR_MEM_AVAILABLE
        MOV AH, 09H
        INT 21H


PRINT_EXTENDED_MEMORY:
        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


        MOV SI, OFFSET STR_MEM_EXTENDED
        ADD SI, 21
        XOR DX, DX
        CALL WORD_TO_DEC
        MOV DX, OFFSET STR_MEM_EXTENDED
        MOV AH, 09H
        INT 21H
```

```
            CALL PRINT_NEW_LINE


ALLOCATE_MEMORY:
        MOV BX, 1000H
        MOV AH, 48H
        INT 21H


        JNC ALLOCATE_SUCCESS
        JMP ALLOCATE_ERROR


        ALLOCATE_SUCCESS:
                MOV DX, OFFSET STR_ALLOCATE_SUCCESS
                JMP ALLOCATE_MEMORY_END
        ALLOCATE_ERROR:
                MOV DX, OFFSET STR_ALLOCATE_ERROR


        ALLOCATE_MEMORY_END:
        MOV AH, 09H
        INT 21H


FREE_MEMORY:
        MOV BX, OFFSET END_OF_PROGRAM
        ADD BX, 10H
        SHR BX, 1
        SHR BX, 1
        SHR BX, 1
        SHR BX, 1


        MOV AH, 4AH
        INT 21H


        JNC FREE_SUCCESS
        JMP FREE_ERROR


        FREE_SUCCESS:
                MOV DX, OFFSET STR_FREE_SUCCESS
                JMP FREE_MEMORY_END
        FREE_ERROR:
                MOV DX, OFFSET STR_FREE_ERROR
```

```
        FREE_MEMORY_END:
        MOV AH, 09H
        INT 21H


PRINT_MCBS:
        ; GET FIRST MCB'S ADDRESS
        MOV AH, 52H
        INT 21H
        MOV AX, ES:[BX-2]
        MOV ES, AX
        MOV CX, 0


        NEXT_MCB:
            CALL PRINT_NEW_LINE
            INC CX
            MOV AL, CL
            CALL PRINT_BYTE
            CALL PRINT_NEW_LINE


            MOV DX, OFFSET STR_OWNER
            MOV AH, 09H
            INT 21H


            ; GET OWNER
            MOV BX, ES:[1H]


            ; MATCH OWNER
            MOV DX, OFFSET STR_FREE
            CMP BX, 0000H
            JE MCB_MATCHED
            MOV DX, OFFSET STR_OSXMSUBM
            CMP BX, 0006H
            JE MCB_MATCHED
            MOV DX, OFFSET STR_TOP_MEM
            CMP BX, 0007H
            JE MCB_MATCHED
            MOV DX, OFFSET STR_MSDOS
            CMP BX, 0008H
            JE MCB_MATCHED
            MOV DX, OFFSET STR_TAKEN386
```

```asm
        CMP BX, 0FFFAH
        JE MCB_MATCHED
        MOV DX, OFFSET STR_BLOCKED386
        CMP BX, 0FFFDH
        JE MCB_MATCHED
        MOV DX, OFFSET STR_OWNED386
        CMP BX, 0FFFEH
        JE MCB_MATCHED

        JMP MCB_NOT_MATCHED


; PRINT OWNER
MCB_MATCHED:
        MOV AH, 09H
        INT 21H
        JMP MCB_MATCH_END
MCB_NOT_MATCHED:
        MOV AL, BH
        CALL PRINT_BYTE
        MOV AL, BL
        CALL PRINT_BYTE


MCB_MATCH_END:
CALL PRINT_NEW_LINE

; GET SIZE
MOV AX, ES:[3H]
MOV BX, 10H
MUL BX

; PRINT SIZE
MOV SI, OFFSET STR_MCB_SIZE
ADD SI, 11
CALL WORD_TO_DEC
MOV DX, OFFSET STR_MCB_SIZE
MOV AH, 09H
INT 21H


; PRINT LAST 8 BYTES
MOV DX, OFFSET STR_LAST_BYTES
```

```
                MOV AH, 09H
                INT 21H


                PUSH CX
                MOV CX, 8
                MOV BX, 0
                MOV AH, 02H
                PRINT_LAST_BYTES:
                        MOV DL, ES:[BX+8H]
                        INT 21H
                        INC BX
                        LOOP PRINT_LAST_BYTES
                        CALL PRINT_NEW_LINE
                POP CX


                ; CHECK IF LAST BLOCK
                MOV AL, ES:[0H]
                CMP AL, 5AH
                JE PRINT_MCBS_END


                ; GET NEXT BLOCK'S ADDRESS
                MOV AX, ES:[3H]
                MOV BX, ES
                ADD BX, AX
                INC BX
                MOV ES, BX


                JMP NEXT_MCB


        PRINT_MCBS_END:




; RETURN TO DOS
                XOR     AL,AL
                MOV     AH,4CH
                INT     21H


    END_OF_PROGRAM:
TESTPC      ENDS
```

```
        END     START      ; MODULE END START - ENTRY POINT
```