**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. 8382 _____ Нечепуренко Н.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

**Цель работы.**

Исследование организации управления основной памятью в операционной системе DOS, функций ядра управления памятью и структур данных.

**Выполнение работы.**

Для исследования основной памяти был написан .com модуль, в результате работы которого на экран выводится следующее:

1. Количество доступной памяти.
2. Размер расширенной памяти.
3. Цепочку блоков управления памятью.

Результат работы модуля приведен на рисунке 1, полный исходный код приведен в Приложении А.

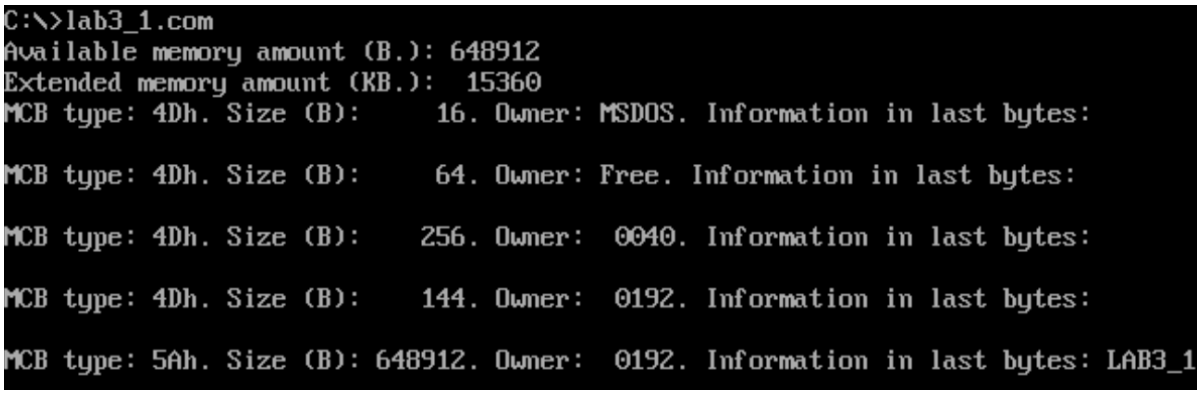


Рисунок 1 – Результат работы программы lab3_1.com

Для каждого блока выводится его размер, владелец и последние восемь байтов этого блока.

Изменим программу так, чтобы она освобождала память, которую не занимает. Результат работы программы приведен на рисунке 2, исходный код в Приложении Б.

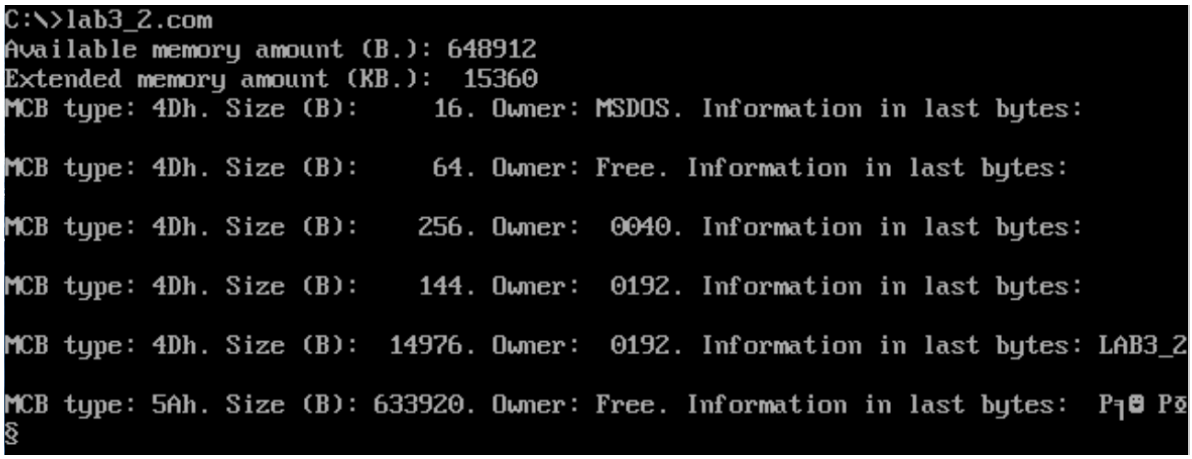


Рисунок 2 – Результат работы программы lab3_2.com

Из рисунка 2 можно сделать вывод, что программа занимает в памяти ~15 Кбайт, остальная же память была освобождена, о чем свидетельствует изменение в строке владельца для последнего блока.

Немного изменим код программы и запросим 64 Кбайта у операционной системы после освобождения памяти. Результат работы программы приведен на рисунке 3, исходный код в Приложении В.

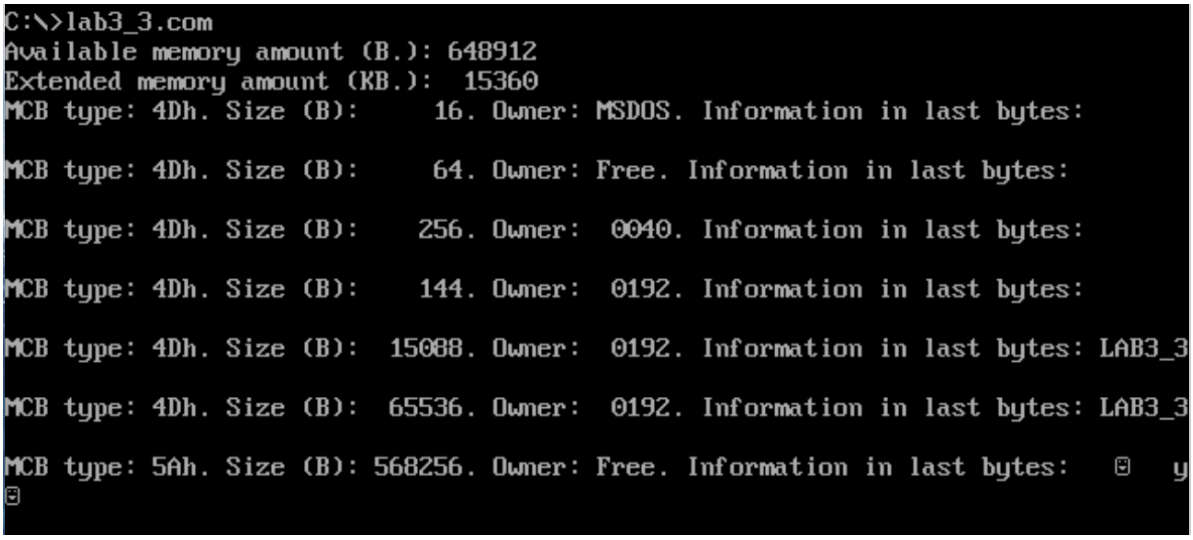


Рисунок 3 – Результат работы программы lab3_3.com

На рисунке 3 можно увидеть запрошенный программой блок памяти. Операционная система выделила блок на полтора Кбайта больше, чем запрашивала программа.

Изменим программу, поместив код выделения памяти до кода освобождения. Результат работы программы приведен на рисунке 4, исходный код в Приложении Г.

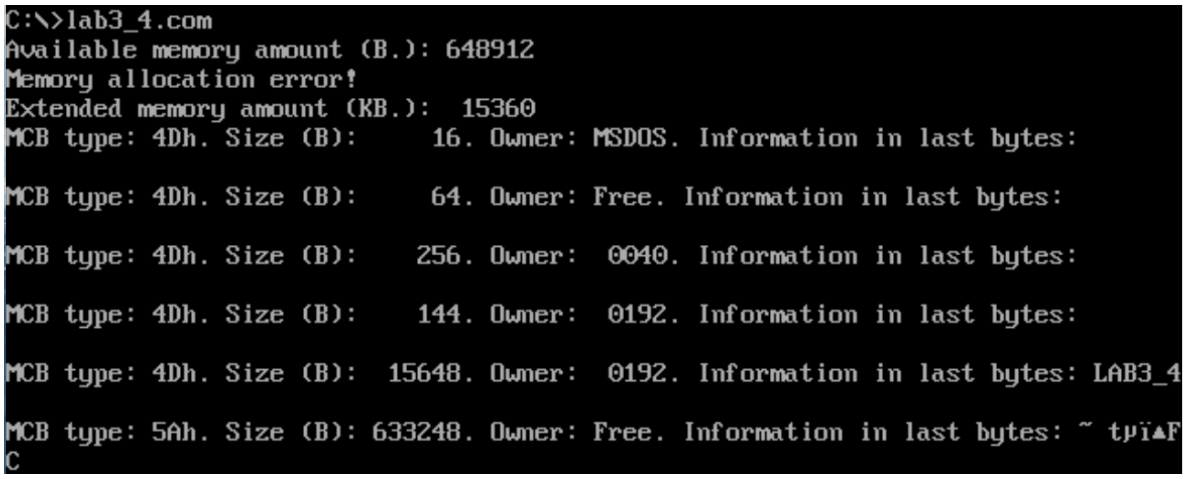


Рисунок 4 – Результат работы программы lab3_4.com

Программа сообщила об ошибке выделения памяти. Это неудивительно, ведь во время запроса 64 Кбайт еще не была освобождена выделенная, но неиспользуемая программой память, а как видно из предыдущих рисунков, другого свободного блока нужного размера не имеется.

**Контрольные вопросы.**

1. Что означает «доступный объём памяти»?

Максимальный объём оперативной памяти, который операционная система может выделить программе.

2. Где MCB блок вашей программы в списке?

MCB блоки программы имеют в строке owner сегментный адрес PSP 192h. Соответствие программ и номеров блоков для удобства приведено в таблице 1.

3. Какой размер памяти занимает программа в каждом случае?

Размеры программ приведены в таблице 1 (см. ниже).

*Таблица 1 – Соответствие программ*

| **Программа** | **Номера МСВ блоков*** | **Размер, Байт** |
|---|---|---|
| lab3_1.com | 4, 5 | 144 + 648912 = 649056 |
| lab3_2.com | 4, 5 | 144 + 14976 = 15120 |
| lab3_3.com | 4, 5, 6 | 144 + 15088 + 65536 = 80768 |
| lab3_4.com | 4, 5 | 144 + 15648 = 15792 |

*Номера МСВ блоков на соответствующих рисунках.

**Выводы.**

В результате выполнения работы были получены навыки работы с функциями ядра управления памятью операционной системы DOS, была проанализирована структура МСВ блоков.

# ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД LAB3_1.COM

```
codeseg segment
    assume cs:codeseg, ds:codeseg, es:nothing, ss:nothing
    org 100h
    start: jmp begin

endline db 13, 10, "$"
available_memory db "Available memory amount (B.): ", "$"
available_memory_number db "      ", "$"
extended_memory db "Extended memory amount (KB.): ", "$"
extended_memory_number db "      ", "$"
mcb_header db "MCB type: ", "$"
mcb_size db "h. Size (B):       ", "$"
mcb_owner db ". Owner: ", "$"
mcb_info db ". Information in last bytes: ", "$"

mcb_owner_free db "Free", "$"
mcb_owner_os db "OS XMS UMB", "$"
mcb_owner_driver db "Upper driver memory", "$"
mcb_owner_msdos db "MSDOS", "$"
mcb_owner_max1 db "Control 386MAX UMB block", "$"
mcb_owner_max2 db "Blocked by 386MAX", "$"
mcb_owner_max3 db "386MAX UMB", "$"
mcb_owner_address db "     ", "$"

begin:
available_memory_label:
    mov di, offset available_memory
    call print
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov bx, 10h
    mul bx
    mov si, offset available_memory_number
    add si, 5
    call WRD_TO_DEC
    mov di, offset available_memory_number
    call print
    mov di, offset endline
    call print
```

```
extended_memory_label:
    mov di, offset extended_memory
    call print
    xor ax, ax
    xor dx, dx
    mov al, 30h ; L
    out 70h, al ;send al to index cmos port
    in al, 71h ;get response
    mov bl, al

    mov al, 31h ; H
    out 70h, al
    in al, 71h
    mov bh, al
    mov ax, bx
    mov si, offset extended_memory_number
    add si, 5
    call WRD_TO_DEC
    mov di, offset extended_memory_number
    call print
    mov di, offset endline
    call print

mcb_label:
    xor ax, ax
    mov ah, 52h
    int 21h
    mov cx, es:[bx-2]
    mov es, cx

mcb_loop:
    ; type
    mov di, offset mcb_header
    call print
    mov al, es:[0]
    call putch

    ; size
    mov ax, es:[3]
    mov bx, 10h
    mul bx
```

```
    mov si, offset mcb_size
    add si, 18
    call WRD_TO_DEC
    mov di, offset mcb_size
    call print

    ; owner
    mov di, offset mcb_owner
    call print
    mov ax, es:[1]
    call show_owner


    ; last eight bytes
    mov di, offset mcb_info
    call print

    mov bx, 0
mcb_info_loop:
    mov dl, es:[bx+8]
    mov ah, 2h
    int 21h
    inc bx
    cmp bx, 8
    jl mcb_info_loop

    mov di, offset endline
    call print
    ; if it is the last mcb
    mov al, es:[0]
    cmp al, 5ah
    je final

    ; not last :)
    mov cx, es:[3]
    mov bx, es
    add bx, cx
    inc bx
    mov es, bx
    jmp mcb_loop
```

```
final:
    mov ax, 4c00h
    int 21h

putch proc near
    ; print char from al
    push ax
    push dx
    call BYTE_TO_HEX
    xchg ax, dx
    mov ah, 2h
    int 21h
    xchg dl, dh
    int 21h
    pop dx
    pop ax
    ret
putch endp

print proc near
    ; prints di content
    push dx
    push ax
    mov ah, 9h
    mov dx, di
    int 21h
    pop ax
    pop dx
    ret
print endp

show_owner proc near
    push di
    push bx
    push ax
    cmp ax, 0
    jne show_owner_else_1
    mov di, offset mcb_owner_free
    jmp show_owner_ret
show_owner_else_1:
    cmp ax, 6
    jne show_owner_else_2
```

```
    mov di, offset mcb_owner_os
    jmp show_owner_ret
show_owner_else_2:
    cmp ax, 7
    jne show_owner_else_3
    mov di, offset mcb_owner_driver
    jmp show_owner_ret
show_owner_else_3:
    cmp ax, 8
    jne show_owner_else_4
    mov di, offset mcb_owner_msdos
    jmp show_owner_ret
show_owner_else_4:
    cmp ax, 0fffah
    jne show_owner_else_5
    mov di, offset mcb_owner_max1
    jmp show_owner_ret
show_owner_else_5:
    cmp ax, 0fffdh
    jne show_owner_else_6
    mov di, offset mcb_owner_max2
    jmp show_owner_ret
show_owner_else_6:
    cmp ax, 0fffeh
    jne show_owner_else_7
    mov di, offset mcb_owner_max3
    jmp show_owner_ret
show_owner_else_7:
    mov di, offset mcb_owner_address
    add di, 4
    call WRD_TO_HEX
    mov di, offset mcb_owner_address
show_owner_ret:
    call print
    pop ax
    pop bx
    pop di
    ret
show_owner 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
;байт в AL переводится в два символа шест. 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

WRD_TO_DEC PROC NEAR
           push  cx
           push  dx
           mov   cx,10
loop_b: div      cx
           or          dl,30h
           mov   [si],dl
           dec   si
           xor   dx,dx
           cmp   ax,10
           jae   loop_b
           cmp   al,00h
           je          endl
           or          al,30h
           mov   [si],al
endl: pop   dx
           pop   cx
           ret
WRD_TO_DEC 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

codeseg ends
end start
```

# ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД LAB3_2.COM

```
codeseg segment
    assume cs:codeseg, ds:codeseg, es:nothing, ss:nothing
    org 100h
    start: jmp begin

endline db 13, 10, "$"
available_memory db "Available memory amount (B.): ", "$"
available_memory_number db "      ", "$"
extended_memory db "Extended memory amount (KB.): ", "$"
extended_memory_number db "      ", "$"
mcb_header db "MCB type: ", "$"
mcb_size db "h. Size (B):       ", "$"
mcb_owner db ". Owner: ", "$"
mcb_info db ". Information in last bytes: ", "$"

mcb_owner_free db "Free", "$"
mcb_owner_os db "OS XMS UMB", "$"
mcb_owner_driver db "Upper driver memory", "$"
mcb_owner_msdos db "MSDOS", "$"
mcb_owner_max1 db "Control 386MAX UMB block", "$"
mcb_owner_max2 db "Blocked by 386MAX", "$"
mcb_owner_max3 db "386MAX UMB", "$"
mcb_owner_address db "     ", "$"

begin:
available_memory_label:
    mov di, offset available_memory
    call print
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov bx, 10h
    mul bx
    mov si, offset available_memory_number
    add si, 5
    call WRD_TO_DEC
    mov di, offset available_memory_number
    call print
    mov di, offset endline
    call print
```

```
    ; reduce to size of program
    mov ah, 4ah
      mov bx, offset program_end
      int 21h

extended_memory_label:
    mov di, offset extended_memory
    call print
    xor ax, ax
    xor dx, dx
    mov al, 30h ; L
    out 70h, al ;send al to index cmos port
    in al, 71h ;get response
    mov bl, al

    mov al, 31h ; H
    out 70h, al
    in al, 71h
    mov bh, al
    mov ax, bx
    mov si, offset extended_memory_number
    add si, 5
    call WRD_TO_DEC
    mov di, offset extended_memory_number
    call print
    mov di, offset endline
    call print

mcb_label:
    xor ax, ax
    mov ah, 52h
    int 21h
    mov cx, es:[bx-2]
    mov es, cx

mcb_loop:
    ; type
    mov di, offset mcb_header
    call print
    mov al, es:[0]
    call putch
```

```
    ; size
    mov ax, es:[3]
    mov bx, 10h
    mul bx
    mov si, offset mcb_size
    add si, 18
    call WRD_TO_DEC
    mov di, offset mcb_size
    call print

    ; owner
    mov di, offset mcb_owner
    call print
    mov ax, es:[1]
    call show_owner


    ; last eight bytes
    mov di, offset mcb_info
    call print

    mov bx, 0
mcb_info_loop:
    mov dl, es:[bx+8]
    mov ah, 2h
    int 21h
    inc bx
    cmp bx, 8
    jl mcb_info_loop

    mov di, offset endline
    call print
    ; if it is the last mcb
    mov al, es:[0]
    cmp al, 5ah
    je final

    ; not last :)
    mov cx, es:[3]
    mov bx, es
    add bx, cx
```

```
    inc bx
    mov es, bx
    jmp mcb_loop


final:
    mov ax, 4c00h
    int 21h

putch proc near
    ; print char from al
    push ax
    push dx
    call BYTE_TO_HEX
    xchg ax, dx
    mov ah, 2h
    int 21h
    xchg dl, dh
    int 21h
    pop dx
    pop ax
    ret
putch endp

print proc near
    ; prints di content
    push dx
    push ax
    mov ah, 9h
    mov dx, di
    int 21h
    pop ax
    pop dx
    ret
print endp

show_owner proc near
    push di
    push bx
    push ax
    cmp ax, 0
    jne show_owner_else_1
```

```
    mov di, offset mcb_owner_free
    jmp show_owner_ret
show_owner_else_1:
    cmp ax, 6
    jne show_owner_else_2
    mov di, offset mcb_owner_os
    jmp show_owner_ret
show_owner_else_2:
    cmp ax, 7
    jne show_owner_else_3
    mov di, offset mcb_owner_driver
    jmp show_owner_ret
show_owner_else_3:
    cmp ax, 8
    jne show_owner_else_4
    mov di, offset mcb_owner_msdos
    jmp show_owner_ret
show_owner_else_4:
    cmp ax, 0fffah
    jne show_owner_else_5
    mov di, offset mcb_owner_max1
    jmp show_owner_ret
show_owner_else_5:
    cmp ax, 0fffdh
    jne show_owner_else_6
    mov di, offset mcb_owner_max2
    jmp show_owner_ret
show_owner_else_6:
    cmp ax, 0fffeh
    jne show_owner_else_7
    mov di, offset mcb_owner_max3
    jmp show_owner_ret
show_owner_else_7:
    mov di, offset mcb_owner_address
    add di, 4
    call WRD_TO_HEX
    mov di, offset mcb_owner_address
show_owner_ret:
    call print
    pop ax
    pop bx
    pop di
```

```
    ret
show_owner 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
;байт в AL переводится в два символа шест. 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

WRD_TO_DEC PROC NEAR
           push  cx
           push  dx
           mov   cx,10
loop_b: div      cx
           or          dl,30h
           mov   [si],dl
           dec   si
           xor   dx,dx
           cmp   ax,10
           jae   loop_b
           cmp   al,00h
           je          endl
           or          al,30h
           mov   [si],al
endl: pop   dx
```

```
            pop   cx
            ret
WRD_TO_DEC 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
program_end:
codeseg ends
end start
```

# ПРИЛОЖЕНИЕ В. ИСХОДНЫЙ КОД LAB3_3.COM

```
codeseg segment
    assume cs:codeseg, ds:codeseg, es:nothing, ss:nothing
    org 100h
    start: jmp begin

endline db 13, 10, "$"
available_memory db "Available memory amount (B.): ", "$"
available_memory_number db "      ", "$"
extended_memory db "Extended memory amount (KB.): ", "$"
extended_memory_number db "      ", "$"
mcb_header db "MCB type: ", "$"
mcb_size db "h. Size (B):       ", "$"
mcb_owner db ". Owner: ", "$"
mcb_info db ". Information in last bytes: ", "$"

mcb_owner_free db "Free", "$"
mcb_owner_os db "OS XMS UMB", "$"
mcb_owner_driver db "Upper driver memory", "$"
mcb_owner_msdos db "MSDOS", "$"
mcb_owner_max1 db "Control 386MAX UMB block", "$"
mcb_owner_max2 db "Blocked by 386MAX", "$"
mcb_owner_max3 db "386MAX UMB", "$"
mcb_owner_address db "     ", "$"

begin:
available_memory_label:
    mov di, offset available_memory
    call print
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov bx, 10h
    mul bx
    mov si, offset available_memory_number
    add si, 5
    call WRD_TO_DEC
    mov di, offset available_memory_number
    call print
    mov di, offset endline
    call print
```

```
    ; reduce to size of program
    mov ah, 4ah
      mov bx, offset program_end
      int 21h

    ; alloc new block
    mov ah, 48h
    mov bx, 1000h
    int 21h

extended_memory_label:
    mov di, offset extended_memory
    call print
    xor ax, ax
    xor dx, dx
    mov al, 30h ; L
    out 70h, al ;send al to index cmos port
    in al, 71h ;get response
    mov bl, al

    mov al, 31h ; H
    out 70h, al
    in al, 71h
    mov bh, al
    mov ax, bx
    mov si, offset extended_memory_number
    add si, 5
    call WRD_TO_DEC
    mov di, offset extended_memory_number
    call print
    mov di, offset endline
    call print

mcb_label:
    xor ax, ax
    mov ah, 52h
    int 21h
    mov cx, es:[bx-2]
    mov es, cx

mcb_loop:
```

```
    ; type
    mov di, offset mcb_header
    call print
    mov al, es:[0]
    call putch

    ; size
    mov ax, es:[3]
    mov bx, 10h
    mul bx
    mov si, offset mcb_size
    add si, 18
    call WRD_TO_DEC
    mov di, offset mcb_size
    call print

    ; owner
    mov di, offset mcb_owner
    call print
    mov ax, es:[1]
    call show_owner


    ; last eight bytes
    mov di, offset mcb_info
    call print

    mov bx, 0
mcb_info_loop:
    mov dl, es:[bx+8]
    mov ah, 2h
    int 21h
    inc bx
    cmp bx, 8
    jl mcb_info_loop

    mov di, offset endline
    call print
    ; if it is the last mcb
    mov al, es:[0]
    cmp al, 5ah
    je final
```

```
    ; not last :)
    mov cx, es:[3]
    mov bx, es
    add bx, cx
    inc bx
    mov es, bx
    jmp mcb_loop


final:
    mov ax, 4c00h
    int 21h

putch proc near
    ; print char from al
    push ax
    push dx
    call BYTE_TO_HEX
    xchg ax, dx
    mov ah, 2h
    int 21h
    xchg dl, dh
    int 21h
    pop dx
    pop ax
    ret
putch endp

print proc near
    ; prints di content
    push dx
    push ax
    mov ah, 9h
    mov dx, di
    int 21h
    pop ax
    pop dx
    ret
print endp

show_owner proc near
```

```
    push di
    push bx
    push ax
    cmp ax, 0
    jne show_owner_else_1
    mov di, offset mcb_owner_free
    jmp show_owner_ret
show_owner_else_1:
    cmp ax, 6
    jne show_owner_else_2
    mov di, offset mcb_owner_os
    jmp show_owner_ret
show_owner_else_2:
    cmp ax, 7
    jne show_owner_else_3
    mov di, offset mcb_owner_driver
    jmp show_owner_ret
show_owner_else_3:
    cmp ax, 8
    jne show_owner_else_4
    mov di, offset mcb_owner_msdos
    jmp show_owner_ret
show_owner_else_4:
    cmp ax, 0fffah
    jne show_owner_else_5
    mov di, offset mcb_owner_max1
    jmp show_owner_ret
show_owner_else_5:
    cmp ax, 0fffdh
    jne show_owner_else_6
    mov di, offset mcb_owner_max2
    jmp show_owner_ret
show_owner_else_6:
    cmp ax, 0fffeh
    jne show_owner_else_7
    mov di, offset mcb_owner_max3
    jmp show_owner_ret
show_owner_else_7:
    mov di, offset mcb_owner_address
    add di, 4
    call WRD_TO_HEX
    mov di, offset mcb_owner_address
```

```
show_owner_ret:
    call print
    pop ax
    pop bx
    pop di
    ret
show_owner 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
;байт в AL переводится в два символа шест. 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

WRD_TO_DEC PROC NEAR
           push  cx
           push  dx
           mov   cx,10
loop_b: div      cx
           or          dl,30h
           mov   [si],dl
           dec   si
           xor   dx,dx
           cmp   ax,10
           jae   loop_b
```

```
            cmp   al,00h
            je          endl
            or          al,30h
            mov   [si],al
endl: pop   dx
            pop   cx
            ret
WRD_TO_DEC 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
program_end:
codeseg ends
end start
```

# ПРИЛОЖЕНИЕ Г. ИСХОДНЫЙ КОД LAB3_4.COM

```
codeseg segment
    assume cs:codeseg, ds:codeseg, es:nothing, ss:nothing
    org 100h
    start: jmp begin

endline db 13, 10, "$"
available_memory db "Available memory amount (B.): ", "$"
available_memory_number db "      ", "$"
extended_memory db "Extended memory amount (KB.): ", "$"
extended_memory_number db "      ", "$"
mcb_header db "MCB type: ", "$"
mcb_size db "h. Size (B):       ", "$"
mcb_owner db ". Owner: ", "$"
mcb_info db ". Information in last bytes: ", "$"

mcb_owner_free db "Free", "$"
mcb_owner_os db "OS XMS UMB", "$"
mcb_owner_driver db "Upper driver memory", "$"
mcb_owner_msdos db "MSDOS", "$"
mcb_owner_max1 db "Control 386MAX UMB block", "$"
mcb_owner_max2 db "Blocked by 386MAX", "$"
mcb_owner_max3 db "386MAX UMB", "$"
mcb_owner_address db "     ", "$"

mem_alloc_error db "Memory allocation error!", 13, 10, "$"

begin:
available_memory_label:
    mov di, offset available_memory
    call print
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h
    mov ax, bx
    mov bx, 10h
    mul bx
    mov si, offset available_memory_number
    add si, 5
    call WRD_TO_DEC
    mov di, offset available_memory_number
    call print
```

```
    mov di, offset endline
    call print

    ; alloc new block
    mov ah, 48h
    mov bx, 1000h
    int 21h
    jnc reduce
    mov di, offset mem_alloc_error
    call print

reduce:
    ; reduce to size of program
    mov ah, 4ah
      mov bx, offset program_end
      int 21h


extended_memory_label:
    mov di, offset extended_memory
    call print
    xor ax, ax
    xor dx, dx
    mov al, 30h ; L
    out 70h, al ;send al to index cmos port
    in al, 71h ;get response
    mov bl, al

    mov al, 31h ; H
    out 70h, al
    in al, 71h
    mov bh, al
    mov ax, bx
    mov si, offset extended_memory_number
    add si, 5
    call WRD_TO_DEC
    mov di, offset extended_memory_number
    call print
    mov di, offset endline
    call print

mcb_label:
```

```
    xor ax, ax
    mov ah, 52h
    int 21h
    mov cx, es:[bx-2]
    mov es, cx

mcb_loop:
    ; type
    mov di, offset mcb_header
    call print
    mov al, es:[0]
    call putch

    ; size
    mov ax, es:[3]
    mov bx, 10h
    mul bx
    mov si, offset mcb_size
    add si, 18
    call WRD_TO_DEC
    mov di, offset mcb_size
    call print

    ; owner
    mov di, offset mcb_owner
    call print
    mov ax, es:[1]
    call show_owner


    ; last eight bytes
    mov di, offset mcb_info
    call print

    mov bx, 0
mcb_info_loop:
    mov dl, es:[bx+8]
    mov ah, 2h
    int 21h
    inc bx
    cmp bx, 8
    jl mcb_info_loop
```

```
    mov di, offset endline
    call print
    ; if it is the last mcb
    mov al, es:[0]
    cmp al, 5ah
    je final

    ; not last :)
    mov cx, es:[3]
    mov bx, es
    add bx, cx
    inc bx
    mov es, bx
    jmp mcb_loop


final:
    mov ax, 4c00h
    int 21h

putch proc near
    ; print char from al
    push ax
    push dx
    call BYTE_TO_HEX
    xchg ax, dx
    mov ah, 2h
    int 21h
    xchg dl, dh
    int 21h
    pop dx
    pop ax
    ret
putch endp

print proc near
    ; prints di content
    push dx
    push ax
    mov ah, 9h
    mov dx, di
```

```
    int 21h
    pop ax
    pop dx
    ret
print endp

show_owner proc near
    push di
    push bx
    push ax
    cmp ax, 0
    jne show_owner_else_1
    mov di, offset mcb_owner_free
    jmp show_owner_ret
show_owner_else_1:
    cmp ax, 6
    jne show_owner_else_2
    mov di, offset mcb_owner_os
    jmp show_owner_ret
show_owner_else_2:
    cmp ax, 7
    jne show_owner_else_3
    mov di, offset mcb_owner_driver
    jmp show_owner_ret
show_owner_else_3:
    cmp ax, 8
    jne show_owner_else_4
    mov di, offset mcb_owner_msdos
    jmp show_owner_ret
show_owner_else_4:
    cmp ax, 0fffah
    jne show_owner_else_5
    mov di, offset mcb_owner_max1
    jmp show_owner_ret
show_owner_else_5:
    cmp ax, 0fffdh
    jne show_owner_else_6
    mov di, offset mcb_owner_max2
    jmp show_owner_ret
show_owner_else_6:
    cmp ax, 0fffeh
    jne show_owner_else_7
```

```
    mov di, offset mcb_owner_max3
    jmp show_owner_ret
show_owner_else_7:
    mov di, offset mcb_owner_address
    add di, 4
    call WRD_TO_HEX
    mov di, offset mcb_owner_address
show_owner_ret:
    call print
    pop ax
    pop bx
    pop di
    ret
show_owner 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
;байт в AL переводится в два символа шест. 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

WRD_TO_DEC PROC NEAR
           push  cx
           push  dx
           mov   cx,10
```

```
loop_b: div     cx
            or          dl,30h
            mov   [si],dl
            dec   si
            xor   dx,dx
            cmp   ax,10
            jae   loop_b
            cmp   al,00h
            je          endl
            or          al,30h
            mov   [si],al
endl: pop   dx
            pop   cx
            ret
WRD_TO_DEC 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
program_end:
codeseg ends
end start
```