МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №3

по дисциплине «Операционные системы»

Тема: Исследование организации управления основной памятью

Студент гр.8382 _____ Фильцин И.В.

Преподаватель _____ Ефремов М.А..

Санкт-Петербург

2020

Цель работы

Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованный в компьютере и способ организации, принятый в ОС. В лабораторной работе рассматривается нестраничная память и способ управления динамическими разделами. Для реализации управления памяью в этом случае строится список занятых и свободных участков памяти. Функции ядра, обеспечивающие управление основной памятью, прссматривают и преобразуют этот список.

В лабораторной работе исследуются структуры данных и работа функций управления памятью ядра операционной системы.

Ход работы

В ходе лабораторной программы был написан программный модуль типа .COM (См исх.код в приложении А), который распечатывает следующую информацию:

1) Количество доступной памяти

2) Размер расширенной памяти

3) Цепочку блоков управления памятью

Результат работы см. на рис. 1.

После этого программа была модифицирована так, чтобы она освобождала память, которую не занимает (См. исх. код в приложении Б). Результат работы модифицированной программы приведён на рис. 2

Полученная программа была снова модифицирована так, чтобы после освобождения памяти, запрашивала 64KB памяти (См. исх. код в приложении В). Результат работы см. на рис. 3

После этого изначальная программы была модифицирована так, чтобы она запрашивала дополнительные 64KB памяти до освобождения памяти (См.

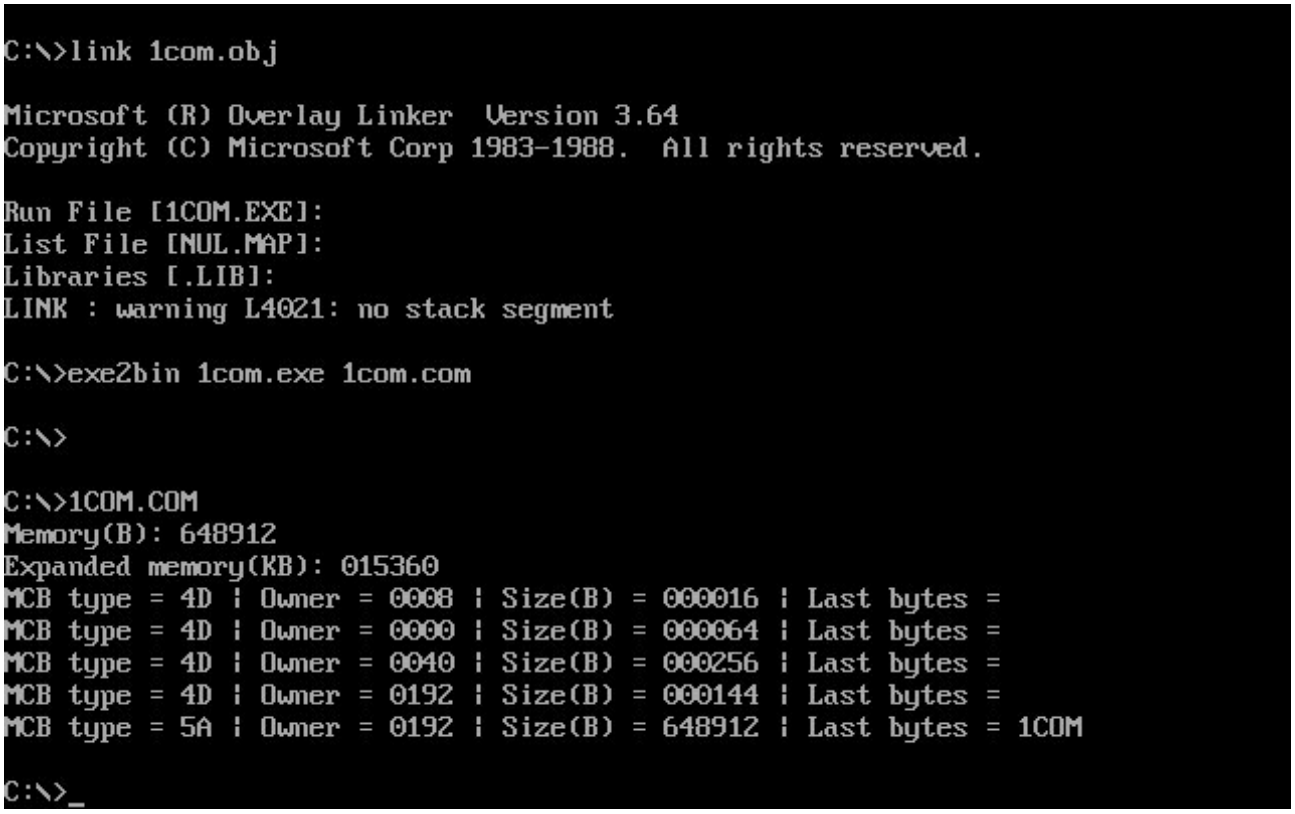
C:\>link 1com.obj
Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.
Run File [1COM.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
C:\>exe2bin 1com.exe 1com.com
C:\>
C:\>1COM.COM
Memory(B): 648912
Expanded memory(KB): 015360
MCB type = 4D ¦ Owner = 0008 ¦ Size(B) = 000016 ¦ Last bytes =
MCB type = 4D ¦ Owner = 0000 ¦ Size(B) = 000064 ¦ Last bytes =
MCB type = 4D ¦ Owner = 0040 ¦ Size(B) = 000256 ¦ Last bytes =
MCB type = 4D ¦ Owner = 0192 ¦ Size(B) = 000144 ¦ Last bytes =
MCB type = 5A ¦ Owner = 0192 ¦ Size(B) = 648912 ¦ Last bytes = 1COM
C:\>_

Рис. 1

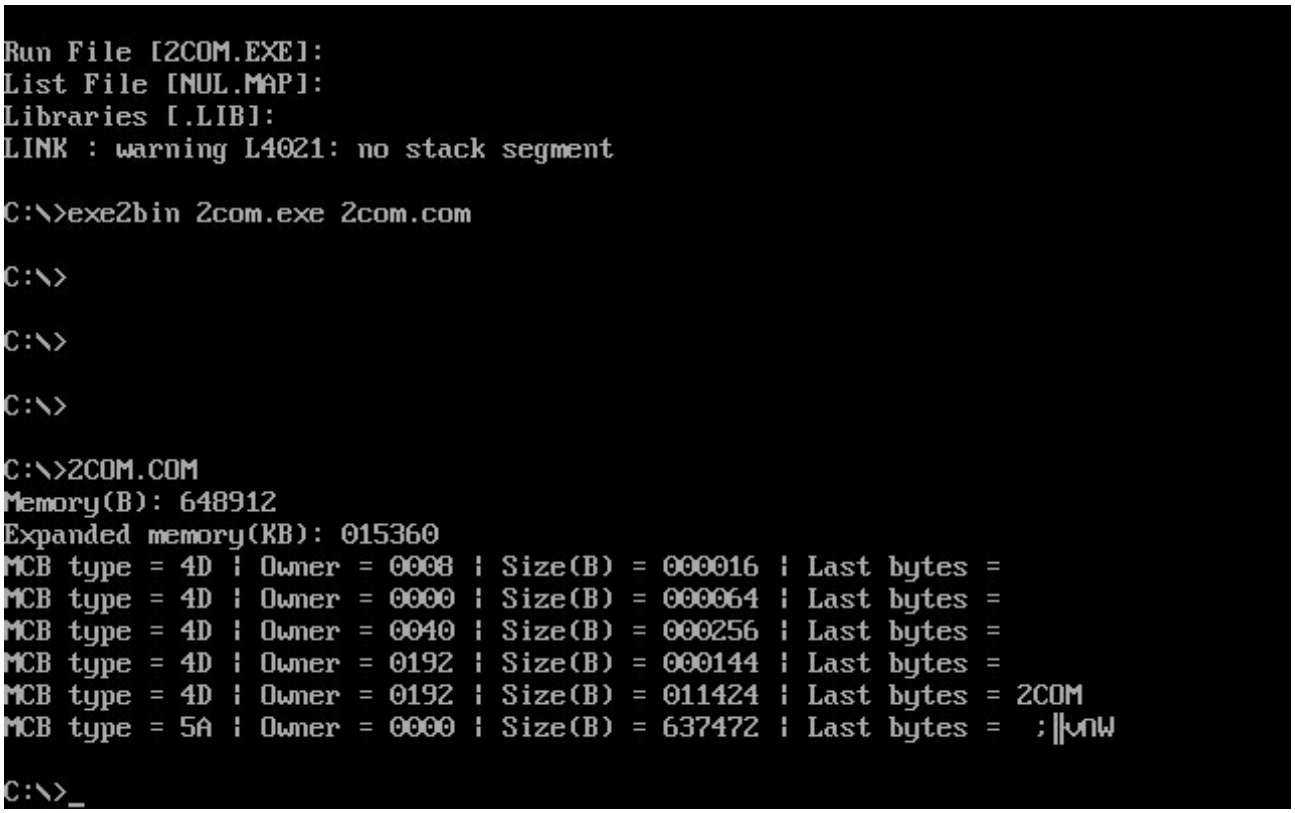
Run File [2COM.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
C:\>exe2bin 2com.exe 2com.com
C:\>
C:\>
C:\>
C:\>2COM.COM
Memory(B): 648912
Expanded memory(KB): 015360
MCB type = 4D ¦ Owner = 0008 ¦ Size(B) = 000016 ¦ Last bytes =
MCB type = 4D ¦ Owner = 0000 ¦ Size(B) = 000064 ¦ Last bytes =
MCB type = 4D ¦ Owner = 0040 ¦ Size(B) = 000256 ¦ Last bytes =
MCB type = 4D ¦ Owner = 0192 ¦ Size(B) = 000144 ¦ Last bytes =
MCB type = 4D ¦ Owner = 0192 ¦ Size(B) = 011424 ¦ Last bytes = 2COM
MCB type = 5A ¦ Owner = 0000 ¦ Size(B) = 637472 ¦ Last bytes = ;‖vnW
C:\>_

Рис. 2

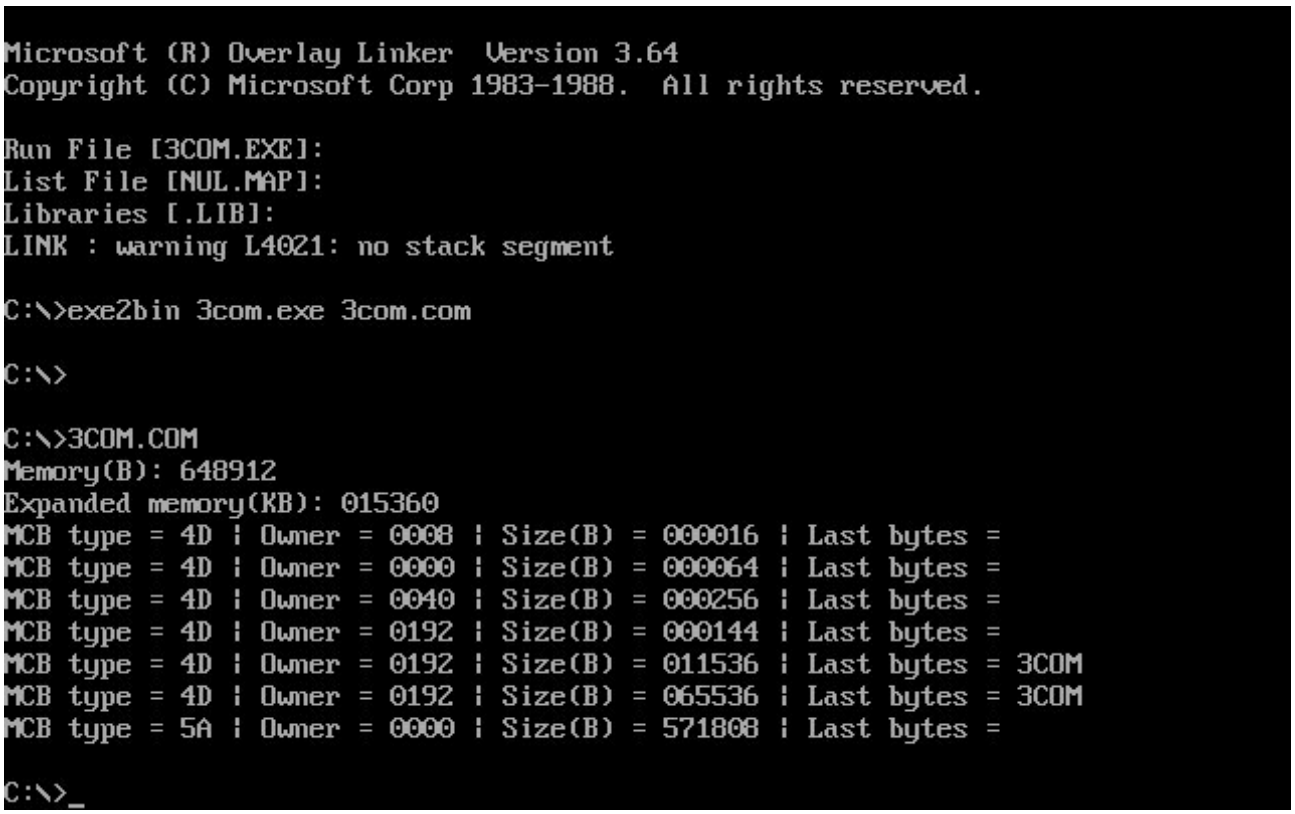

Рис. 3

исх. код в приложении Г). Результат работы приведён на рис. 4

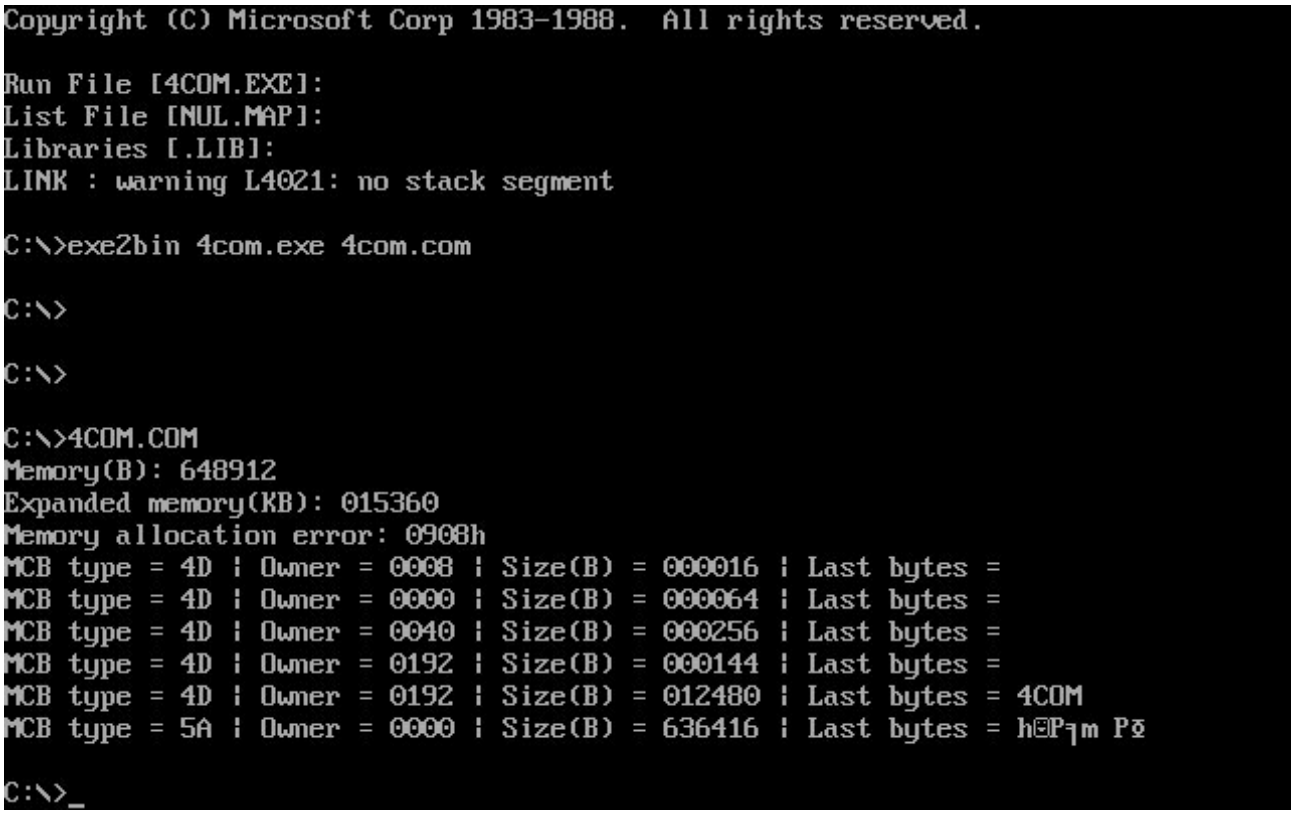

Рис. 4

Контрольные вопросы

1) Что означает "доступный объем памяти"?

Объем памяти, который доступен для программы.

2) Где MCB блок Вашей программы в списке?

1ая версия программы - 4 и 5 блоки

2ая версия программы - 4 и 5 блоки

3ья версия программы - 4, 5 и 6 блоки

4ая версия программы - 4 и 5 блоки

3) Какой размер памяти занимает программа в каждом случае?

1ая версия программы - 648912Б + 144Б = 649056Б

2ая версия программы - 11424Б + 144Б = 11568Б

3ья версия программы - 11536Б + 144Б + 65536Б = 77216Б

4ая версия программы - 12480Б + 144Б = 12624Б

Вывод

В ходе лабораторной работы была исследована работа функций управления памятью ядра операционной системы.

Приложение А. Исходный код программы (Версия 1)

```
testpc segment
     assume CS:testpc, ds:testpc, es:nothing, ss:
       nothing
     org 100h
start: jmp begin

memory_label db 'Memory(B): ', '$'
memory_value db '000000', 13, 10, '$'

expanded_memory_label db 'Expanded memory(KB): ', '$'
expanded_memory_value db '000000', 13, 10, '$'

mcb_start db 'MCB type = ', '$'
mcb_owner db ' | Owner = ', '$'
mcb_owner_value db '0000', '$'
mcb_size db ' | Size(B) = ', '$'
mcb_size_value db '00000', '$'
mcb_last db " | Last bytes = ", '$'
rn db 13, 10, '$'

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

wrd_to_dec proc near
  push bx
  mov bx, 10
  in_loop:
    div bx
    add dl, 030h
    mov [si], dl
    xor dx, dx
    dec si
    cmp ax, 0
    jne in_loop
  pop bx
  ret
wrd_to_dec endp

begin:
  mov dx, offset memory_label
  mov ah, 09h
  int 21h

  mov ah, 04ah
  mov bx, 0ffffh
  int 21h
```

```
mov ax, 16
mul bx

mov si, offset memory_value
add si, 5

call wrd_to_dec

mov dx, offset memory_value
mov ah, 09h
int 21h

mov dx, offset expanded_memory_label
int 21h

mov al, 030h
out 070h, al
in al, 071h
mov bl, al
mov al, 031h
out 070h, al
in al, 071h

mov si, offset expanded_memory_value
add si, 5

mov ah, al
```

```
mov al, bl
mov dx, 0
call wrd_to_dec

mov dx, offset expanded_memory_value
mov ah, 09h
int 21h



mov ah, 052h
int 21h

mov es, es:[bx - 2]

mcb_print:
  mov dx, offset mcb_start
  mov ah, 09h
  int 21h

  mov al, es:[0h]
  call byte_to_hex
  mov cx, ax
  mov dl, cl
  mov ah, 02h
  int 21h
  mov dl, ch
  int 21h
```

```
mov dx, offset mcb_owner
mov ah, 09h
int 21h

mov ax, es:[01h]
mov di, offset mcb_owner_value
add di, 3
call wrd_to_hex

mov dx, offset mcb_owner_value
mov ah, 09h
int 21h

mov dx, offset mcb_size
mov ah, 09h
int 21h

mov ax, es:[03h]
mov bx, 16
mul bx
mov si, offset mcb_size_value
add si, 5
call wrd_to_dec
mov dx, offset mcb_size_value
mov ah, 09h
int 21h
```

```
  mov si, 08h
  mov cx, 8
  mov ah, 02h

  last_print:
    mov dl, es:[si]
    int 21h
    inc si
    loop last_print

  mov dx, offset rn
  mov ah, 09h
  int 21h

  mov al, es:[0h]
  cmp al, 05ah
  je finish

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

finish:
  xor al, al
```

```
    mov ah, 4ch
    int 21h

testpc ends
      end start
```

Приложение Б. Исходный код программы (Версия 2)

```
testpc segment
     assume CS:testpc, ds:testpc, es:nothing, ss:
       nothing
     org 100h
start: jmp begin

memory_label db 'Memory(B): ', '$'
memory_value db '000000', 13, 10, '$'

expanded_memory_label db 'Expanded memory(KB): ', '$'
expanded_memory_value db '000000', 13, 10, '$'

mcb_start db 'MCB type = ', '$'
mcb_owner db ' | Owner = ', '$'
mcb_owner_value db '0000', '$'
mcb_size db ' | Size(B) = ', '$'
mcb_size_value db '00000', '$'
mcb_last db " | Last bytes = ", '$'
rn db 13, 10, '$'

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

wrd_to_dec proc near
  push bx
  mov bx, 10
  in_loop:
    div bx
    add dl, 030h
    mov [si], dl
    xor dx, dx
    dec si
    cmp ax, 0
    jne in_loop
  pop bx
  ret
wrd_to_dec endp

begin:
  mov dx, offset memory_label
  mov ah, 09h
  int 21h

  mov ah, 04ah
  mov bx, 0ffffh
  int 21h
```

```
mov ax, 16
mul bx

mov si, offset memory_value
add si, 5

call wrd_to_dec

mov dx, offset memory_value
mov ah, 09h
int 21h

mov dx, offset expanded_memory_label
int 21h

mov al, 030h
out 070h, al
in al, 071h
mov bl, al
mov al, 031h
out 070h, al
in al, 071h

mov si, offset expanded_memory_value
add si, 5

mov ah, al
```

```
mov al, bl
mov dx, 0
call wrd_to_dec

mov dx, offset expanded_memory_value
mov ah, 09h
int 21h

mov bx, offset end_label
mov ah, 04ah
int 21h

mov ah, 052h
int 21h

mov es, es:[bx - 2]

mcb_print:
  mov dx, offset mcb_start
  mov ah, 09h
  int 21h

  mov al, es:[0h]
  call byte_to_hex
  mov cx, ax
  mov dl, cl
  mov ah, 02h
```

```
int 21h
mov dl, ch
int 21h

mov dx, offset mcb_owner
mov ah, 09h
int 21h

mov ax, es:[01h]
mov di, offset mcb_owner_value
add di, 3
call wrd_to_hex

mov dx, offset mcb_owner_value
mov ah, 09h
int 21h

mov dx, offset mcb_size
mov ah, 09h
int 21h

mov ax, es:[03h]
mov bx, 16
mul bx
mov si, offset mcb_size_value
add si, 5
call wrd_to_dec
```

```
mov dx, offset mcb_size_value
mov ah, 09h
int 21h

mov si, 08h
mov cx, 8
mov ah, 02h

last_print:
  mov dl, es:[si]
  int 21h
  inc si
  loop last_print

mov dx, offset rn
mov ah, 09h
int 21h

mov al, es:[0h]
cmp al, 05ah
je finish

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

```
  finish:
    xor al, al
    mov ah, 4ch
    int 21h

end_label:

testpc ends
     end start
```

Приложение В. Исходный код программы (Версия 3)

```
testpc segment
      assume CS:testpc, ds:testpc, es:nothing, ss:
        nothing
      org 100h
start: jmp begin

memory_label db 'Memory(B): ', '$'
memory_value db '000000', 13, 10, '$'

expanded_memory_label db 'Expanded memory(KB): ', '$'
expanded_memory_value db '000000', 13, 10, '$'

mcb_start db 'MCB type = ', '$'
mcb_owner db ' | Owner = ', '$'
mcb_owner_value db '0000', '$'
mcb_size db ' | Size(B) = ', '$'
mcb_size_value db '00000', '$'
mcb_last db " | Last bytes = ", '$'
rn db 13, 10, '$'

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

wrd_to_dec proc near
  push bx
  mov bx, 10
  in_loop:
    div bx
    add dl, 030h
    mov [si], dl
    xor dx, dx
    dec si
    cmp ax, 0
    jne in_loop
  pop bx
  ret
wrd_to_dec endp

begin:
  mov dx, offset memory_label
  mov ah, 09h
  int 21h

  mov ah, 04ah
  mov bx, 0ffffh
  int 21h
```

```
mov ax, 16
mul bx

mov si, offset memory_value
add si, 5

call wrd_to_dec

mov dx, offset memory_value
mov ah, 09h
int 21h

mov dx, offset expanded_memory_label
int 21h

mov al, 030h
out 070h, al
in al, 071h
mov bl, al
mov al, 031h
out 070h, al
in al, 071h

mov si, offset expanded_memory_value
add si, 5

mov ah, al
```

```
mov al, bl
mov dx, 0
call wrd_to_dec

mov dx, offset expanded_memory_value
mov ah, 09h
int 21h

mov bx, offset end_label
mov ah, 04ah
int 21h

mov bx, 01000h
mov ah, 048h
int 21h

mov ah, 052h
int 21h

mov es, es:[bx - 2]

mcb_print:
  mov dx, offset mcb_start
  mov ah, 09h
  int 21h

  mov al, es:[0h]
```

```
call byte_to_hex
mov cx, ax
mov dl, cl
mov ah, 02h
int 21h
mov dl, ch
int 21h

mov dx, offset mcb_owner
mov ah, 09h
int 21h

mov ax, es:[01h]
mov di, offset mcb_owner_value
add di, 3
call wrd_to_hex

mov dx, offset mcb_owner_value
mov ah, 09h
int 21h

mov dx, offset mcb_size
mov ah, 09h
int 21h

mov ax, es:[03h]
mov bx, 16
```

```
mul bx
mov si, offset mcb_size_value
add si, 5
call wrd_to_dec
mov dx, offset mcb_size_value
mov ah, 09h
int 21h

mov si, 08h
mov cx, 8
mov ah, 02h

last_print:
  mov dl, es:[si]
  int 21h
  inc si
  loop last_print

mov dx, offset rn
mov ah, 09h
int 21h

mov al, es:[0h]
cmp al, 05ah
je finish

mov ax, es
```

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

  finish:
    xor al, al
    mov ah, 4ch
    int 21h

end_label:

testpc ends
      end start
```

Приложение Г. Исходный код программы (Версия 4)

```
testpc segment
     assume CS:testpc, ds:testpc, es:nothing, ss:
       nothing
     org 100h
start: jmp begin

memory_label db 'Memory(B): ', '$'
memory_value db '000000', 13, 10, '$'

expanded_memory_label db 'Expanded memory(KB): ', '$'
expanded_memory_value db '000000', 13, 10, '$'

mcb_start db 'MCB type = ', '$'
mcb_owner db ' | Owner = ', '$'
mcb_owner_value db '0000', '$'
mcb_size db ' | Size(B) = ', '$'
mcb_size_value db '00000', '$'
mcb_last db " | Last bytes = ", '$'
rn db 13, 10, '$'

error_48h db 'Memory allocation error: ', '$'
error_code db '0000h', 13, 10, '$'

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

wrd_to_dec proc near
  push bx
  mov bx, 10
  in_loop:
    div bx
    add dl, 030h
    mov [si], dl
    xor dx, dx
    dec si
    cmp ax, 0
    jne in_loop
  pop bx
  ret
wrd_to_dec endp

begin:
  mov dx, offset memory_label
  mov ah, 09h
  int 21h

  mov ah, 04ah
```

```
mov bx, 0ffffh
int 21h

mov ax, 16
mul bx

mov si, offset memory_value
add si, 5

call wrd_to_dec

mov dx, offset memory_value
mov ah, 09h
int 21h

mov dx, offset expanded_memory_label
int 21h

mov al, 030h
out 070h, al
in al, 071h
mov bl, al
mov al, 031h
out 070h, al
in al, 071h

mov si, offset expanded_memory_value
```

```
add si, 5

mov ah, al
mov al, bl
mov dx, 0
call wrd_to_dec

mov dx, offset expanded_memory_value
mov ah, 09h
int 21h

mov bx, 01000h
mov ah, 048h
int 21h
jnc continue

mov dx, offset error_48h
mov ah, 09h
int 21h

mov di, offset error_code
add di, 3
call wrd_to_hex

mov dx, offset error_code
mov ah, 09h
int 21h
```

```
continue:

mov bx, offset end_label
mov ah, 04ah
int 21h



mov ah, 052h
int 21h

mov es, es:[bx - 2]

mcb_print:
  mov dx, offset mcb_start
  mov ah, 09h
  int 21h

  mov al, es:[0h]
  call byte_to_hex
  mov cx, ax
  mov dl, cl
  mov ah, 02h
  int 21h
  mov dl, ch
  int 21h
```

```
mov dx, offset mcb_owner
mov ah, 09h
int 21h

mov ax, es:[01h]
mov di, offset mcb_owner_value
add di, 3
call wrd_to_hex

mov dx, offset mcb_owner_value
mov ah, 09h
int 21h

mov dx, offset mcb_size
mov ah, 09h
int 21h

mov ax, es:[03h]
mov bx, 16
mul bx
mov si, offset mcb_size_value
add si, 5
call wrd_to_dec
mov dx, offset mcb_size_value
mov ah, 09h
int 21h
```

```
  mov si, 08h
  mov cx, 8
  mov ah, 02h

  last_print:
    mov dl, es:[si]
    int 21h
    inc si
    loop last_print

  mov dx, offset rn
  mov ah, 09h
  int 21h

  mov al, es:[0h]
  cmp al, 05ah
  je finish

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

finish:
  xor al, al
  mov ah, 4ch
```

```
   int 21h

end_label:

testpc ends
     end start
```