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. 9382	 Юрьев С.Ю.
Преподаватель	 Ефремов М.А

Санкт-Петербург

Цель работы.

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

Сведения, использованные для составления программы.

Учет занятой и свободной памяти ведется при помощи списка блоков управления памятью MCB (Memory Control Block).

МСВ занимает 16 байт (параграф) и располагается всегда с адреса кратного 16 (адрес сегмента ОП) и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет.

Структура МСВ представлена в табл. 1.

Смещение	Длина поля (байт)	Содержимое поля
00h	1	тип МСВ:
		5Ah, если последний в списке,
	4Dh, если не последний	
01h	2	Сегментный адрес PSP владельца участка памяти, либо 0000h - свободный участок, 0006h - участок принадлежит драйверу OS XMS UMB 0007h - участок является исключенной верхней памятью драйверов 0008h - участок принадлежит MS DOS FFFAh - участок занят управляющим блоком 386MAX UMB FFFDh - участок заблокирован 386MAX
03h	2	FFFEh - участок принадлежит 386MAX UMB Размер участка в параграфах
05h	3	Зарезервирован
100 M	750	A A A
08h	8	"SC" - если участок принадлежит MS DOS, то в нем системный код "SD" - если участок принадлежит MS DOS, то в нем системные данные

Табл. 1. Структура МСВ.

По сегментному адресу и размеру участка памяти, контролируемого этим МСВ можно определить местоположение следующего МСВ в списке.

Адрес первого MCB хранится во внутренней структуре MS DOS,

называемой "List of Lists" (список списков). Доступ к указателю на эту структуру можно получить, используя функцию 52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:ВХ будет указывать на список списков. Слово по адресу ES:[ВХ-2] и есть адрес самого первого МСВ.

Размер расширенной памяти находится в ячейках 30h, 31h CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта.

Ход работы.

Был написан и отлажен программный модуль типа .СОМ, который выводит на экран следующую информацию:

- 1) Количество доступной памяти
- 2) Размер расширенной памяти
- 3) Цепочку блоков управления памятью На рис. 1 представлен вывод программы.

```
:\>LAB3 1.COM
Available memory: 648912 b
Extended memory: 15360 kb
                 MS DOS
                                    Size:
                                               16
      1 Owner:
last 8 bytes:
CB:
      2 Owner:
                 free
                                    Size:
                                              64
last 8 bytes:
                 0040
MCB:
      3 Owner:
                                    Size:
                                              256
last 8 butes:
                 0192
                                    Size:
                                              144
1CB:
      4 Owner:
last 8 bytes:
      5 Owner:
                                    Size: 648912
last 8 bytes: LAB3_1
```

Рис. 1. Результат выполнения lab3 1.com

По рисунку видно, что программа занимает всю свободную память.

Далее был получен lab3_2.com, в котором была освобождена память, которую программа не занимает. На рис. 2 представлен вывод программы.

```
C:\>LAB3_Z.COM
Available memory: 648912 b
Extended memory: 15360 kb
                                  Size:
                                            16
     1 Owner: MS DOS
last 8 bytes:
MCB: 2 Owner: free
                                  Size:
                                            64
last 8 bytes:
ICB: 3 Owner: 0040
                                           256
                                  Size:
last 8 bytes:
1CB: 4 Owner: 0192
                                  Size:
                                           144
last 8 butes:
    5 Owner: 0192
                                  Size:
                                           944
1CB:
last 8 bytes: LAB3_2
ICB:
                                  Size: 647952
    6 Owner: free
last 8 bytes: u4 60 Aδ
```

Рис. 2. Результат выполнения lab3 2.com

По рисунку видно, что программа занимает не всю свободную память. Освобожденная память находится в шестом блоке.

Далее был получен lab3_3.com, в котором после освобождения памяти программа запрашивает 64 Кб памяти. На рис. 3 представлен вывод программы.

```
C:\>LAB3_3.COM
Available memory: 648912 b
No error
Extended memory: 15360 kb
MCB: 1 Owner: MS DOS
                                 Size:
                                           16
last 8 bytes:
MCB: 2 Owner: free
                                 Size:
                                          64
last 8 bytes:
MCB: 3 Owner: 0040
                                 Size:
                                          256
last 8 bytes:
                                 Size:
                                          144
MCB: 4 Owner: 0192
last 8 bytes:
MCB: 5 Owner: 0192
                                 Size:
                                         1008
last 8 bytes: LAB3_3
                                 Size: 65536
MCB: 6 Owner: 0192
last 8 bytes: LAB3_3
                                 Size: 582336
     7 Owner: free
last 8 bytes: 1@error
```

Рис. 3. Результат выполнения lab3_3.com

По рисунку видно, что дополнительно выделенная память относится к шестому блоку. Сегментный адрес PSP владельца участка памяти для пятого и шестого блока совпадают.

Далеебыл получен lab3_4.com, в котором программа сначала запрашивает дополнительно 64 Кб, а потом освобождает память. На рис. 4 представлен вывод программы.

```
C:\>LAB3 4.COM
Available memory: 648912 b
Error
Extended memory: 15360 kb
                                  Size:
                                            16
     1 Owner:
                MS DOS
last 8 bytes:
MCB: 2 Owner: free
                                  Size:
                                            64
last 8 bytes:
1CB: 3 Owner: 0040
                                  Size:
                                           256
last 8 bytes:
MCB: 4 Owner:
                0192
                                  Size:
                                           144
last 8 bytes:
MCB: 5 Owner:
                0192
                                  Size:
                                          1008
last 8 bytes: LAB3 4
MCB: 6 Owner: free
                                  Size: 647888
last 8 bytes: 🗷 &8%u
```

Рис. 3. Результат выполнения lab3 4.com

По рисунку видно, что при попытке выделить дополнительную память произошла ошибка, поэтому в отличие от предыдущего варианты программы дополнительная память не была выделена.

Ответы на контрольные вопросы

Контрольные вопросы по лабораторной работе №3

1) Что означает "доступный объем памяти"?

Ответ: объём памяти, который занимает и использует программа

2) Где МСВ блок Вашей программы в списке?

Ответ: он находится в конце списка МСВ блоков, перед ним располагается среда, которая тоже принадлежит программе, его можно отследить по одинаковому сегментному адресу PSP

3) Какой размер памяти занимает программа в каждом случае?

Ответ: изначально, программа занимает всю доступную память(649056 байта), потом только то, что ей оставил программист(1404 байта) после очищения неиспользуемой памяти, затем этот же размер, а также 64 Кб, которые выделили, а потом снова всю доступную память(649056 байта).

Выводы.

В ходе выполнения данной работы были исследованы структуры данных и работа функций управления памятью ядра ОС. Были рассмотрены нестраничная память и способ управления динамическими разделами.

ПРИЛОЖЕНИЕ А.

Исходный код программы lab3_1.asm.

```
TESTPC SEGMENT
           ASSUME
                      CS:TESTPC, DS:TESTPC,
ES:NOTHING, SS:NOTHING
           ORG 100H
START:
           JMP
                 BEGIN
; Данные
                      'Available memory:
AvailableMem
                 db
b',0DH,0AH,'$'
                            'Extended memory:
ExtMem
                      db
kb',0DH,0AH,'$'
                                    $'
MCB
                      db
                            'MCB:
                 db
                      'Owner: $'
Owner
AreaSize
                 db
                      'Size:
                               $'
                 db
                      0DH,0AH,'last 8 bytes: $'
LastBytes
FREE
                 db 'free $'
                 db 'OS XMS UMB
XMS
TM
                      db ' driver memory
DOS
                 db 'MS DOS
                 db 'busy by 386MAX UMB $'
Busy
Block
                 db 'blocked by 386MAX $'
OWN_386
                 db '386MAX UMB
                                      $'
Empty
                 db
EndL
                 db 0Dh, 0Ah, '$'
; Процедуры
TETR_TO_HEX PROC near
           and AL,0Fh
           cmp AL,09
           ibe 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 АХ
           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
                                  ;в АН
           pop CX
младшая
           ret
BYTE TO HEX ENDP
```

```
WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в АХ - число, DI - адрес последнего символа
            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
; перевод в 10c/c, SI - адрес поля младшей
цифры
            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
PRINT proc near
            mov ah, 09h
            int 21h
            ret
PRINT endp
WORD_TO_DEC PROC near
            push cx
            push dx
```

```
mov cx, 10
loop1:
           div cx
                 dl, 30h
           or
           mov [si], dl
           dec si
           xor dx, dx
           cmp ax, 0
           jnz loop1
           pop dx
           pop cx
           ret
WORD_TO_DEC ENDP
;-----
; Код
BEGIN:
;available mem
           mov ah, 4ah
           mov bx, 0ffffh
           int 21h
           mov ax, bx
           mov cx, 16
           mul cx
           mov si, offset AvailableMem + 23
           call word_to_dec
           mov dx, offset AvailableMem
           call print
!-----
;extended mem
           xor ax, ax
           xor dx, dx
           mov AL, 30h; запись адреса ячейки
CMOS
           out 70h, AL
           in AL, 71h ; чтение младшего байта
           mov BL, AL ; расщиренной памяти
           mov AL, 31h; запись адреса ячейки
CMOS
           out 70h, AL
           in AL, 71h ; чтение старшего байта
                            ; размера
расширенной памяти
           mov bh, al
           mov ax, bx
           mov si, offset extmem+22
           call word to dec
           mov dx, offset extmem
           call print
;MCB
```

```
xor ax, ax
           mov ah, 52h
           int 21h
           mov ax, es:[bx-2]
           mov es, ax
           xor cx, cx
           inc cx
next mcb:
;mcb number.....
           mov si, offset mcb+7
           mov al, cl
           push cx
           call byte to dec
           mov dx, offset mcb
           call print
;owner.....
           mov dx, offset owner
           call print
           xor
                 ah, ah
           mov al, es:[0]
           push ax
           mov ax, es:[1]
           cmp ax, 0
           mov dx, offset free
                 printOwn
           ie
                ax, 6
           cmp
           mov dx, offset xms
                 printOwn
           je
           cmp ax, 7
           mov dx, offset tm
           je
                 printOwn
           cmp ax, 8
           mov dx, offset dos
                 printOwn
           je
           cmp ax, Offfah
           mov dx, offset busy
           ie
                 printOwn
           cmp ax, 0fffdh
           mov dx, offset block
                 printOwn
           je
                ax, Offfeh
           cmp
           mov dx, offset own 386
                 printOwn
           je
           mov di, offset empty+4
           call wrd to hex
           mov dx, offset empty
printOwn:
           call print
;size.....
           mov ax, es:[3]
           mov cx, 16
           mul cx
```

```
mov si, offset areasize+11
           call word_to_dec
           mov dx, offset areasize
           call print
;data.....
           xor
                 dx, dx
           mov dx, offset lastbytes
           call print
           mov cx, 8
                 di, di
           xor
symbol:
           mov dl, es:[di+8]
           mov ah, 02h
           int 21h
           inc di
           loop symbol
           mov dx, offset endl
           call print
;.....
           mov ax,es:[3]
           mov bx,es
           add bx,ax
           inc bx
           mov es,bx
           pop ax
           рор сх
           inc cx
           cmp al,5Ah; проверка на не
последний ли это сегмент
           je
                 exit
           cmp al,4Dh
           jne exit
           jmp next_mcb
exit: ; Выход в DOS
           xor AL,AL
           mov AH,4Ch
           int 21H
TESTPC ENDS
           END START
                  Исходный код программы lab3 2.asm.
TESTPC SEGMENT
           ASSUME
                       CS:TESTPC, DS:TESTPC,
ES:NOTHING, SS:NOTHING
           ORG 100H
                 BEGIN
START:
           IMP
```

; Данные AvailableMem

b',0DH,0AH,'\$'

db

'Available memory:

```
db
ExtMem
                             'Extended memory:
kb',0DH,0AH,'$'
                                     $'
                       db
                             'MCB:
MCB
Owner
                 db
                       'Owner: $'
                 db
                       'Size:
                                $'
AreaSize
                       0DH,0AH,'last 8 bytes: $'
LastBytes
                 db
                 db 'free
FREE
XMS
                 db 'OS XMS UMB
TM
                       db 'driver memory
                 db 'MS DOS
DOS
                                    $'
Busy
                 db 'busy by 386MAX UMB $'
                 db 'blocked by 386MAX $'
Block
                 db '386MAX UMB
                                       $'
OWN 386
Empty
                 db
                                   $'
                 db 0Dh, 0Ah, '$'
EndL
; Процедуры
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 АХ
           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 АН
младшая
           ret
BYTE_TO_HEX ENDP
!-----
WRD TO HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в АХ - число, DI - адрес последнего символа
           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
; перевод в 10c/c, SI - адрес поля младшей
цифры
            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 I
            or AL,30h
            mov [SI],AL
end_l:
            pop DX
            pop CX
            ret
BYTE_TO_DEC ENDP
PRINT proc near
            mov ah, 09h
            int 21h
            ret
PRINT endp
WORD_TO_DEC PROC near
            push cx
            push dx
            mov cx, 10
loop1:
            div cx
                  dl, 30h
            or
            mov [si], dl
            dec si
            xor dx, dx
            cmp ax, 0
            jnz loop1
            pop dx
```

```
pop cx
            ret
WORD_TO_DEC ENDP
FREE_MEM PROC
            push ax
            push bx
            push cx
           push dx
           lea ax, endofprogram
           mov bx, 10h
           xor dx, dx
            div bx
           inc ax
           mov bx, ax
           mov al, 0
           mov ah, 4Ah
           int 21h
            pop dx
            pop cx
            pop bx
            pop ax
           ret
FREE MEM
            ENDP
; Код
BEGIN:
;available mem
           mov ah, 4ah
           mov bx, 0ffffh
           int 21h
           mov ax, bx
           mov cx, 16
           mul cx
           mov si, offset AvailableMem + 23
            call word to dec
           mov dx, offset AvailableMem
           call print
           call free mem
;-----
;extended mem
           xor ax, ax
           xor dx, dx
           mov AL, 30h; запись адреса ячейки
CMOS
            out 70h, AL
            in AL, 71h ; чтение младшего байта
           mov BL, AL ; расщиренной памяти
```

```
mov AL, 31h; запись адреса ячейки
CMOS
           out 70h, AL
           in AL, 71h; чтение старшего байта
                             ; размера
расширенной памяти
           mov bh, al
           mov ax, bx
           mov si, offset extmem+22
           call word to dec
           mov dx, offset extmem
           call print
;-----
;MCB
           xor
                 ax, ax
           mov ah, 52h
           int 21h
           mov ax, es:[bx-2]
           mov es, ax
           xor cx, cx
           inc
                 \mathsf{CX}
next_mcb:
;mcb number.....
           mov si, offset mcb+7
           mov al, cl
           push cx
           call byte_to_dec
           mov dx, offset mcb
           call print
;owner.....
           mov dx, offset owner
           call print
           xor
                 ah, ah
           mov al, es:[0]
           push ax
           mov ax, es:[1]
           cmp ax, 0
           mov dx, offset free
                 printOwn
           je
           cmp ax, 6
           mov dx, offset xms
           ie
                 printOwn
                 ax, 7
           cmp
                 dx, offset tm
           mov
           je
                 printOwn
                 ax, 8
           cmp
           mov dx, offset dos
                 printOwn
           je
                 ax, Offfah
           cmp
           mov dx, offset busy
                 printOwn
           je
           cmp ax, 0fffdh
           mov dx, offset block
```

```
printOwn
           je
                ax, Offfeh
           cmp
           mov dx, offset own_386
           je
                 printOwn
           mov di, offset empty+4
           call wrd_to_hex
           mov dx, offset empty
printOwn:
           call print
;size.....
           mov ax, es:[3]
           mov cx, 16
           mul cx
           mov si, offset areasize+11
           call word_to_dec
           mov dx, offset areasize
           call print
;data.....
                 dx, dx
           xor
           mov dx, offset lastbytes
           call print
           mov cx, 8
           xor di, di
symbol:
           mov dl, es:[di+8]
           mov ah, 02h
           int 21h
           inc di
           loop symbol
           mov dx, offset endl
           call print
mov ax,es:[3]
           mov bx,es
           add bx,ax
           inc bx
           mov es,bx
           pop ax
           pop cx
           inc cx
           cmp al,5Ah; проверка на не
последний ли это сегмент
                 exit
           je
           cmp al,4Dh
           jne exit
           jmp next_mcb
exit: ; Выход в DOS
           xor AL,AL
           mov AH,4Ch
           int 21H
```

endOfProgram:

END START

Исходный код программы lab3 3.asm.

```
TESTPC SEGMENT
                     CS:TESTPC, DS:TESTPC,
          ASSUME
ES:NOTHING, SS:NOTHING
          ORG 100H
START:
          JMP
                BEGIN
; Данные
AvailableMem
                db
                     'Available memory:
b',0DH,0AH,'$'
                           'Extended memory:
ExtMem
                     db
kb',0DH,0AH,'$'
                                   $'
MCB
                     db
                           'MCB:
                     'Owner: $'
Owner
                db
                db
                     'Size:
                             $'
AreaSize
                db
                     0DH,0AH,'last 8 bytes: $'
LastBytes
                db 'free $'
FREE
                db 'OS XMS UMB
XMS
TM
                     db ' driver memory
                db 'MS DOS
DOS
                db 'busy by 386MAX UMB $'
Busy
                db 'blocked by 386MAX $'
Block
                db '386MAX UMB
OWN 386
                                    $'
                db
Empty
ErrStr
                db
                     'Error',0Dh,0Ah,'$'
                db
                     'No error',0Dh,0Ah,'$'
noErrStr
                db 0Dh,0Ah,'$'
EndL
; Процедуры
TETR TO HEX PROC near
           and AL,0Fh
           cmp AL,09
          ibe 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 АХ
           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
                                  ;в АН
           pop CX
младшая
           ret
BYTE_TO_HEX ENDP
;-----
WRD TO HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в АХ - число, DI - адрес последнего символа
           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
; перевод в 10c/c, SI - адрес поля младшей
цифры
           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 I
           or AL,30h
           mov [SI],AL
end I:
           pop DX
           pop CX
           ret
BYTE TO DEC ENDP
PRINT proc near
           mov ah, 09h
```

```
int 21h
            ret
PRINT endp
;-----
WORD_TO_DEC PROC near
            push cx
            push dx
            mov cx, 10
loop1:
            div cx
                  dl, 30h
            or
            mov [si], dl
            dec si
            xor dx, dx
            cmp ax, 0
            jnz loop1
            pop dx
            pop cx
            ret
WORD_TO_DEC ENDP
            PROC
FREE_MEM
            push ax
            push bx
            push cx
            push dx
            lea ax, endofprogram
            mov bx, 10h
            xor dx, dx
            div bx
            inc ax
            mov bx, ax
            mov al, 0
            mov ah, 4Ah
            int 21h
            pop dx
            pop cx
            pop bx
            pop ax
            ret
FREE_MEM
            ENDP
; Код
BEGIN:
;available mem
            mov ah, 4ah
            mov bx, 0ffffh
            int 21h
            mov ax, bx
            mov cx, 16
```

```
mul cx
            mov si, offset AvailableMem + 23
            call word to dec
            mov dx, offset AvailableMem
            call print
           call free_mem
            push ax
            push bx
            push dx
            mov bx, 1000h
            mov ah, 48h
            int
                  21h
           јс err ;проверка флага CF
           jmp noerr
err:
            mov dx, offset errstr
            call print
           JMP end1
noerr:
            mov dx,offset noerrstr
            call print
end1:
            pop dx
            pop bx
            pop ax
;-----
;extended mem
           xor ax, ax
           xor dx, dx
            mov AL, 30h; запись адреса ячейки
CMOS
            out 70h, AL
            in AL, 71h ; чтение младшего байта
            mov BL, AL ; расщиренной памяти
            mov AL, 31h; запись адреса ячейки
CMOS
            out 70h, AL
            in AL, 71h ; чтение старшего байта
                             ; размера
расширенной памяти
            mov bh, al
            mov ax, bx
            mov si, offset extmem+22
            call word to dec
            mov dx, offset extmem
           call print
;MCB
```

```
xor ax, ax
           mov ah, 52h
           int 21h
           mov ax, es:[bx-2]
           mov es, ax
           xor cx, cx
           inc cx
next mcb:
;mcb number.....
           mov si, offset mcb+7
           mov al, cl
           push cx
           call byte to dec
           mov dx, offset mcb
           call print
;owner.....
           mov dx, offset owner
           call print
           xor
                 ah, ah
           mov al, es:[0]
           push ax
           mov ax, es:[1]
           cmp ax, 0
           mov dx, offset free
                 printOwn
           ie
           cmp ax, 6
           mov dx, offset xms
                 printOwn
           je
           cmp ax, 7
           mov dx, offset tm
           je
                 printOwn
           cmp ax, 8
           mov dx, offset dos
                 printOwn
           ie
           cmp ax, Offfah
           mov dx, offset busy
           ie
                 printOwn
           cmp ax, 0fffdh
           mov dx, offset block
                 printOwn
           je
                ax, Offfeh
           cmp
           mov dx, offset own 386
                 printOwn
           je
           mov di, offset empty+4
           call wrd to hex
           mov dx, offset empty
printOwn:
           call print
;size.....
           mov ax, es:[3]
           mov cx, 16
           mul cx
```

```
mov si, offset areasize+11
           call word_to_dec
           mov dx, offset areasize
           call print
;data.....
           xor dx, dx
           mov dx, offset lastbytes
           call print
           mov cx, 8
                 di, di
           xor
symbol:
           mov dl, es:[di+8]
           mov ah, 02h
           int 21h
           inc di
           loop symbol
           mov dx, offset endl
           call print
;.....
           mov ax,es:[3]
           mov bx,es
           add bx,ax
           inc bx
           mov es,bx
           pop ax
           рор сх
           inc cx
           cmp al,5Ah; проверка на не
последний ли это сегмент
           je
                 exit
           cmp al,4Dh
           jne exit
           jmp next_mcb
exit: ; Выход в DOS
           xor AL,AL
           mov AH,4Ch
           int 21H
endOfProgram:
TESTPC ENDS
           END START
```

Исходный код программы lab3_4.asm.

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC,

ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

; Данные

```
AvailableMem
                 db
                       'Available memory:
b',0DH,0AH,'$'
                       db
                             'Extended memory:
ExtMem
kb',0DH,0AH,'$'
                                     $'
MCB
                       db
                             'MCB:
                       'Owner: $'
Owner
                 db
                              $'
                       'Size:
AreaSize
                 db
                 db
                       0DH,0AH,'last 8 bytes: $'
lastBytes
                 db 'free
FREE
                 db 'OS XMS UMB
XMS
TM
                       db ' driver memory
DOS
                 db 'MS DOS
                                    $'
                 db 'busy by 386MAX UMB $'
Busy
                 db 'blocked by 386MAX $'
Block
OWN 386
                 db '386MAX UMB
                                      $'
                 db
Empty
                       'Error',0Dh,0Ah,'$'
ErrStr
                 db
                 db
                       'No error',0Dh,0Ah,'$'
noErrStr
                 db 0Dh,0Ah,'$'
EndL
; Процедуры
TETR TO HEX PROC near
           and AL,0Fh
           cmp AL,09
           ibe 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 АХ
           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 АН
младшая
           ret
BYTE_TO_HEX ENDP
:-----
WRD TO HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в АХ - число, DI - адрес последнего символа
           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
; перевод в 10с/с, SI - адрес поля младшей
цифры
            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 I
            or AL,30h
            mov [SI],AL
end_l:
            pop DX
            pop CX
            ret
BYTE_TO_DEC ENDP
PRINT proc near
            mov ah, 09h
            int 21h
            ret
PRINT endp
WORD_TO_DEC PROC near
            push cx
            push dx
            mov cx, 10
loop1:
            div cx
                  dl, 30h
            or
            mov [si], dl
            dec si
```

```
xor dx, dx
            cmp ax, 0
            jnz loop1
            pop dx
            pop cx
            ret
WORD_TO_DEC ENDP
FREE_MEM PROC
            push ax
            push bx
            push cx
            push dx
            lea ax, endofprogram
            mov bx, 10h
            xor dx, dx
            div bx
            inc ax
            mov bx, ax
            mov al, 0
            mov ah, 4Ah
            int 21h
            pop dx
            pop cx
            pop bx
            pop ax
            ret
FREE_MEM
            ENDP
; Код
BEGIN:
;available mem
            mov ah, 4ah
            mov bx, Offffh
            int 21h
            mov ax, bx
            mov cx, 16
            mul cx
            mov si, offset AvailableMem + 23
            call word_to_dec
            mov dx, offset AvailableMem
            call print
            push ax
            push bx
            push dx
            mov bx, 1000h
            mov ah, 48h
                  21h
            int
```

```
јс err ;проверка флага CF
           jmp noerr
err:
           mov dx, offset errstr
           call print
           JMP end1
noerr:
           mov dx,offset noerrstr
           call print
end1:
            pop dx
            pop bx
           pop ax
           call free_mem
;extended mem
           xor ax, ax
           xor dx, dx
           mov AL, 30h; запись адреса ячейки
CMOS
            out 70h, AL
           in AL, 71h ; чтение младшего байта
            mov BL, AL ; расщиренной памяти
            mov AL, 31h; запись адреса ячейки
CMOS
            out 70h, AL
           in AL, 71h ; чтение старшего байта
                             ; размера
расширенной памяти
           mov bh, al
           mov ax, bx
           mov si, offset extmem+22
           call word to dec
           mov dx, offset extmem
           call print
;-----
;MCB
           xor ax, ax
           mov ah, 52h
           int 21h
           mov ax, es:[bx-2]
           mov es, ax
           xor cx, cx
           inc
                 CX
next mcb:
;mcb number.....
           mov si, offset mcb+7
           mov al, cl
            push cx
```

```
call byte_to_dec
           mov dx, offset mcb
           call print
;owner.....
           mov dx, offset owner
           call print
           xor
                 ah, ah
           mov al, es:[0]
           push ax
           mov ax, es:[1]
                ax, 0
           cmp
           mov dx, offset free
                 printOwn
           je
           cmp ax, 6
           mov
                 dx, offset xms
                 printOwn
           je
                 ax, 7
           cmp
           mov
                 dx, offset tm
                 printOwn
           je
                 ax, 8
           cmp
           mov dx, offset dos
                 printOwn
           je
           cmp ax, Offfah
           mov dx, offset busy
                 printOwn
           je
           cmp ax, 0fffdh
           mov dx, offset block
                 printOwn
           je
           cmp ax, Offfeh
           mov dx, offset own 386
           je
                 printOwn
           mov di, offset empty+4
           call wrd to hex
           mov dx, offset empty
printOwn:
           call print
;size.....
           mov ax, es:[3]
           mov cx, 16
           mul cx
           mov si, offset areasize+11
           call word_to_dec
           mov dx, offset areasize
           call print
;data.....
                 dx, dx
           xor
           mov dx, offset lastbytes
           call print
           mov cx, 8
           xor
                 di, di
symbol:
```

mov dl, es:[di+8]

```
mov ah, 02h
           int 21h
           inc di
           loop symbol
           mov dx, offset endl
           call print
;.....
           mov ax,es:[3]
           mov bx,es
           add bx,ax
           inc bx
           mov es,bx
           pop ax
           pop cx
           inc cx
           cmp al,5Ah; проверка на не
последний ли это сегмент
                 exit
           je
           cmp al,4Dh
           jne exit
           jmp next_mcb
exit: ; Выход в DOS
           xor AL,AL
           mov AH,4Ch
           int 21H
endOfProgram:
TESTPC ENDS
           END START
```