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.

Студентка гр. 8382 _____ Наконечная А. Ю.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2020

Цель работы.

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

Постановка задачи.

Требуется написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

Количество доступной памяти.

Размер расширенной памяти.

Выводит цепочку блоков управления памятью.

Выполнение работы.

Была создана функция MAIN, которая позволяет распечатать информацию о: количестве доступной памяти, размере расширенной памяти. А также позволяет вывести цепочку блоков управления памятью.

Количество доступной памяти было найдено с помощью функции 4Ah прерывания 21h. Размер расширенной памяти был определен с помощью обращения к ячейкам 30h, 31h. Адрес первого блока управления памятью был получен с помощью функции 52h прерывания 21h.

В результате выполнения первой программы была получена информация, представленная на рисунке 1:

```
Available memory 648912 bytes

Extended memory 15360 bytes

MCB
Type 4D ¦ PSP segment 0008 ¦ Size 16     ¦ SC/SD
--------------------------------------------------------------------------------
Type 4D ¦ PSP segment 0000 ¦ Size 64     ¦ SC/SD DPMILOAD
--------------------------------------------------------------------------------
Type 4D ¦ PSP segment 0040 ¦ Size 256    ¦ SC/SD
--------------------------------------------------------------------------------
Type 4D ¦ PSP segment 0192 ¦ Size 144    ¦ SC/SD
--------------------------------------------------------------------------------
Type 5A ¦ PSP segment 0192 ¦ Size 648912 ¦ SC/SD LR3_1
--------------------------------------------------------------------------------
```

Рисунок 1 — результат выполнения программы lr3_1.com

После этого программа была изменена таким образом, чтобы она освобождала память, которую не занимает. В результате выполнения второй программы была получена информация, представленная на рисунке 2:

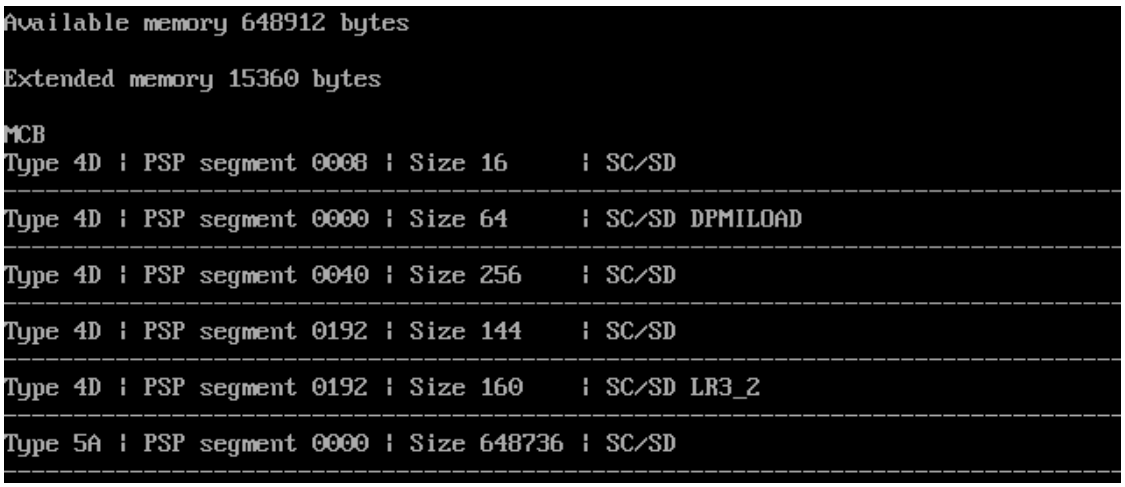

Рисунок 2 — результат выполнения программы lr3_2.com

Далее программа была изменена так, чтобы после освобождения памяти она запрашивала 64 Кб памяти функцией 48h прерывания 21h. В результате выполнения третьей программы была получена информация, представленная на рисунке 3:

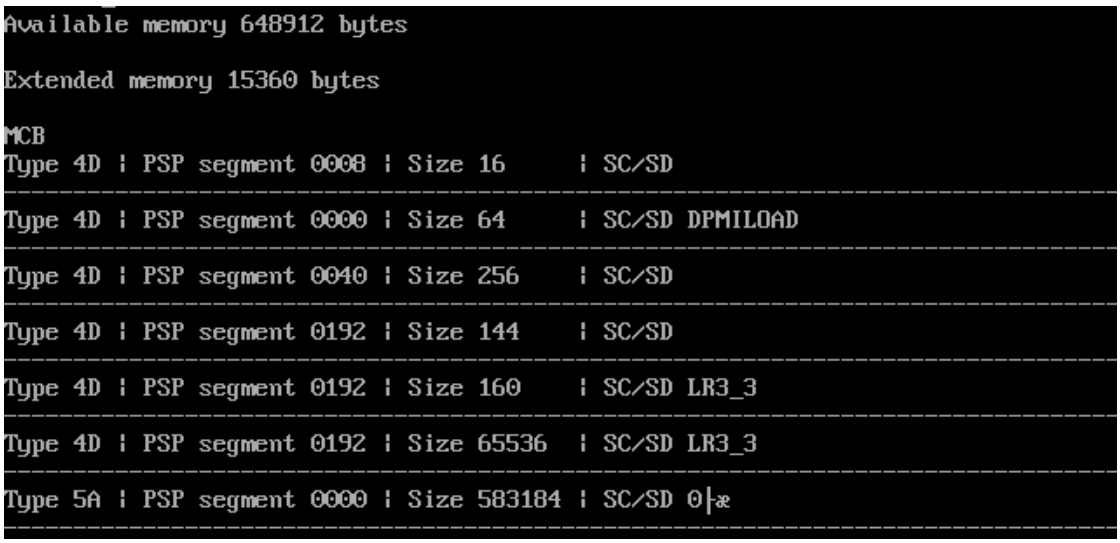

Рисунок 3 — результат выполнения программы lr3_3.com

В последней программе был представлен код первой программы, в который были добавлены запрос на выделение памяти размером 64 Кб и освобождение памяти. Необходимо учесть, что выделение памяти происходит до освобождения памяти, которую программа не занимает. В результате выполнения последней программы была получена информация, представленная на рисунке 4:

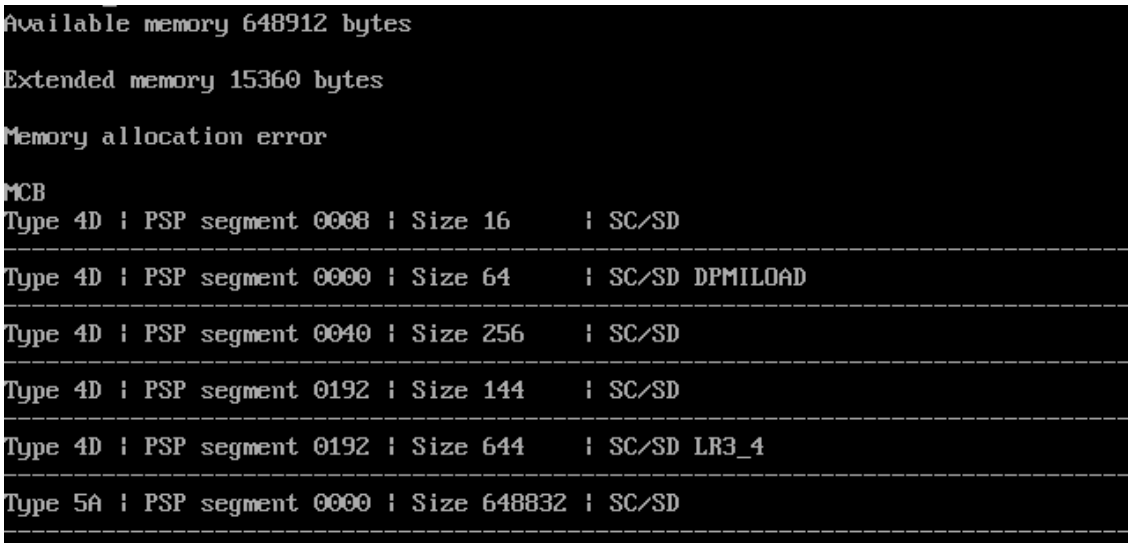

Рисунок 4 — результат выполнения программы lr3_4.com

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти

1) Что означает “Доступный объём памяти”?

Это тот объём памяти, который доступен для выполнения программы.

2) Где MCB блок Вашей программы в списке?

В первой и второй программе — это 4 и 5 блок. В третьей программе — это 4, 5 и 6 блок. В последней программе — это 4 и 5 блок.

3) Какой размер памяти занимает программа в каждом случае?

Первая программа занимает 649 056 байт.

Вторая программа занимает 304 байт.

Третья программа занимает 304 байт + 64 Кб, которые были запрошены

Последняя программа занимает 788 байт.

Выводы.

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

ПРИЛОЖЕНИЕ А

Исходный код программы

lr3_1.asm

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

; Данные
AVAILABLE_MEM_STR db 'Available memory        bytes', 0DH,0AH, '$'
EXTENDED_MEM_STR db 'Extended memory       bytes', 0DH,0AH, '$'
MCB_STR db 'MCB ', 0DH,0AH,'$'
TYPE_STR db 'Type $'
OWNER_ADRESS_STR db ' | PSP segment     $'
SIZE_STR db ' | Size        $'
SC_SD_STR db '| SC/SD $'
END_PRINT db 0DH,0AH,'$'
END_LINE                                                                    db
'-------------------------------------------------------------------------
-----$'

; 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 шестн. числа в AX
      push CX
      mov AH,AL
      call TETR_TO_HEX
      xchg AL,AH
      mov CL,4
      shr AL,CL
      call TETR_TO_HEX ;в AL старшая цифра
      pop CX         ;в AH младшая
      ret
BYTE_TO_HEX ENDP
;--------------------------------
```

```
WRD_TO_HEX PROC near
; перевод в 16 с/с 16-ти разрядного числа
; в AX - 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_l
            or AL,30h
            mov [SI],AL
      end_l:
            pop DX
            pop CX
            ret
BYTE_TO_DEC ENDP
;--------------------------------

SIZE_MEM PROC
      xor CX,CX
```

```
DIVISION:
      div BX
      push DX
      inc CX
      xor DX,DX
      cmp AX,0
            jz SYMBOL
      jmp DIVISION
SYMBOL:
      pop DX
      add DL,30h
      mov [DI],DL
      inc DI
      loop SYMBOL
      ret
SIZE_MEM ENDP

; Код
BEGIN:

MAIN PROC near
; Количество доступной памяти
      mov AH,4ah
      mov BX,0FFFFh
      int 21h
      mov AX,BX
      mov BX,16
      mul BX
      lea DI,AVAILABLE_MEM_STR + 17
      mov BX,10
      call SIZE_MEM
      mov DX,offset AVAILABLE_MEM_STR
      mov AH,09h
      int 21h
      mov DX, offset END_PRINT
      mov AH,09h
      int 21h

; Количество расширенной памяти
      mov AL,30h
      out 70h,AL
      in AL,71h
      mov BL,AL
      mov AL,31h
      out 70h,AL
      in AL,71h
      mov BH,AL
      mov AX,BX
      lea DI,EXTENDED_MEM_STR + 16
```

```
      mov DX,0
      mov BX,10
      call SIZE_MEM
      mov DX,offset EXTENDED_MEM_STR
      mov AH,09h
      int 21h
      mov DX,offset END_PRINT
      mov AH,09h
      int 21h

; Цепочка блоков управления памятью
      mov AH,52h
      int 21h
      mov CX,ES:[BX - 2]
      mov ES,CX
      mov DX,offset MCB_STR
      mov AH,09h
      int 21h

MCB:
; Тип MCB
      mov DX,offset TYPE_STR
      mov AH,09h
      int 21h
      mov AL, ES:[00h]
      push AX
      push DX
      push CX
      call BYTE_TO_HEX
      xor CX,CX
      mov CH,AH
      mov DL,AL
      mov AH,02h
      int 21h
      mov DL,CH
      mov AH,02h
      int 21h
      pop CX
      pop DX
      pop AX

; Адрес владельца
      mov AX,ES:[01h]
      lea DI,OWNER_ADRESS_STR + 18
      call WRD_TO_HEX
      mov DX,offset OWNER_ADRESS_STR
      mov AH,09h
      int 21h
```

```
; Размер
      mov AX,ES:[03h]
      lea DI,SIZE_STR + 8
      mov BX,16
      mul BX
      mov BX,10
      call SIZE_MEM
      mov DX,offset SIZE_STR
      mov AH,09h
      int 21h

; Последние 8 байтов
      mov DX,offset SC_SD_STR
      mov AH,09h
      int 21h
      mov BX,0
GO:
      mov DL,ES:[BX + 08h]
      mov AH,02h
      int 21h
      inc BX
      cmp BX,8
            jl GO
      mov DX,offset END_PRINT
      mov AH,09h
      int 21h
      mov  DX,offset END_LINE
      mov AH,09h
      int 21h
      mov AL,ES:[00h]
      cmp AL,5Ah
            je STOP
      xor CX,CX
      mov CX,ES:[03h]
      mov BX,ES
      add BX,CX
      inc BX
      mov ES,BX
      jmp MCB
STOP:
      ret



MAIN ENDP

call MAIN

; Выход в DOS
```

```
xor AL,AL
mov AH,4Ch
int 21H

TESTPC ENDS
END START
```

ПРИЛОЖЕНИЕ В

Исходный код программы

lr3_2.asm

```
ESTPC SEGMENT
ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
ORG 100H
START: JMP BEGIN

; Данные
AVAILABLE_MEM_STR db 'Available memory        bytes', 0DH,0AH, '$'
EXTENDED_MEM_STR db 'Extended memory       bytes', 0DH,0AH, '$'
MCB_STR db 'MCB ', 0DH,0AH,'$'
TYPE_STR db 'Type $'
OWNER_ADRESS_STR db ' | PSP segment     $'
SIZE_STR db ' | Size        $'
SC_SD_STR db '| SC/SD $'
END_PRINT db 0DH,0AH,'$'
END_LINE                                                                db
'-------------------------------------------------------------------------
----------$'

; 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 шестн. числа в AX
      push CX
      mov AH,AL
      call TETR_TO_HEX
      xchg AL,AH
      mov CL,4
      shr AL,CL
      call TETR_TO_HEX ;в AL старшая цифра
      pop CX         ;в AH младшая
      ret
BYTE_TO_HEX ENDP
;--------------------------------
```

```
WRD_TO_HEX PROC near
; перевод в 16 с/с 16-ти разрядного числа
; в AX - 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_l
            or AL,30h
            mov [SI],AL
      end_l:
            pop DX
            pop CX
            ret
BYTE_TO_DEC ENDP
;--------------------------------

SIZE_MEM PROC
      xor CX,CX
```

```
DIVISION:
      div BX
      push DX
      inc CX
      xor DX,DX
      cmp AX,0
            jz SYMBOL
      jmp DIVISION
SYMBOL:
      pop DX
      add DL,30h
      mov [DI],DL
      inc DI
      loop SYMBOL
      ret
SIZE_MEM ENDP

; Код
BEGIN:

MAIN PROC near
; Количество доступной памяти
      mov AH,4ah
      mov BX,0FFFFh
      int 21h
      mov AX,BX
      mov BX,16
      mul BX
      lea DI,AVAILABLE_MEM_STR + 17
      mov BX,10
      call SIZE_MEM
      mov DX,offset AVAILABLE_MEM_STR
      mov AH,09h
      int 21h
      mov DX, offset END_PRINT
      mov AH,09h
      int 21h

; Количество расширенной памяти
      mov AL,30h
      out 70h,AL
      in AL,71h
      mov BL,AL
      mov AL,31h
      out 70h,AL
      in AL,71h
      mov BH,AL
      mov AX,BX
      lea DI,EXTENDED_MEM_STR + 16
```

```
      mov DX,0
      mov BX,10
      call SIZE_MEM
      mov DX,offset EXTENDED_MEM_STR
      mov AH,09h
      int 21h
      mov DX,offset END_PRINT
      mov AH,09h
      int 21h

; Освобождение не занятой памяти
      mov AX,offset END_FREE
      mov ah,4Ah
      int 21h

; Цепочка блоков управления памятью
      mov AH,52h
      int 21h
      mov CX,ES:[BX - 2]
      mov ES,CX
      mov DX,offset MCB_STR
      mov AH,09h
      int 21h

MCB:
; Тип MCB
      mov DX,offset TYPE_STR
      mov AH,09h
      int 21h
      mov AL, ES:[00h]
      push AX
      push DX
      push CX
      call BYTE_TO_HEX
      xor CX,CX
      mov CH,AH
      mov DL,AL
      mov AH,02h
      int 21h
      mov DL,CH
      mov AH,02h
      int 21h
      pop CX
      pop DX
      pop AX

; Адрес владельца
      mov AX,ES:[01h]
      lea DI,OWNER_ADRESS_STR + 18
```

```
      call WRD_TO_HEX
      mov DX,offset OWNER_ADRESS_STR
      mov AH,09h
      int 21h

; Размер
      mov AX,ES:[03h]
      lea DI,SIZE_STR + 8
      mov BX,16
      mul BX
      mov BX,10
      call SIZE_MEM
      mov DX,offset SIZE_STR
      mov AH,09h
      int 21h

; Последние 8 байтов
      mov DX,offset SC_SD_STR
      mov AH,09h
      int 21h
      mov BX,0
GO:
      mov DL,ES:[BX + 08h]
      mov AH,02h
      int 21h
      inc BX
      cmp BX,8
            jl GO
      mov DX,offset END_PRINT
      mov AH,09h
      int 21h
      mov  DX,offset END_LINE
      mov AH,09h
      int 21h
      mov AL,ES:[00h]
      cmp AL,5Ah
            je STOP
      xor CX,CX
      mov CX,ES:[03h]
      mov BX,ES
      add BX,CX
      inc BX
      mov ES,BX
      jmp MCB
STOP:
      ret
```

```
MAIN ENDP

call MAIN
      ; Выход в DOS
      xor AL,AL
      mov AH,4Ch
      int 21H
END_FREE:
      TESTPC ENDS
      END START
```

ПРИЛОЖЕНИЕ С

Исходный код программы

lr3_2.asm

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

; Данные
AVAILABLE_MEM_STR db 'Available memory        bytes', 0DH,0AH, '$'
EXTENDED_MEM_STR db 'Extended memory       bytes', 0DH,0AH, '$'
MCB_STR db 'MCB ', 0DH,0AH,'$'
TYPE_STR db 'Type $'
OWNER_ADRESS_STR db ' | PSP segment     $'
SIZE_STR db ' | Size        $'
SC_SD_STR db '| SC/SD $'
END_PRINT db 0DH,0AH,'$'
END_LINE                                                                db
'-------------------------------------------------------------------------
----------$'

; 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 шестн. числа в AX
      push CX
      mov AH,AL
      call TETR_TO_HEX
      xchg AL,AH
      mov CL,4
      shr AL,CL
      call TETR_TO_HEX ;в AL старшая цифра
      pop CX         ;в AH младшая
      ret
BYTE_TO_HEX ENDP
;--------------------------------
```

```
WRD_TO_HEX PROC near
; перевод в 16 с/с 16-ти разрядного числа
; в AX - 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_l
            or AL,30h
            mov [SI],AL
      end_l:
            pop DX
            pop CX
            ret
BYTE_TO_DEC ENDP
;--------------------------------

SIZE_MEM PROC
      xor CX,CX
```

```
DIVISION:
      div BX
      push DX
      inc CX
      xor DX,DX
      cmp AX,0
            jz SYMBOL
      jmp DIVISION
SYMBOL:
      pop DX
      add DL,30h
      mov [DI],DL
      inc DI
      loop SYMBOL
      ret
SIZE_MEM ENDP

; Код
BEGIN:

MAIN PROC near
; Количество доступной памяти
      mov AH,4ah
      mov BX,0FFFFh
      int 21h
      mov AX,BX
      mov BX,16
      mul BX
      lea DI,AVAILABLE_MEM_STR + 17
      mov BX,10
      call SIZE_MEM
      mov DX,offset AVAILABLE_MEM_STR
      mov AH,09h
      int 21h
      mov DX, offset END_PRINT
      mov AH,09h
      int 21h

; Количество расширенной памяти
      mov AL,30h
      out 70h,AL
      in AL,71h
      mov BL,AL
      mov AL,31h
      out 70h,AL
      in AL,71h
      mov BH,AL
      mov AX,BX
      lea DI,EXTENDED_MEM_STR + 16
```

```
      mov DX,0
      mov BX,10
      call SIZE_MEM
      mov DX,offset EXTENDED_MEM_STR
      mov AH,09h
      int 21h
      mov DX,offset END_PRINT
      mov AH,09h
      int 21h

; Освобождение не занятой памяти
      mov AX,offset END_FREE
      mov AH,4Ah
      int 21h

; Выделение памяти
      mov BX, 1000h
      mov AH, 48h
      int 21h

; Цепочка блоков управления памятью
      mov AH,52h
      int 21h
      mov CX,ES:[BX - 2]
      mov ES,CX
      mov DX,offset MCB_STR
      mov AH,09h
      int 21h

MCB:
; Тип MCB
      mov DX,offset TYPE_STR
      mov AH,09h
      int 21h
      mov AL, ES:[00h]
      push AX
      push DX
      push CX
      call BYTE_TO_HEX
      xor CX,CX
      mov CH,AH
      mov DL,AL
      mov AH,02h
      int 21h
      mov DL,CH
      mov AH,02h
      int 21h
      pop CX
      pop DX
```

```
      pop AX

; Адрес владельца
      mov AX,ES:[01h]
      lea DI,OWNER_ADRESS_STR + 18
      call WRD_TO_HEX
      mov DX,offset OWNER_ADRESS_STR
      mov AH,09h
      int 21h

; Размер
      mov AX,ES:[03h]
      lea DI,SIZE_STR + 8
      mov BX,16
      mul BX
      mov BX,10
      call SIZE_MEM
      mov DX,offset SIZE_STR
      mov AH,09h
      int 21h

; Последние 8 байтов
      mov DX,offset SC_SD_STR
      mov AH,09h
      int 21h
      mov BX,0
GO:
      mov DL,ES:[BX + 08h]
      mov AH,02h
      int 21h
      inc BX
      cmp BX,8
            jl GO
      mov DX,offset END_PRINT
      mov AH,09h
      int 21h
      mov  DX,offset END_LINE
      mov AH,09h
      int 21h
      mov AL,ES:[00h]
      cmp AL,5Ah
            je STOP
      xor CX,CX
      mov CX,ES:[03h]
      mov BX,ES
      add BX,CX
      inc BX
      mov ES,BX
      jmp MCB
```

```
STOP:
      ret



MAIN ENDP

call MAIN
      ; Выход в DOS
      xor AL,AL
      mov AH,4Ch
      int 21H
END_FREE:
      TESTPC ENDS
      END START
```

ПРИЛОЖЕНИЕ D

Исходный код программы

lr3_2.asm

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

; Данные
AVAILABLE_MEM_STR db 'Available memory        bytes', 0DH,0AH, '$'
EXTENDED_MEM_STR db 'Extended memory       bytes', 0DH,0AH, '$'
MCB_STR db 'MCB ', 0DH,0AH,'$'
TYPE_STR db 'Type $'
OWNER_ADRESS_STR db ' | PSP segment     $'
SIZE_STR db ' | Size        $'
SC_SD_STR db '| SC/SD $'
END_PRINT db 0DH,0AH,'$'
END_LINE                                                                db
'-------------------------------------------------------------------------
----------$'
ERR_STR db 'Memory allocation error', 0DH,0AH,'$'
SUCCESS_STR db 'Success', 0DH,0AH,'$'

; 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 шестн. числа в AX
      push CX
      mov AH,AL
      call TETR_TO_HEX
      xchg AL,AH
      mov CL,4
      shr AL,CL
      call TETR_TO_HEX ;в AL старшая цифра
      pop CX         ;в AH младшая
      ret
```

```
BYTE_TO_HEX ENDP
;-------------------------------

WRD_TO_HEX PROC near
; перевод в 16 с/с 16-ти разрядного числа
; в AX - 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_l
            or AL,30h
            mov [SI],AL
      end_l:
            pop DX
            pop CX
            ret
BYTE_TO_DEC ENDP
;-------------------------------
```

```
SIZE_MEM PROC
      xor CX,CX
DIVISION:
      div BX
      push DX
      inc CX
      xor DX,DX
      cmp AX,0
            jz SYMBOL
      jmp DIVISION
SYMBOL:
      pop DX
      add DL,30h
      mov [DI],DL
      inc DI
      loop SYMBOL
      ret
SIZE_MEM ENDP

; Код
BEGIN:

MAIN PROC near
; Количество доступной памяти
      mov AH,4ah
      mov BX,0FFFFh
      int 21h
      mov AX,BX
      mov BX,16
      mul BX
      lea DI,AVAILABLE_MEM_STR + 17
      mov BX,10
      call SIZE_MEM
      mov DX,offset AVAILABLE_MEM_STR
      mov AH,09h
      int 21h
      mov DX, offset END_PRINT
      mov AH,09h
      int 21h

; Количество расширенной памяти
      mov AL,30h
      out 70h,AL
      in AL,71h
      mov BL,AL
      mov AL,31h
      out 70h,AL
      in AL,71h
      mov BH,AL
```

```
      mov AX,BX
      lea DI,EXTENDED_MEM_STR + 16
      mov DX,0
      mov BX,10
      call SIZE_MEM
      mov DX,offset EXTENDED_MEM_STR
      mov AH,09h
      int 21h
      mov DX,offset END_PRINT
      mov AH,09h
      int 21h

; Выделение памяти
      mov BX, 1000h
      mov AH, 48h
      int 21h

; Проверка CF
      jc CF
SUCCESS:
      mov DX,offset SUCCESS_STR
      mov AH,09h
      int 21h
      mov DX,offset END_PRINT
      mov AH,09h
      int 21h
      jmp CONTINUE
CF:
      mov DX,offset ERR_STR
      mov AH,09h
      int 21h
      mov DX,offset END_PRINT
      mov AH,09h
      int 21h

CONTINUE:
; Освобождение не занятой памяти
      mov AX,offset END_FREE
      mov AH,4Ah
      int 21h

; Цепочка блоков управления памятью
      mov AH,52h
      int 21h
      mov CX,ES:[BX - 2]
      mov ES,CX
      mov DX,offset MCB_STR
      mov AH,09h
      int 21h
```

```
MCB:
; Тип MCB
      mov DX,offset TYPE_STR
      mov AH,09h
      int 21h
      mov AL, ES:[00h]
      push AX
      push DX
      push CX
      call BYTE_TO_HEX
      xor CX,CX
      mov CH,AH
      mov DL,AL
      mov AH,02h
      int 21h
      mov DL,CH
      mov AH,02h
      int 21h
      pop CX
      pop DX
      pop AX

; Адрес владельца
      mov AX,ES:[01h]
      lea DI,OWNER_ADRESS_STR + 18
      call WRD_TO_HEX
      mov DX,offset OWNER_ADRESS_STR
      mov AH,09h
      int 21h

; Размер
      mov AX,ES:[03h]
      lea DI,SIZE_STR + 8
      mov BX,16
      mul BX
      mov BX,10
      call SIZE_MEM
      mov DX,offset SIZE_STR
      mov AH,09h
      int 21h

; Последние 8 байтов
      mov DX,offset SC_SD_STR
      mov AH,09h
      int 21h
      mov BX,0
GO:
      mov DL,ES:[BX + 08h]
```

```
      mov AH,02h
      int 21h
      inc BX
      cmp BX,8
            jl GO
      mov DX,offset END_PRINT
      mov AH,09h
      int 21h
      mov  DX,offset END_LINE
      mov AH,09h
      int 21h
      mov AL,ES:[00h]
      cmp AL,5Ah
            je STOP
      xor CX,CX
      mov CX,ES:[03h]
      mov BX,ES
      add BX,CX
      inc BX
      mov ES,BX
      jmp MCB
STOP:
      ret



MAIN ENDP

call MAIN
      ; Выход в DOS
      xor AL,AL
      mov AH,4Ch
      int 21H
END_FREE:
      TESTPC ENDS
      END START
```