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

Студентка гр. 7383 Маркова А.В.

Преподаватель Ефремов М.А.

Санкт-Петербург

2019

Постановка задачи.

Исследование структур данных и работы функций управления памятью операционной системы.

Описание процедур, используемых в работе:

- TETR_TO_HEX – вспомогательная функция, которая переводит из двоичной в шестнадцатеричную систему счисления, используется для работы функции BYTE_TO_HEX;
- BYTE_TO_HEX – переводит байтовое число из регистра AL в шестнадцатеричную систему счисления;
- WRD_TO_HEX – переводит число из регистра AX в шестнадцатеричную систему;
- BYTE_TO_DEC – переводит байтовое число в десятичную систему счисления;
- LINE_OUTPUT – выводит сообщение на экран;
- AVAILABLE_MEM – выведение на экран информации о количестве доступной памяти;
- FREE_MEMORY – освобождение памяти, которую программа не занимает;
- MEMORY_REQUEST – запрашивание программой памяти;
- EXTENDED_MEMORY – выведение на экран информации о размере расширенной памяти;
- CHAIN_OF_MCB – выведение на экран цепочки блоков управления памятью.

Таблица 1 – Описание структур данных:

Название	**Тип**	**Назначение**
extended_memory_size	db	Размер расширенной памяти
available_memory	db	Количество доступной памяти
data_of_mcb	db	Данные о MCB
message	db	Сообщение об ошибки
endl	db	Новая строка
mcb	db	Обозначения

Ход работы.

1. Был написан код программы.
2. Смонтирован виртуальный диск k с каталогом tasm.
3. Было произведено транслирование программы с помощью строки tasm «имя файла», в последствии чего создался объектный файл, результат вызова команды показан на рис. 1.

Рисунок 1 – Транслирование программы

4. Линковка загрузочного модуля с помощью команды tlink/t.
5. Получен LR3_1.COM модуль из исходного кода.
6. Результаты работы программы показаны на рис. 2 – 5.

Рисунок 2 – Результат выполнения программы

Все доступные 648912 байт отдаются программе.

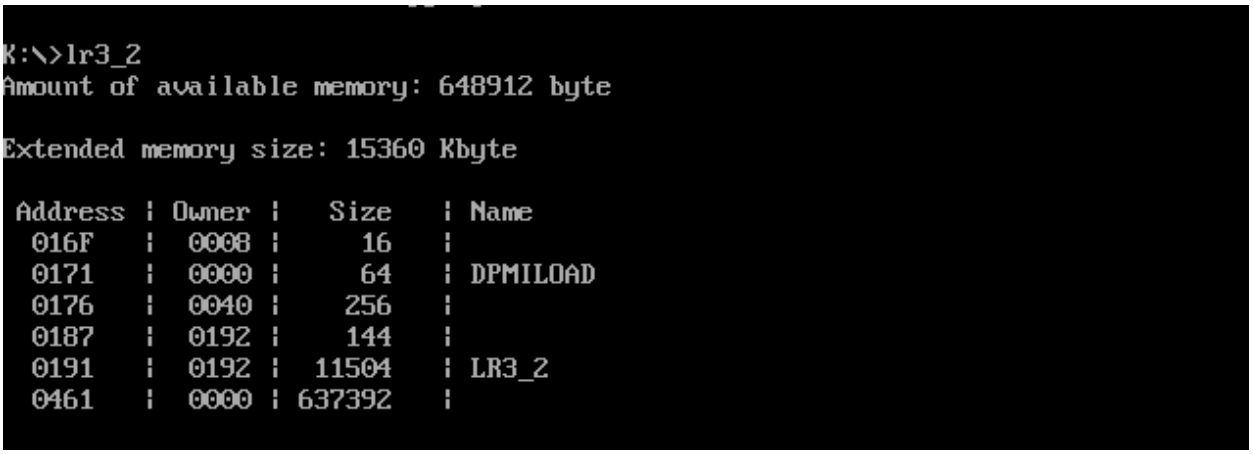

Рисунок 3 – Результат выполнения программы с первой модификацией

Код исходного модуля, созданный вначале, был изменён. Добавлена процедура освобождения памяти, которую не занимает программа. После выполнения данной функции, появляется новый блок свободной памяти (0000h) размера 637392 байт.

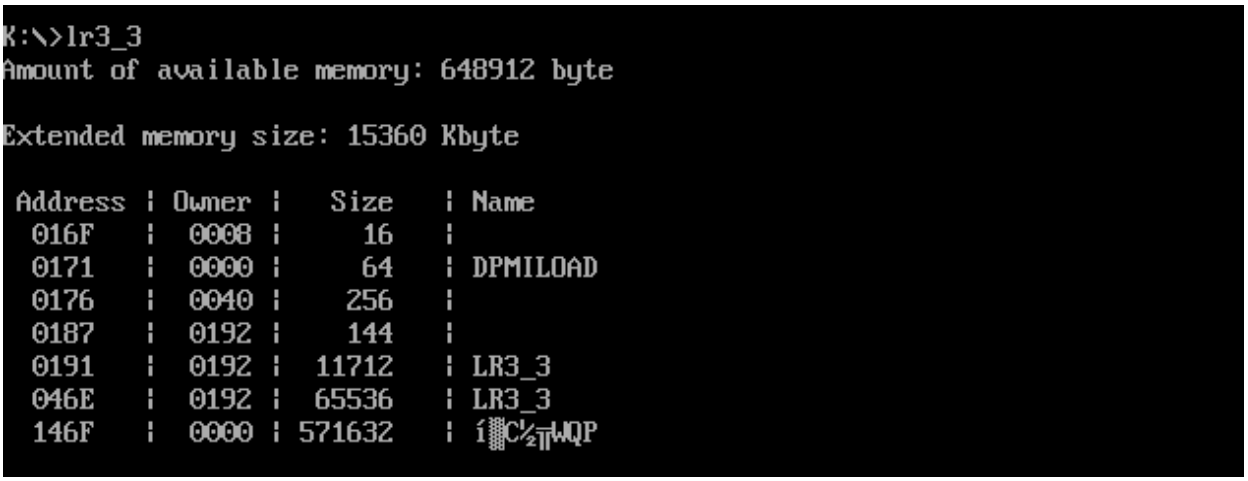

Рисунок 4 – Результат выполнения программы со второй модификацией

В программе после освобождения памяти запускается процедура запрашивания 64 Кб памяти. Из свободного участка создаётся новый блок размером 65536 байта (64 Кб), и следует за блоком программы.

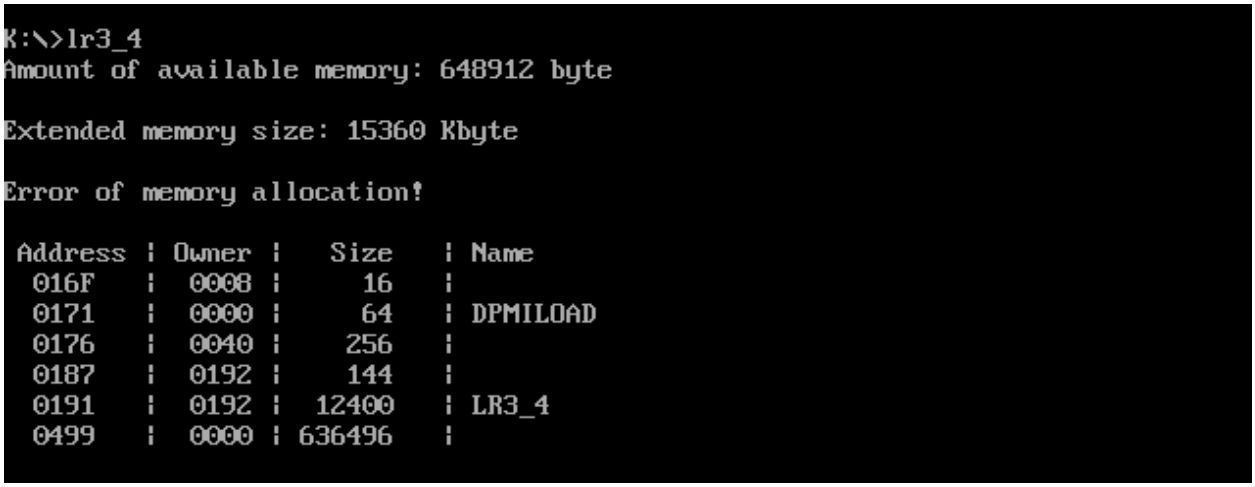

Рисунок 5 – Результат выполнения программы с третьей модификацией

В программе с третьей модификацией были поменяны местами процедуры выделения и освобождения памяти, а также добавлена обработка завершения функций ядра. При попытке выделить 64 Кб памяти появляется ошибка, из – за того, что вся доступная память занята программой. На экран выводится сообщение об ошибки, затем происходит освобождение памяти.

Выводы.

В ходе данной лабораторной работы были исследованы структуры данных и работа функций управления памятью ядра операционной системы. Был написан текст исходного .COM файла, который выводит на экран количество доступной памяти, размер расширенной памяти и цепочку блоков управления памяти. Было замечено, что изначально вся доступная память отдаётся под управление программы, но при необходимости при помощи функции 4Ah можно освободить неиспользуемую программой память. Также при помощи функции 48h можно дополнительно выделить память. Код программы представлен в приложении А.

Ответы на контрольные вопросы.

1. Что означает «доступный объём памяти»?

Доступный объём памяти – максимальный объём памяти, в который можно загружать пользовательские программы.

2. Где MCB блок вашей программы в списке?

В первой, второй и четвёртой программах MCB блок имеет адрес 0191h, который является программным блоком и начинается с PSP. В третьей программе MCB блок имеет адрес 0191h, выделенная память в 64Кб тоже относится к блоку управления памяти – 046Eh. Также в каждой из программ присутствует ещё один блок MCB, имеющий адрес 0187h и размер 144 байт. Это блок управления памятью для области среды программы.

3. Какой размер памяти занимает программа в каждом случае?

В первом случае (lr3_1.com) программа занимает всю выделенную память: 648912 байт.

Во втором случае (lr3_2.com) программа занимает столько, сколько ей необходимо: (648912 – 637392 – 16) = 11504 байт.

В третьем случае (lr3_3.com) программа занимает свой размер и объём выделенной памяти: (648912 – 571632 – 65536 – 2*16) = 11712 байт.

В четвертом случае (lr3_4.com) сначала запросили выделить память, но процедура не была выполнена, а затем освободили неиспользуемую память: (648912 – 636496 – 16) = 12400 байт.

ПРИЛОЖЕНИЕ А

LR3_1.ASM

```
EOFLine EQU '$'                                         ; определение символьной константы
                                                        ; $ - "конец строки"
TESTPC SEGMENT                                          ; определение начала сегмента
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H                                            ; смещение
START: JMP BEGIN                                        ; переход на метку
;ДАННЫЕ
;-------------------------------------------------------------------------------------
extended_memory_size db 'Extended memory size:       Kbyte',       0DH,0AH,EOFLine
available_memory     db 'Amount of available memory:        byte', 0DH,0AH,EOFLine
data_of_mcb          db '         |       |          | ',         EOFLine
endl                 db ' ',0DH,0AH,                               EOFLine
mcb                  db ' Address | Owner |   Size   | Name     ', 0DH,0AH,EOFLine
;ПРОЦЕДУРЫ
;-------------------------------------------------------------------------------------
TETR_TO_HEX PROC near                                   ; из двоичной в шестнадцатеричную сс
    and  AL, 0Fh                                        ; PROC near - вызывается в том же сегменте, в котором определена
    cmp  AL, 09
    jbe  NEXT
    add  AL, 07
NEXT:    add AL, 30h
    ret
TETR_TO_HEX ENDP

BYTE_TO_HEX PROC near                                   ; байтовое число в шестнадцатеричную сс
    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                                    ; шестнадцатибитовое число в шестнадцатеричную сс
    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                                   ; байтовое число в десятичную сс
    push CX
    push 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
;--------------------------------------------------------------------------------------
LINE_OUTPUT PROC near                                   ; вывод строки
    push AX
    mov  AH, 09h
    int  21H
    pop  AX
    ret
LINE_OUTPUT ENDP

AVAILABLE_MEM PROC near                                 ;вывод информации о кол-ве доступной памяти
    mov  AH, 4Ah                                        ;расширение блока памяти
    mov  BX, 0ffffh                                     ;заведомо большое число => расширение неудачно
    int  21h                                            ;запуск функции 4Ah прерывания int 21h
    mov  AX, BX                                         ;в BX - записан наибольший доступный блок
    mov  BX, 0010h                                      ;умножаем на 16, чтоб получить результат в байтах
    mul  BX                                             ;dx:ax = ax*bx ,кол-во параграфов * 16 байт
    mov  SI, offset available_memory                    ;в результате получаем большое число, которое хранится в двух регистрах
    add  SI, 33
    call BYTE_TO_DEC
    mov  DX, offset available_memory
    call LINE_OUTPUT
    mov  dx, offset endl
    call LINE_OUTPUT
    ret
AVAILABLE_MEM ENDP

EXTENDED_MEMORY PROC near                               ;вывод информации о размере расширенной памяти
    mov  AL,  30h                                       ;запись адреса ячейки CMOS
    out  70h, AL                                        ;вывод значения из al в порт 70h
    in   AL,  71h                                       ;получение в al значение из 71h (младший байт)
    mov  BL,  AL                                        ;перенос в bl
    mov  AL,  31h                                       ;запись вдреса ячейки CMOS
    out  70h, AL
    in   AL,  71h                                       ;получение старшего байта
    mov  BH,  AL
    mov  AX,  BX
    xor  DX,  DX                                        ;чтобы при выводе числа из dx и ax, не было лишнего
    mov  SI,  offset extended_memory_size
    add  SI,  26
    call BYTE_TO_DEC
    mov  DX,  offset extended_memory_size
    call LINE OUTPUT
```

```
    ret
EXTENDED_MEMORY ENDP

CHAIN_OF_MCB PROC
    mov  dx, offset mcb
    call LINE_OUTPUT
    mov  AH, 52h                                    ;функция, которая в es:bx возвращает list of lists
    int  21h                                        ;вызов функции
    mov  BX, ES:[BX-2]                              ;получение адреса первого MCB блока
    mov  ES, BX

    print_MCB:
        mov  AX, ES
        mov  DI, offset data_of_mcb                 ;заполнение адреса MCB блока
        add  DI, 5
        call WRD_TO_HEX
        mov  AX, ES:[0001h]                         ;получение сегментного адреса PSP владельца
        mov  DI, offset data_of_mcb
        add  DI, 15
        call WRD_TO_HEX
        mov  AX, ES:[0003h]                         ;получение размера участка в параграфах
        mov  SI, offset data_of_mcb
        add  SI, 24
        xor  DX, DX
        mov  BX, 0010h
        mul  BX                                     ;перевод в байты
        call BYTE_TO_DEC
        mov  DX, offset data_of_mcb
        call LINE_OUTPUT
        push BX
        mov  CX, 0008h
        mov  BX, 0008h                              ;для вывода последних 8 байт
        mov  AH, 0002h

        print:
            mov DL, byte ptr ES:[BX]
            inc BX
            int 21h
        loop print

        pop  BX
        mov  DX, offset endl
        call LINE_OUTPUT
        mov  AX, ES
        inc  AX
        add  AX, ES:[0003h]
        mov  BL, ES:[0000h]
        mov  ES, AX
        cmp  BL, 4Dh
        je   print_MCB

    ret
CHAIN_OF_MCB ENDP
;-----------------------------------------------------------------------------------------
BEGIN:

    call AVAILABLE_MEM
    call EXTENDED_MEMORY
    call CHAIN_OF_MCB
    xor  AL, AL
    mov  AH, 4Ch
    int  21h
    TESTPC   ENDS
        END START                                   ;конец модуля. START - точка входа
```

LR3_2.ASM

```
EOFLine EQU '$'                                         ; определение символьной константы
                                                        ; $ - "конец строки"
TESTPC SEGMENT                                          ; определение начала сегмента
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H                                            ; смещение
START: JMP BEGIN                                        ; переход на метку
;ДАННЫЕ
;-------------------------------------------------------------------------------------
extended_memory_size db 'Extended memory size:       Kbyte',       0DH,0AH,EOFLine
available_memory     db 'Amount of available memory:        byte', 0DH,0AH,EOFLine
data_of_mcb          db '          |       |          | ',         EOFLine
endl                 db ' ',0DH,0AH,                               EOFLine
mcb                  db ' Address | Owner |   Size   | Name     ', 0DH,0AH,EOFLine
;ПРОЦЕДУРЫ
;-------------------------------------------------------------------------------------
TETR_TO_HEX PROC near                                   ; из двоичной в шестнадцатеричную сс
    and  AL, 0Fh                                        ; PROC near - вызывается в том же сегменте, в котором определена
    cmp  AL, 09
    jbe  NEXT
    add  AL, 07
NEXT:    add AL, 30h
    ret
TETR_TO_HEX ENDP

BYTE_TO_HEX PROC near                                   ; байтовое число в шестнадцатеричную сс
    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                                    ; шестнадцатибитовое число в шестнадцатеричную сс
    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                                   ; байтовое число в десятичную сс
    push CX
    push 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
;--------------------------------------------------------------------------------
LINE_OUTPUT PROC near                                   ; вывод строки
    push AX
    mov  AH, 09h
    int  21H
    pop  AX
    ret
LINE_OUTPUT ENDP

AVAILABLE_MEM PROC near                                 ;вывод информации о кол-ве доступной памяти
    mov  AH, 4Ah                                        ;расширение блока памяти
    mov  BX, 0ffffh                                     ;заведомо большое число => расширение неудачно
    int  21h                                            ;запуск функции 4Ah прерывания int 21h
    mov  AX, BX                                         ;в BX - записан наибольший доступный блок
    mov  BX, 0010h                                      ;умножаем на 16, чтоб получить результат в байтах
    mul  BX                                             ;dx:ax = ax*bx ,кол-во параграфов * 16 байт
    mov  SI, offset available_memory                    ;в результате получаем большое число, которое хранится в двух регистрах
    add  SI, 33
    call BYTE_TO_DEC
    mov  DX, offset available_memory
    call LINE_OUTPUT
    mov  dx, offset endl
    call LINE_OUTPUT
    ret
AVAILABLE_MEM ENDP

FREE_MEMORY PROC near
    mov AH, 4Ah                                         ;функция 4Ah прерывания 21h для освобождения памяти
    mov BX, offset end_programs                         ;смещение на конец программы - размер памяти программы
    int 21h
    ret
FREE_MEMORY ENDP

EXTENDED_MEMORY PROC near                               ;вывод информации о размере расширенной памяти
    mov  AL,  30h                                       ;запись адреса ячейки CMOS
    out  70h, AL                                        ;вывод значения из al в порт 70h
    in   AL,  71h                                       ;получение в al значение из 71h (младший байт)
    mov  BL,  AL                                        ;перенос в bl
    mov  AL,  31h                                       ;запись вдреса ячейки CMOS
    out  70h, AL
    in   AL,  71h                                       ;получение старшего байта
    mov  BH,  AL
```

```
    mov  AX,  BX
    xor  DX,  DX                                    ;чтобы при выводе числа из dx и ax, не было лишнего
    mov  SI,  offset extended_memory_size
    add  SI,  26
    call BYTE_TO_DEC
    mov  DX,  offset extended_memory_size
    call LINE_OUTPUT
    mov  dx, offset endl
    call LINE_OUTPUT
    ret
EXTENDED_MEMORY ENDP

CHAIN_OF_MCB PROC
    mov  dx, offset mcb
    call LINE_OUTPUT
    mov  AH, 52h                                    ;функция, которая в es:bx возвращает list of lists
    int  21h                                        ;вызов функции
    mov  BX, ES:[BX-2]                              ;получение адреса первого MCB блока
    mov  ES, BX

    print_MCB:
        mov  AX, ES
        mov  DI, offset data_of_mcb                 ;заполнение адреса MCB блока
        add  DI, 5
        call WRD_TO_HEX
        mov  AX, ES:[0001h]                         ;получение сегментного адреса PSP владельца
        mov  DI, offset data_of_mcb
        add  DI, 15
        call WRD_TO_HEX
        mov  AX, ES:[0003h]                         ;получение размера участка в параграфах
        mov  SI, offset data_of_mcb
        add  SI, 24
        xor  DX, DX
        mov  BX, 0010h
        mul  BX                                     ;перевод в байты
        call BYTE_TO_DEC
        mov  DX, offset data_of_mcb
        call LINE_OUTPUT
        push BX
        mov  CX, 0008h
        mov  BX, 0008h                              ;для вывода последних 8 байт
        mov  AH, 0002h

        print:
            mov DL, byte ptr ES:[BX]
            inc BX
            int 21h
        loop print

        pop  BX
        mov  DX, offset endl
        call LINE_OUTPUT
        mov  AX, ES
        inc  AX
        add  AX, ES:[0003h]
        mov  BL, ES:[0000h]
        mov  ES, AX

        cmp  BL, 4Dh
        je   print_MCB

    ret
CHAIN_OF_MCB ENDP
;-------------------------------------------------------------------------------------------
BEGIN:
    call AVAILABLE_MEM
    call EXTENDED_MEMORY
    call FREE_MEMORY
    call CHAIN_OF_MCB
    xor  AL, AL
    mov  AH, 4Ch
    int  21h
    end_programs:
    TESTPC   ENDS
        END START                                   ;конец модуля, START - точка входа
```

LR3_3.ASM

```
EOFLine EQU '$'                                                 ; определение символьной константы
                                                                ; $ - "конец строки"
TESTPC SEGMENT                                                  ; определение начала сегмента
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H                                                    ; смещение
START: JMP BEGIN                                                ; переход на метку
;ДАННЫЕ
;-------------------------------------------------------------------------------------
extended_memory_size db 'Extended memory size:       Kbyte',       0DH,0AH,EOFLine
available_memory     db 'Amount of available memory:        byte', 0DH,0AH,EOFLine
data_of_mcb          db '         |       |          | ',          EOFLine
endl                 db ' ',0DH,0AH,                               EOFLine
mcb                  db ' Address | Owner |   Size   | Name     ', 0DH,0AH,EOFLine
;ПРОЦЕДУРЫ
;-------------------------------------------------------------------------------------
TETR_TO_HEX PROC near                                           ; из двоичной в шестнадцатеричную сс
    and  AL, 0Fh                                                ; PROC near - вызывается в том же сегменте, в котором определена
    cmp  AL, 09
    jbe  NEXT
    add  AL, 07
NEXT:    add AL, 30h
    ret
TETR_TO_HEX ENDP

BYTE_TO_HEX PROC near                                           ; байтовое число в шестнадцатеричную сс
    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                                            ; шестнадцатибитовое число в шестнадцатеричную сс
    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                                           ; байтовое число в десятичную сс
    push CX
    push 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
;--------------------------------------------------------------------------------------
LINE_OUTPUT PROC near                                   ; вывод строки
    push AX
    mov  AH, 09h
    int  21H
    pop  AX
    ret
LINE_OUTPUT ENDP

AVAILABLE_MEM PROC near                                 ;вывод информации о кол-ве доступной памяти
    mov  AH, 4Ah                                        ;расширение блока памяти
    mov  BX, 0ffffh                                     ;заведомо большое число => расширение неудачно
    int  21h                                            ;запуск функции 4Ah прерывания int 21h
    mov  AX, BX                                         ;в BX - записан наибольший доступный блок
    mov  BX, 0010h                                      ;умножаем на 16, чтоб получить результат в байтах
    mul  BX                                             ;dx:ax = ax*bx ,кол-во параграфов * 16 байт
    mov  SI, offset available_memory                    ;в результате получаем большое число, которое хранится в двух регистрах
    add  SI, 33
    call BYTE_TO_DEC
    mov  DX, offset available_memory
    call LINE_OUTPUT
    mov  dx, offset endl
    call LINE_OUTPUT
    ret
AVAILABLE_MEM ENDP

FREE_MEMORY PROC near
    mov AH, 4Ah                                         ;функция 4Ah прерывания 21h для освобождения памяти
    mov BX, offset end_programs                         ;смещение на конец программы - размер памяти программы
    int 21h
    ret
FREE_MEMORY ENDP

MEMORY_REQUEST PROC near                                ;запрос 64Кб памяти
    mov AH, 48h                                         ;функция 48h прерывания int 21h для выделения памяти
    mov BX, 1000h                                       ;запрошенное количество памяти в 16-байтовых параграфах 1000 = 64Кб
    int 21h
    ret
MEMORY_REQUEST ENDP

EXTENDED_MEMORY PROC near                               ;вывод информации о размере расширенной памяти
    mov  AL,  30h                                       ;запись адреса ячейки CMOS
```

```
    out  70h, AL                                    ;вывод значения из al в порт 70h
    in   AL,  71h                                   ;получение в al значение из 71h (младший байт)
    mov  BL,  AL                                    ;перенос в bl
    mov  AL,  31h                                   ;запись вдреса ячейки CMOS
    out  70h, AL
    in   AL,  71h                                   ;получение старшего байта
    mov  BH,  AL
    mov  AX,  BX
    xor  DX,  DX                                    ;чтобы при выводе числа из dx и ax, не было лишнего
    mov  SI,  offset extended_memory_size
    add  SI,  26
    call BYTE_TO_DEC
    mov  DX,  offset extended_memory_size
    call LINE_OUTPUT
    mov  dx, offset endl
    call LINE_OUTPUT
    ret
EXTENDED_MEMORY ENDP

CHAIN_OF_MCB PROC
    mov  dx, offset mcb
    call LINE_OUTPUT
    mov  AH, 52h                                    ;функция, которая в es:bx возвращает list of lists
    int  21h                                        ;вызов функции
    mov  BX, ES:[BX-2]                              ;получение адреса первого MCB блока
    mov  ES, BX

    print_MCB:
        mov  AX, ES
        mov  DI, offset data_of_mcb                 ;заполнение адреса MCB блока
        add  DI, 5
        call WRD_TO_HEX
        mov  AX, ES:[0001h]                         ;получение сегментного адреса PSP владельца
        mov  DI, offset data_of_mcb
        add  DI, 15
        call WRD_TO_HEX
        mov  AX, ES:[0003h]                         ;получение размера участка в параграфах
        mov  SI, offset data_of_mcb
        add  SI, 24
        xor  DX, DX
        mov  BX, 0010h
        mul  BX                                     ;перевод в байты
        call BYTE_TO_DEC
        mov  DX, offset data_of_mcb
        call LINE_OUTPUT
        push BX
        mov  CX, 0008h
        mov  BX, 0008h                              ;для вывода последних 8 байт
        mov  AH, 0002h

        print:
            mov DL, byte ptr ES:[BX]
            inc BX
            int 21h
        loop print

        pop  BX
        mov  DX, offset endl
        call LINE_OUTPUT
        mov  AX, ES
        inc  AX
        add  AX, ES:[0003h]
        mov  BL, ES:[0000h]
        mov  ES, AX
        cmp  BL, 4Dh
        je   print_MCB

    ret
CHAIN_OF_MCB ENDP
;-----------------------------------------------------------------------------------------
BEGIN:
    call AVAILABLE_MEM
    call EXTENDED_MEMORY
    call FREE_MEMORY
    call MEMORY_REQUEST
    call CHAIN_OF_MCB
    xor  AL, AL
    mov  AH, 4Ch
    int  21h
    end_programs:
    TESTPC   ENDS
        END START                                   ;конец модуля, START - точка входа
```

LR3_4.ASM

```
EOFLine EQU '$'                                                 ; определение символьной константы
                                                                ; $ - "конец строки"
TESTPC SEGMENT                                                  ; определение начала сегмента
       ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
       ORG 100H                                                 ; смещение
START: JMP BEGIN                                                ; переход на метку
;ДАННЫЕ
;-------------------------------------------------------------------------------------------
extended_memory_size db 'Extended memory size:       Kbyte',       0DH,0AH,EOFLine
available_memory     db 'Amount of available memory:        byte', 0DH,0AH,EOFLine
data_of_mcb          db '          |       |          | ',          EOFLine
message              db 'Error of memory allocation!',             0DH,0AH,EOFLine
endl                 db ' ',0DH,0AH,                               EOFLine
mcb                  db ' Address | Owner |   Size   | Name     ', 0DH,0AH,EOFLine
;ПРОЦЕДУРЫ
;-------------------------------------------------------------------------------------------
TETR_TO_HEX PROC near                                           ; из двоичной в шестнадцатеричную сс
    and  AL, 0Fh                                                ; PROC near - вызывается в том же сегменте, в котором определена
    cmp  AL, 09
    jbe  NEXT
    add  AL, 07
NEXT:    add AL, 30h
    ret
TETR_TO_HEX ENDP

BYTE_TO_HEX PROC near                                           ; байтовое число в шестнадцатеричную сс
    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                                            ; шестнадцатибитовое число в шестнадцатеричную сс
    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                                           ; байтовое число в десятичную сс
    push CX
    push 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
;--------------------------------------------------------------------------------
LINE_OUTPUT PROC near                                   ; вывод строки
    push AX
    mov  AH, 09h
    int  21H
    pop  AX
    ret
LINE_OUTPUT ENDP

AVAILABLE_MEM PROC near                                 ;вывод информации о кол-ве доступной памяти
    mov  AH, 4Ah                                        ;расширение блока памяти
    mov  BX, 0ffffh                                     ;заведомо большое число => расширение неудачно
    int  21h                                            ;запуск функции 4Ah прерывания int 21h
    mov  AX, BX                                         ;в BX - записан наибольший доступный блок
    mov  BX, 0010h                                      ;умножаем на 16, чтоб получить результат в байтах
    mul  BX                                             ;dx:ax = ax*bx ,кол-во параграфов * 16 байт
    mov  SI, offset available_memory                    ;в результате получаем большое число, которое хранится в двух регистрах
    add  SI, 33
    call BYTE_TO_DEC
    mov  DX, offset available_memory
    call LINE_OUTPUT
    mov  dx, offset endl
    call LINE_OUTPUT
    ret
AVAILABLE_MEM ENDP

FREE_MEMORY PROC near
    mov AH, 4Ah                                         ;функция 4Ah прерывания 21h для освобождения памяти
    mov BX, offset end_programs                         ;смещение на конец программы - размер памяти программы
    int 21h
    ret
FREE_MEMORY ENDP

MEMORY_REQUEST PROC near                                ;запрос 64Кб памяти
    mov AH, 48h                                         ;функция 48h прерывания int 21h для выделения памяти
    mov BX, 1000h                                       ;запрошенное количество памяти в 16-байтовых параграфах 1000 = 64Кб
    int 21h
    jc error_memory                                     ;если CF = 1, т.е возникла ошибка
    ret
    error_memory:
        mov  DX, offset message
        call LINE_OUTPUT
        mov  DX,offset endl
        call LINE_OUTPUT
    ret
MEMORY_REQUEST ENDP

EXTENDED_MEMORY PROC near                               ;вывод информации о размере расширенной памяти
    mov  AL,  30h                                       ;запись адреса ячейки CMOS
```

```
    out  70h, AL                                    ;вывод значения из al в порт 70h
    in   AL,  71h                                   ;получение в al значение из 71h (младший байт)
    mov  BL,  AL                                    ;перенос в bl
    mov  AL,  31h                                   ;запись вдреса ячейки CMOS
    out  70h, AL
    in   AL,  71h                                   ;получение старшего байта
    mov  BH,  AL
    mov  AX,  BX
    xor  DX,  DX                                    ;чтобы при выводе числа из dx и ax, не было лишнего
    mov  SI,  offset extended_memory_size
    add  SI,  26
    call BYTE_TO_DEC
    mov  DX,  offset extended_memory_size
    call LINE_OUTPUT
    mov  dx, offset endl
    call LINE_OUTPUT
    ret
EXTENDED_MEMORY ENDP

CHAIN_OF_MCB PROC
    mov  dx, offset mcb
    call LINE_OUTPUT
    mov  AH, 52h                                    ;функция, которая в es:bx возвращает list of lists
    int  21h                                        ;вызов функции
    mov  BX, ES:[BX-2]                              ;получение адреса первого MCB блока
    mov  ES, BX

    print_MCB:
        mov  AX, ES
        mov  DI, offset data_of_mcb                 ;заполнение адреса MCB блока
        add  DI, 5
        call WRD_TO_HEX
        mov  AX, ES:[0001h]                         ;получение сегментного адреса PSP владельца
        mov  DI, offset data_of_mcb
        add  DI, 15
        call WRD_TO_HEX
        mov  AX, ES:[0003h]                         ;получение размера участка в параграфах
        mov  SI, offset data_of_mcb
        add  SI, 24
        xor  DX, DX
        mov  BX, 0010h
        mul  BX                                     ;перевод в байты
        call BYTE_TO_DEC
        mov  DX, offset data_of_mcb
        call LINE_OUTPUT
        push BX
        mov  CX, 0008h
        mov  BX, 0008h                              ;для вывода последних 8 байт
        mov  AH, 0002h

        print:
            mov DL, byte ptr ES:[BX]
            inc BX
            int 21h
        loop print

        pop  BX
        mov  DX, offset endl
        call LINE_OUTPUT
        mov  AX, ES
        inc  AX
        add  AX, ES:[0003h]
        mov  BL, ES:[0000h]
        mov  ES, AX
        cmp  BL, 4Dh
        je   print_MCB

    ret
CHAIN_OF_MCB ENDP
;-----------------------------------------------------------------------------------------
BEGIN:
    call AVAILABLE_MEM
    call EXTENDED_MEMORY
    call MEMORY_REQUEST
    call FREE_MEMORY
    call CHAIN_OF_MCB
    xor  AL, AL
    mov  AH, 4Ch
    int  21h
    end_programs:
    TESTPC   ENDS
        END START                                   ;конец модуля, START - точка входа
```