

**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. 8383

\_\_\_\_\_

Максимова А.А.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

### Цель работы.

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

### Основные теоретические положения.

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

MCB имеет следующую структуру:

Смещение	Длина поля (байт)	Содержимое поля
00h	1	тип MCB: 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 FFFEh - участок принадлежит 386MAX UMB
03h	2	Размер участка в параграфах
05h	3	Зарезервирован
08h	8	"SC" - если участок принадлежит MS DOS, то в нем системный код "SD" - если участок принадлежит MS DOS, то в нем системные данные

Рисунок 1 - Структура MCB

Адрес первого MCB хранится во внутренней структуре MS DOS, называемой "List of Lists". Доступ к указателю на эту структуру можно получить, используя функцию 52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX-2] и есть адрес самого первого MCB.

Размер расширенной памяти находится в ячейках 30h, 31h CMOS.

## Процедуры, используемые в программе.

Таблица 1 - Процедуры

Название:	Предназначение:
TETR_TO_HEX	Процедура перевода тетрады в шестнадцатеричную цифру (символ)
BYTE_TO_HEX	Процедура перевода байта AL в два символа шестнадцатеричного числа
WRD_TO_HEX	Процедура перевода слова в шестнадцатеричную систему счисления
BYTE_TO_DEC	Процедура перевода байта в десятичную систему счисления.
GET_WRD_DEC	Процедура перевода слова в десятичную систему счисления.
PRINTF	Процедура печати
GET_A_MEM	Процедура, используемая для определения количества доступной памяти и вывода этого значения на экран.
GET_E_MEM	Процедура, используемая для определения размера расширенной памяти и вывода этого значения на экран.
START_	Процедура, используемая для вывода типа MCB и "разделения" блоков при выводе на экран.
GET_ADRESS	Процедура, необходимая для определения адреса MCB блока и вывода этого значения на экран.
GET_SIZE_MEM	Процедура, необходимая для определения размера участка в параграфах и вывода этого значения на экран.

GET_SYMBOLS	Процедура для получения и вывода на экран последних 8 байт блока MCB.
GET_MCB	Процедура для вызова START_, GET_ADRESS, GET_SIZE_MEM, GET_MCB для всех блоков MCB.
FREE_MEMORY	Процедура, используемая для освобождения памяти.
GIVE_ME_MEMORY	Процедура, используемая для запрашивания памяти.

### Выполнение работы.

Был написан и отлажен программный модуль типа **.COM**, который выбирает и распечатывает следующую информацию:

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

Код модуля LR3\_1.ASM см. в приложении А, тестирование модуля см. на рис. 2 и 3.

```

Z:\>C:

C:\>masm lr3_1.asm,,,;
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

47284 + 443591 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link lr3_1.obj,,,;
Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

LINK : warning L4021: no stack segment

C:\>exe2bin lr3_1.exe lr3_1.com

C:\>lr3_1.com>res3_1.txt

```

Рисунок 2 - Запуск LR3\_1.COM

```
RES3_1.TXT — Блокнот
Файл  Правка  Формат  Вид  Справка

Programm: first.
1. Amount of available memory:      648912 bytes.
2. Extended memory size:           15360  kbytes.
3. Memory management block chain:
-----
MCB TYPE: 4Dh
BELONGS MS DOS - address:           0008h
Memory size:      16  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
FREE AREA - address:                0000h
Memory size:      64  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:                0040h
Memory size:     256  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:                0192h
Memory size:     144  bytes.
Last eight bytes:
-----
MCB TYPE: 5Ah
Segment address PSP:                0192h
Memory size:  648912 bytes.
Last eight bytes:  LR3_1
```

Рисунок 3 - Тестирование LR3\_1.COM

Программа была изменена таким образом, чтобы она освобождала память, которую она не занимает. Для этого использовалась функция 4Ah прерывания 21H. Как видно из рисунка 5, программа занимает теперь лишь необходимое количество памяти, а освобожденная память хранится в новом блоке (последнем).

Код модуля LR3\_2.ASM см. в приложении Б, тестирование модуля см. на рис. 4 и 5.

```
C:\>masm lr3_2.asm,,,;
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

47284 + 443591 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link lr3_2.obj,,,;
Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

LINK : warning L4021: no stack segment

C:\>exe2bin lr3_2.exe lr3_2.com
C:\>lr3_2.com>res3_2.txt
```

Рисунок 4 - Запуск LR3\_2.COM

```
RES3_2.TXT — Блокнот
Файл  Правка  Формат  Вид  Справка

Programm: second.
1. Amount of available memory:      648912 bytes.
2. Free memory!
3. Extended memory size:           15360 kbytes.
4. Memory management block chain:
-----
MCB TYPE: 4Dh
BELONGS MS DOS - address:          0008h
Memory size:      16 bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
FREE AREA - address:              0000h
Memory size:      64 bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:              0040h
Memory size:     256 bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:              0192h
Memory size:     144 bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:              0192h
Memory size:    23344 bytes.
Last eight bytes: LR3_2
-----
MCB TYPE: 5Ah
FREE AREA - address:              0000h
Memory size:    625552 bytes.
Last eight bytes: A:AUИИP
```

Рисунок 5 - Тестирование LR3\_2.COM

Программа была изменена так, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 4Ah прерывания 21H. Как видно из рисунка 7, был создан новый блок (предпоследний), который занимает 64Кб. В последнем блоке хранится свободная память.

Код модуля LR3\_3.ASM см. в приложении В, тестирование модуля см. на рис. 6 и 7.

```
C:\>masm lr3_3.asm,,,;
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

47284 + 443591 Bytes symbol space free

      0 Warning Errors
      0 Severe Errors

C:\>link lr3_3.obj,,,;
Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

LINK : warning L4021: no stack segment

C:\>exe2bin lr3_3.exe lr3_3.com

C:\>lr3_3.com>res3_3.txt
```

Рисунок 6 - Запуск LR3\_3.com

```
RES3_3.TXT — Блокнот
Файл  Правка  Формат  Вид  Справка

Programm: third.
1. Amount of available memory:    648912 bytes.
2. Free memory!
3. Program requests memory
4. Extended memory size:         15360  kbytes.
5. Memory management block chain:
-----
MCB TYPE: 4Dh
BELONGS MS DOS - address:         0008h
Memory size:    16  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
FREE AREA - address:              0000h
Memory size:    64  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:              0040h
Memory size:    256  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:              0192h
Memory size:    144  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:              0192h
Memory size:  24144  bytes.
Last eight bytes: LR3_3
-----
MCB TYPE: 4Dh
Segment address PSP:              0192h
Memory size:  65536  bytes.
Last eight bytes: LR3_3
-----
MCB TYPE: 5Ah
FREE AREA - address:              0000h
Memory size:  559200  bytes.
Last eight bytes:
```

Рисунок 7 - Тестирование LR3\_3.com

Программа была изменена таким образом, чтобы запрос 64Кб памяти осуществлялся до ее освобождения. Проверяется флаг CF, если он равен единице, (что означает, что произошла ошибка) выводится сообщение на экран. В результате память не выделяется.

Код модуля LR3\_4.ASM см. в приложении Г, тестирование модуля см. на рис. 8 и 9.

```
C:\>masm lr3_4.asm,,,;
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

47284 + 443591 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link lr3_4.obj,,,;
Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

LINK : warning L4021: no stack segment

C:\>exe2bin lr3_4.exe lr3_4.com
C:\>lr3_4.com>res3_4.txt
```

Рисунок 8 - Запуск LR3\_4.COM

```
RES3_4.TXT - Блокнот
Файл  Правка  Формат  Вид  Справка

Programm: fourth.
1. Amount of available memory:  648912  bytes.
3. Program requests memory
A crash occurred while allocating memory!
2. Free memory!
4. Extended memory size:        15360  kbytes.
5. Memory management block chain:
-----
MCB TYPE: 4Dh
BELONGS MS DOS - address:      0008h
Memory size:  16  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
FREE AREA - address:          0000h
Memory size:  64  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:          0040h
Memory size: 256  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:          0192h
Memory size: 144  bytes.
Last eight bytes:
-----
MCB TYPE: 4Dh
Segment address PSP:          0192h
Memory size: 25088 bytes.
Last eight bytes: LR3_4
-----
MCB TYPE: 5Ah
FREE AREA - address:          0000h
Memory size: 623808 bytes.
Last eight bytes: <-->%F
```

Рисунок 9 - Тестирование LR3\_4.COM

## Ответы на контрольные вопросы.

### 1) Что означает "доступный объем памяти"?

Доступный объем памяти - это максимальный и доступный размер памяти, выделенной операционной системой, необходимый для запуска и выполнения программы.

### 2) Где МСВ блок Вашей программы в списке?

- В первой программе МСВ блок программы пятый в списке (последний).
- Во второй программе МСВ блок программы пятый в списке (предпоследний). Последний блок отведен под освобожденную память.
- В третьей программе МСВ блоки программы пятый и шестой (память, выделенная по запросу на 64 Кб) . Седьмой блок отведен под освобожденную память.



- В четвертой программе MCB блок программы пятый в списке (предпоследний). Последний блок отведен под освобожденную память.

### **3) Какой размер памяти занимает программа в каждом случае?**

- Первая программа занимает 648912 байт, то есть все доступную ей память.
- Вторая программа занимает 23344 байта, так как неиспользуемая память была освобождена.
- Третья программа занимает 89680 (24144 + 65536) байт, из них 65536 были запрошены дополнительно, а 24144 - непосредственно используются программой.
- Четвертая программа занимает необходимые 25088 байт, дополнительные 64Кб в результате ошибки выделены не были (свободной памяти не было).

### **Выводы.**

В результате выполнения лабораторной работы были исследованы структуры данных и работа функций управления памятью ядра операционной системы.

## ПРИЛОЖЕНИЕ А

### Содержимое файлы LR3\_1.ASM

COMMENT @

Максимова Анастасия, группа 8383, 3 лабораторная - 1 часть

@

CODE

SEGMENT

ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP

MAIN

EOF

EQU '\$'

SETPRECISION EQU 40

SETPR EQU 20

;ДАННЫЕ

PROG DB 0DH, 0AH, 0AH, 'Programm: first.',

EOF

AVAILABLE\_MEM DB 0DH, '1. Amount of available memory: bytes.',

EOF ;количество доступной памяти

EXTENDED\_MEM DB 0DH, 0AH, 0AH, '2. Extended memory size:

kbytes.', EOF ;размер расширенной памяти

MCB DB 0DH, 0AH, 0AH, '3. Memory management block chain:',

EOF

DELIMITER DB 0DH, 0AH, '-----

-----', EOF

TYPE1 DB 0DH, 0AH, 'MCB TYPE: 4Dh',

EOF

TYPE2 DB 0DH, 0AH, 'MCB TYPE: 5Ah',

EOF ;5Ah - last

```

CASE1_      DB  0DH,  0AH,  'FREE    AREA    -    address:
0000h',      EOF
CASE2_      DB  0DH,  0AH,  'BELONGS  DRIVER  OS    XMS    UMB    -
address:0006h', EOF
CASE3_      DB  0DH,  0AH,  'TOP    MEMORY  DRIVER    -    address:
0007h',      EOF
CASE4_      DB  0DH,  0AH,  'BELONGS    MS    DOS    -    address:
0008h',      EOF
CASE5_      DB  0DH,  0AH,  'CONTROL  UNIT  386MAX  UMB    -    address:
FFFAh',      EOF
CASE6_      DB  0DH,  0AH,  'BLOCKED    386MAX    -    address:
FFFDh',      EOF
CASE7_      DB  0DH,  0AH,  'BELONGS    386MAX    UMB    -    address:
FFFEh',      EOF
CASE_       DB  0DH,  0AH,      'Segment    address    PSP:
h',          EOF

SIZE_MEM    DB  0DH,  0AH,  'Memory  size:                      bytes.',
EOF
LAST        DB  0DH,  0AH,  'Last    eight  bytes:          ',
EOF

```

;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
        push    CX
        push    DX
        xor     AH,AH
        xor     DX,DX
        mov     CX,10

loop_bd:
        div     CX
        or      DL,30h

        ; перевод в ascii
        mov     [SI],DL
        dec     SI
        xor     DX,DX
        cmp     AX,10
        jae     loop_bd
        cmp     AL,00h
        je      end_1
        or      AL,30h
        mov     [SI],AL

end_1:
        pop     DX
        pop     CX
        ret

BYTE_TO_DEC     ENDP
;-----
GET_WRD_DEC     PROC NEAR
        BYTE_TO_DEC

```

; ИЗ

```

        push CX
        push DX

        mov CX, 10
loop_wd:
        div CX
        or DL, 30h
        ; перевод в ascii
        mov [SI], DL
        dec SI
        xor DX, DX

        cmp AX, 10
        jae loop_wd

        cmp AL, 00h
        je end_wb

        or AL, 30h
        mov [SI], AL
end_wb:

        pop DX
        pop CX
        retn

GET_WRD_DEC ENDP
;-----
PRINTF PROC NEAR
        push AX
        mov AH, 09h
        int 21h
        pop AX
        retn
PRINTF ENDP

```

```

;-----
GET_A_MEM          PROC NEAR          ;1 ЗАДАНИЕ - РАСПЕЧАТАТЬ КОЛИЧЕСТВО
    ДОСТУПНОЙ ПАМЯТИ В 10 С.С.

    push DX
    push AX
    push BX
    push SI

    mov     DX, OFFSET PROG
    call PRINTF
    xor     DX, DX

    mov     SI, OFFSET AVAILABLE_MEM
    add     SI, OFFSET SETPRECISION

    mov     AH, 04Ah          ;изменить размер блока память
    mov     BX, 0FFFFh        ;заведомо большая память
    int     21h               ;в BX будет лежать
    максимальный размер доступной памяти для этого блока

    mov     AX, 10h
    mul     BX                 ;параграф
    call    GET_WRD_DEC        ;перевод в 10 с с

    mov     DX, OFFSET AVAILABLE_MEM
    call PRINTF

    pop     SI
    pop     BX
    pop     AX
    pop     DX

    retn

```

```

GET_A_MEM          ENDP
;-----
GET_E_MEM          PROC NEAR          ;2 ЗАДАНИЕ - РАСПЕЧАТАТЬ РАЗМЕР
    РАСШИРЕННОЙ ПАМЯТИ В 10 С.С.

    push    AX
    push    BX
    push    DX
    push    SI

    mov     SI, OFFSET EXTENDED_MEM
    add     SI, OFFSET SETPRECISION

    xor     DX, DX
    mov     AL, 30h          ;запись адреса ячейки CMOS
    out     70h, AL
    in      AL, 71h          ;чтение младшего байта
    mov     BL, AL           ;размер расширенной памяти

    mov     AL, 31h          ;запись адреса ячейки CMOS
    out     70h, AL
    in      AL, 71h          ;чтение старшего байта
    mov     AH, AL           ;размер расширенной памяти
    mov     Al, BL

    call    GET_WRD_DEC      ;перевод в 10 с с

    mov     DX, OFFSET EXTENDED_MEM
    call    PRINTF

    pop     SI
    pop     DX
    pop     BX
    pop     AX

```



```

                                retn
GET_E_MEM                      ENDP
;-----
START_                          PROC NEAR
                                push    DX
                                push    AX
                                push    CX
                                push    SI

                                mov     DX, OFFSET DELIMITER
                                call    PRINTF

                                cmp     AX, 4Dh                ;печать типа
                                jne     cout
                                mov     DX, OFFSET TYPE1
                                jmp     exit

cout:
                                mov     DX, OFFSET TYPE2

exit:
                                call    PRINTF
                                pop     SI
                                pop     CX
                                pop     AX
                                pop     DX
                                retn

START_                          ENDP
;-----
GET_ADDRESS                     PROC NEAR
                                push    AX
                                push    ES
                                push    DX
                                push    DI

                                mov     AX, ES:[01h]

```

```

                                cmp     AX, 0000h
                                jne     case2
                                mov     DX, OFFSET CASE1_
                                jmp     write
case2:
                                cmp     AX, 0006h
                                jne     case3
                                mov     DX, OFFSET CASE2_
                                jmp     write
case3:
                                cmp     AX, 0007h
                                jne     case4
                                mov     DX, OFFSET CASE3_
                                jmp     write
case4:
                                cmp     AX, 0008h
                                jne     case5
                                mov     DX, OFFSET CASE4_
                                jmp     write
case5:
                                cmp     AX, 0FFFAh
                                jne     case6
                                mov     DX, OFFSET CASE5_
                                jmp     write
case6:
                                cmp     AX, 0FFFDh
                                jne     case7
                                mov     DX, OFFSET CASE6_
                                jmp     write
case7:
                                cmp     AX, 0FFFEh
                                jne     case0
                                mov     DX, OFFSET CASE7_

```

```

                                jmp      write
case0:
                                xor      DI, DI
                                xor      DX, DX
                                mov      DI, OFFSET CASE_
                                add      DI, OFFSET SETPRECISION
                                call     WRD_TO_HEX
                                mov      DX, OFFSET CASE_
write:
                                call     PRINTF
end_a:
                                pop      DI
                                pop      DX
                                pop      ES
                                pop      AX
                                retn
GET_ADRESS                     ENDP
;-----
GET_SIZE_MEM  PROC  NEAR
                                push     SI
                                push     BX
                                push     ES
                                push     AX
                                push     DX

                                mov      SI, OFFSET SIZE_MEM
                                add      SI, OFFSET SETPR

                                mov      BX, ES:[03h] ;размер участка в параграфах -
перевести в 10 с/с

                                mov      AX, 10h
                                mul      BX              ;параграф
                                call     GET_WRD_DEC      ;перевод в 10 с с

```

```

        mov     DX, OFFSET SIZE_MEM
        call    PRINTF

        pop     DX
        pop     AX
        pop     ES
        pop     BX
        pop     DX
        retn

GET_SIZE_MEM    ENDP
;-----
GET_SYMBOLS     PROC NEAR
        push    DX
        push    DI
        push    CX
        push    AX
        push    ES

        mov     DX, OFFSET LAST
        call    PRINTF

        mov     DI, 0
        mov     CX, 8

symbol:
        xor     AX, AX
        mov     AH, 02h
        mov     DL, ES:[08h + DI] ;08h - там лежат символы
        int     21h
        inc     DI
        loop    symbol

        pop     ES
        pop     AX

```

```

        pop        CX
        pop        DI
        pop        DX
        retn

GET_SYMBOLS    ENDP

;-----
GET_MCB        PROC NEAR                                ;3 ЗАДАНИЕ - ВЫВОД ЦЕПОЧКИ
        БЛОКОВ УПРАВЛЕНИЯ ПАМЯТЬЮ

        push DX
        push AX
        push ES
        push BX

        mov        DX, OFFSET MCB
        call PRINTF

        mov        AH, 52h                                ;доступ к указателю
        на структуру

        int 21h                                ;list of lists
        mov        AX, ES:[BX - 02h] ;получение адреса
        mov        ES, AX                                ;первого MCB

replay:

        xor        AX, AX
        mov        AL, ES:[00h]                                ;type MCB
        push AX

        call START_
        call GET_ADRESS
        call GET_SIZE_MEM
        call GET_SYMBOLS

        pop        AX
        cmp        AX, 4Dh
        jne        end_MCB

```

```

                                mov     AX, ES:[03h]      ;размер участка в параграфах
                                mov     BX, ES            ;адрес начала
                                add     BX, AX            ;адрес конца
                                inc     BX               ;в BX - адрес след
элементы
                                mov     ES, BX
                                jmp     replay
end_MCB:
                                pop     BX
                                pop     ES
                                pop     AX
                                pop     DX
                                retn
GET_MCB      ENDP
;-----
MAIN:
    call GET_A_MEM
    call  GET_E_MEM
    call GET_MCB

;выход в DOS
    sub     AL, AL
    mov     AH, 4Ch
    int     21h

CODE      ENDS
          END  START

```

## ПРИЛОЖЕНИЕ Б

### Содержимое файлы LR3\_2.ASM

COMMENT @

Максимова Анастасия, группа 8383, 3 лабораторная - 2 часть

@

CODE

SEGMENT

ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING

ORG 100H

START:

JMP

MAIN

EOF

EQU

'\$'

SETPRECISION EQU

40

SETPR

EQU

20

;ДАННЫЕ

PROG

DB

0DH, 0AH, 0AH, 'Programm: second. ',

EOF

AVAILABLE\_MEM DB

0DH, '1. Amount of available memory: bytes.',

EOF

;количество доступной памяти

FR\_MEM

DB

0DH, '2. Free memory!',

EOF

;количество доступной памяти

EXTENDED\_MEM DB

0DH, 0AH, 0AH, '3. Extended memory size:

kbytes.', EOF

;размер расширенной памяти

MCB

DB

0DH, 0AH, 0AH, '4. Memory management block chain:',

EOF

DELIMITER

DB

0DH, 0AH, '-----

-----',

EOF

TYPE1

DB

0DH, 0AH, 'MCB TYPE: 4Dh ',

EOF

```

TYPE2          DB  0DH, 0AH, 'MCB TYPE: 5Ah    ',
EOF            ;5Ah - last

CASE1_         DB  0DH, 0AH, 'FREE AREA - address:
0000h',        EOF
CASE2_         DB  0DH, 0AH, 'BELONGS DRIVER OS XMS UMB -
address:0006h', EOF
CASE3_         DB  0DH, 0AH, 'TOP MEMORY DRIVER  - address:
0007h',        EOF
CASE4_         DB  0DH, 0AH, 'BELONGS MS DOS  - address:
0008h',        EOF
CASE5_         DB  0DH, 0AH, 'CONTROL UNIT 386MAX UMB - address:
FFFAh',        EOF
CASE6_         DB  0DH, 0AH, 'BLOCKED 386MAX - address:
FFFDh',        EOF
CASE7_         DB  0DH, 0AH, 'BELONGS 386MAX UMB - address:
FFFEh',        EOF
CASE_          DB  0DH, 0AH, 'Segment address PSP:
h',            EOF

SIZE_MEM       DB  0DH, 0AH, 'Memory size:      bytes.',
EOF
LAST           DB  0DH, 0AH, 'Last eight bytes:  ',
EOF

;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
        push CX
        push DX
        xor  AH, AH
        xor  DX, DX
        mov  CX, 10

loop_bd:

        div     CX
        or      DL, 30h           ; перевод в ascii
        mov     [SI], DL
        dec     SI
        xor     DX, DX
        cmp     AX, 10
        jae     loop_bd
        cmp     AL, 00h
        je      end_1
        or      AL, 30h           ; перевод в ascii
        mov     [SI], AL

end_1:

        pop DX
        pop CX
        ret
BYTE_TO_DEC     ENDP
;-----

```

```

GET_WRD_DEC      PROC NEAR      ; из BYTE_TO_DEC
                  push CX
                  push DX

                  mov  CX, 10

loop_wd:
                  div  CX
                  or   DL, 30h      ; перевод в ascii
                  mov  [SI], DL
                  dec  SI
                  xor  DX, DX

                  cmp  AX, 10      ; если в целой части есть, что еще
          делить
                  jae  loop_wd

                  cmp  AL, 00h
                  je   end_wb

                  or   AL, 30h
                  mov  [SI], AL

end_wb:
                  pop  DX
                  pop  CX
                  retn

GET_WRD_DEC      ENDP

;-----
PRINTF          PROC NEAR
                  push  AX
                  mov   AH, 09h
                  int   21h
                  pop   AX
                  retn

```

PRINTF                    ENDP

;-----

GET\_A\_MEM                    PROC NEAR                    ;1 ЗАДАНИЕ - РАСПЕЧАТАТЬ КОЛИЧЕСТВО  
                              ДОСТУПНОЙ ПАМЯТИ В 10 С.С.

                              push DX  
                              push AX  
                              push BX  
                              push SI

                              mov DX, OFFSET PROG  
                              call PRINTF  
                              xor DX, DX

                              mov SI, OFFSET AVAILABLE\_MEM  
add SI, OFFSET SETPRECISION

                              mov AH, 04Ah                    ;изменить размер блока память  
                              mov BX, 0FFFFh                    ;заведомо большая память  
                              int 21h                                ;в BX будет лежать  
                              максимальный размер доступный для этого блока

                              mov AX, 10h  
                              mul BX                                ;параграф  
call GET\_WRD\_DEC                    ;перевод в 10 с с

                              mov DX, OFFSET AVAILABLE\_MEM  
call PRINTF

                              pop SI  
                              pop BX  
                              pop AX  
                              pop DX

```

                                retn
GET_A_MEM                      ENDP
;-----
GET_E_MEM                      PROC NEAR                                ;2 ЗАДАНИЕ - РАСПЕЧАТАТЬ РАЗМЕР
                                РАСШИРЕННОЙ ПАМЯТИ В 10 С.С.
                                push    AX
                                push    BX
                                push    DX
                                push    SI

                                mov     SI, OFFSET EXTENDED_MEM
                                add     SI, OFFSET SETPRECISION

                                xor     DX, DX
                                mov     AL, 30h                        ;запись адреса ячейки CMOS
                                out     70h, AL
                                in      AL, 71h                        ;чтение младшего байта
                                mov     BL, AL                        ;размер расширенной памяти

                                mov     AL, 31h                        ;запись адреса ячейки CMOS
                                out     70h, AL
                                in      AL, 71h                        ;чтение старшего байта
                                mov     AH, AL                        ;размер расширенной памяти
                                mov     Al, BL

                                call    GET_WRD_DEC                    ;перевод в 10 с с

                                mov     DX, OFFSET EXTENDED_MEM
                                call    PRINTF

                                pop     SI
                                pop     DX
                                pop     BX
                                pop     AX

```

```

                                retn
GET_E_MEM                      ENDP
;-----
START_                          PROC NEAR
                                push    DX
                                push    AX
                                push    CX
                                push    SI

                                mov     DX, OFFSET DELIMITER
                                call    PRINTF

                                cmp     AX, 4Dh                ;печать типа
                                jne     cout
                                mov     DX, OFFSET TYPE1
                                jmp     exit

cout:
                                mov     DX, OFFSET TYPE2

exit:
                                call    PRINTF
                                pop     SI
                                pop     CX
                                pop     AX
                                pop     DX
                                retn

START_                          ENDP
;-----
GET_ADDRESS                    PROC NEAR
                                push    AX
                                push    ES
                                push    DX
                                push    DI

```

```

        mov     AX, ES:[01h]

        cmp     AX, 0000h
        jne     case2
        mov     DX, OFFSET CASE1_
        jmp     write

case2:
        cmp     AX, 0006h
        jne     case3
        mov     DX, OFFSET CASE2_
        jmp     write

case3:
        cmp     AX, 0007h
        jne     case4
        mov     DX, OFFSET CASE3_
        jmp     write

case4:
        cmp     AX, 0008h
        jne     case5
        mov     DX, OFFSET CASE4_
        jmp     write

case5:
        cmp     AX, 0FFFAh
        jne     case6
        mov     DX, OFFSET CASE5_
        jmp     write

case6:
        cmp     AX, 0FFFDh
        jne     case7
        mov     DX, OFFSET CASE6_
        jmp     write

case7:
        cmp     AX, 0FFFEh

```

```

        jne      case0
        mov      DX, OFFSET CASE7_
        jmp      write

case0:

        xor      DI, DI
        xor      DX, DX
        mov      DI, OFFSET CASE_
        add      DI, OFFSET SETPRECISION
        call     WRD_TO_HEX
        mov      DX, OFFSET CASE_

write:

        call     PRINTF

end_a:

        pop      DI
        pop      DX
        pop      ES
        pop      AX

        retn

GET_ADRESS      ENDP
;-----
GET_SIZE_MEM    PROC NEAR
        push     SI
        push     BX
        push     ES
        push     AX
        push     DX

        mov      SI, OFFSET SIZE_MEM
        add      SI, OFFSET SETPR

        mov      BX, ES:[03h] ;размер участка в параграфах -
        переводит в 10 с/с

```



```

mov     AX, 10h
mul     BX                      ;параграф
call    GET_WRD_DEC            ;перевод в 10 с с

```

```

mov     DX, OFFSET SIZE_MEM
call    PRINTF

```

```

pop     DX
pop     AX
pop     ES
pop     BX
pop     DX
retn

```

```
GET_SIZE_MEM ENDP
```

```
;-----
```

```
GET_SYMBOLS PROC NEAR
```

```

push    DX
push    DI
push    CX
push    AX
push    ES

```

```

mov     DX, OFFSET LAST
call    PRINTF

```

```

mov     DI, 0
mov     CX, 8

```

```
symbol:
```

```

xor     AX, AX
mov     AH, 02h
mov     DL, ES:[08h + DI] ;08h - там лежат символы
int     21h
inc     DI
loop    symbol

```

```

        pop        ES
        pop        AX
        pop        CX
        pop        DI
        pop        DX
        retn
GET_SYMBOLS    ENDP
;-----
GET_MCB        PROC NEAR                ;3 ЗАДАНИЕ - ВЫВОД ЦЕПОЧКИ
        БЛОКОВ УПРАВЛЕНИЯ ПАМЯТЬЮ
        push DX
        push AX
        push ES
        push BX

        mov        DX, OFFSET MCB
        call PRINTF

        mov        AH, 52h                ;доступ к указателю
на структуру
        int 21h                ;list of lists
        mov        AX, ES:[BX - 02h] ;получение адреса
        mov        ES, AX                ;первого MCB
replay:
        xor        AX, AX
        mov        AL, ES:[00h]        ;type MCB
        push AX

        call START_
        call GET_ADRESS
        call GET_SIZE_MEM
        call GET_SYMBOLS

```

```

pop        AX
cmp        AX, 4Dh
jne        end_MCB

mov        AX, ES:[03h]        ;размер участка в параграфах
mov        BX, ES              ;адрес начала
add        BX, AX              ;адрес конца
inc        BX                  ;в BX - адрес след
элемента

mov        ES, BX
jmp        replay

end_MCB:

pop        BX
pop        ES
pop        AX
pop        DX
retn

GET_MCB    ENDP

;-----
FREE_MEMORY PROC NEAR        ;освобождение памяти
push AX
push BX
push DX

mov        DX, OFFSET FR_MEM
call PRINTF

mov        AH, 4Ah
mov        BX,  OFFSET END_PR
int        21h

pop        DX
pop        BX
pop        AX

```

```

                                retn
FREE_MEMORY                    ENDP
;-----
MAIN:
    call GET_A_MEM
    call FREE_MEMORY
    call  GET_E_MEM
    call GET_MCB

    ;выход в DOS
    sub     AL, AL
    mov     AH, 4Ch
    int     21h
END_PR  DB    0
CODE    ENDS
END     START

```

## ПРИЛОЖЕНИЕ В

### Содержимое файлы LR3\_3.ASM

```
COMMENT @
Максимова Анастасия, группа 8383, 3 лабораторная - 3 часть
@

CODE                SEGMENT
                    ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
                    ORG 100H

START:              JMP          MAIN

EOF                EQU          '$'
SETPRECISION        EQU          40
SETPR               EQU          20

;ДАННЫЕ
PROG                DB    0DH, 0AH, 0AH, 'Programm: third.',
EOF
AVAILABLE_MEM       DB    0DH, '1. Amount of available memory:
bytes.', EOF        ;количество доступной памяти
FR_MEM              DB    0DH, '2. Free memory!',
EOF                ;количество доступной памяти
GIVE_MEM            DB    0DH, '3. Program requests memory',
EOF
EXTENDED_MEM        DB    0DH, 0AH, 0AH, '4. Extended memory size:
kbytes.', EOF       ;размер расширенной памяти

MCB                 DB    0DH, 0AH, 0AH, '5. Memory management block chain:',
EOF
DELIMITER           DB    0DH, 0AH, '-----'
-----', EOF
TYPE1               DB    0DH, 0AH, 'MCB TYPE: 4Dh    ',
EOF
TYPE2               DB    0DH, 0AH, 'MCB TYPE: 5Ah    ',
EOF                ;5Ah - last

CASE1_              DB    0DH, 0AH, 'FREE AREA - address:
0000h', EOF
CASE2_              DB    0DH, 0AH, 'BELONGS DRIVER OS XMS UMB -
address:0006h', EOF
CASE3_              DB    0DH, 0AH, 'TOP MEMORY DRIVER  - address:
0007h', EOF
CASE4_              DB    0DH, 0AH, 'BELONGS MS DOS  - address:
0008h', EOF
CASE5_              DB    0DH, 0AH, 'CONTROL UNIT 386MAX UMB - address:
FFFAh', EOF
```

```

CASE6_          DB    0DH, 0AH, 'BLOCKED 386MAX - address:
FFFDh',        EOF
CASE7_          DB    0DH, 0AH, 'BELONGS 386MAX UMB - address:
FFFEh',        EOF
CASE_           DB    0DH, 0AH, 'Segment address PSP:
h',            EOF

SIZE_MEM        DB    0DH, 0AH, 'Memory size:          bytes.',
EOF
LAST            DB    0DH, 0AH, 'Last eight bytes:  ',
EOF

;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
                push CX
                push DX
                xor  AH,AH
                xor  DX,DX
                mov  CX,10

loop_bd:
                div  CX
                or   DL,30h           ; перевод в ascii
                mov  [SI],DL
                dec  SI
                xor  DX,DX
                cmp  AX,10
                jae  loop_bd
                cmp  AL,00h
                je   end_1
                or   AL,30h
                mov  [SI],AL

end_1:
                pop  DX
                pop  CX
                ret

BYTE_TO_DEC     ENDP
;-----
GET_WRD_DEC     PROC NEAR           ; из BYTE_TO_DEC

                push CX
                push DX

                mov  CX, 10

loop_wd:
                div  CX
                or   DL, 30h         ; перевод в ascii
                mov  [SI], DL
                dec  SI
                xor  DX, DX

                cmp  AX, 10

```

```

                                jae  loop_wd

                                cmp  AL, 00h
                                je    end_wb

                                or     AL, 30h
                                mov    [SI], AL

end_wb:

                                pop  DX
                                pop  CX
                                retn

GET_WRD_DEC      ENDP
;-----
PRINTF          PROC  NEAR
                push  AX
                mov    AH, 09h
                int    21h
                pop    AX
                retn
PRINTF          ENDP
;-----
GET_A_MEM        PROC  NEAR                                ;1 ЗАДАНИЕ - РАСПЕЧАТАТЬ КОЛИЧЕСТВО
ДОСТУПНОЙ ПАМЯТИ В 10 С.С.

                push  DX
                push  AX
                push  BX
                push  SI

                mov    DX, OFFSET  PROG
                call  PRINTF
                xor     DX, DX

                mov     SI, OFFSET AVAILABLE_MEM
                add     SI, OFFSET SETPRECISION

                mov     AH, 04Ah                                ;изменить размер блока
память

                mov     BX, 0FFFFh                                ;заведомо большая память
                int     21h                                ;в BX будет лежать
максимальный размер доступный для этого блока

                mov     AX, 10h
                mul     BX                                ;параграф
                call    GET_WRD_DEC                        ;перевод в 10 с с

                mov     DX, OFFSET AVAILABLE_MEM
                call  PRINTF

                pop     SI

```



```

        pop        BX
        pop        AX
        pop        DX

        retn

GET_A_MEM      ENDP
;-----
GET_E_MEM      PROC NEAR                                ;2 ЗАДАНИЕ - РАСПЕЧАТАТЬ РАЗМЕР
РАСШИРЕННОЙ ПАМЯТИ В 10 С.С.
        push       AX
        push       BX
        push       DX
        push       SI

        mov        SI, OFFSET EXTENDED_MEM
        add        SI, OFFSET SETPRECISION

        xor        DX, DX
        mov        AL, 30h                                ;запись адреса ячейки
CMOS
        out        70h, AL
        in         AL, 71h                                ;чтение младшего байта
        mov        BL, AL                                ;размер расширенной
памяти

        mov        AL, 31h                                ;запись адреса ячейки
CMOS
        out        70h, AL
        in         AL, 71h                                ;чтение старшего байта
        mov        AH, AL                                ;размер расширенной
памяти

        mov        AL, BL

        call       GET_WRD_DEC                            ;перевод в 10 с с

        mov        DX, OFFSET EXTENDED_MEM
        call       PRINTF

        pop        SI
        pop        DX
        pop        BX
        pop        AX

        retn

GET_E_MEM      ENDP
;-----
START_         PROC NEAR
        push       DX
        push       AX
        push       CX
        push       SI

```

```

mov     DX, OFFSET DELIMITER
call    PRINTF

cmp     AX, 4Dh                ;печать типа

jne     cout
mov     DX, OFFSET TYPE1
jmp     exit

cout:
mov     DX, OFFSET TYPE2

exit:
call    PRINTF
pop     SI
pop     CX
pop     AX
pop     DX
retn

START_
ENDP
;-----
GET_ADDRESS    PROC NEAR
push  AX
push  ES
push  DX
push  DI

mov     AX, ES:[01h]

cmp     AX, 0000h
jne     case2
mov     DX, OFFSET CASE1_
jmp     write

case2:
cmp     AX, 0006h
jne     case3
mov     DX, OFFSET CASE2_
jmp     write

case3:
cmp     AX, 0007h
jne     case4
mov     DX, OFFSET CASE3_
jmp     write

case4:
cmp     AX, 0008h
jne     case5
mov     DX, OFFSET CASE4_
jmp     write

case5:
cmp     AX, 0FFFAh
jne     case6
mov     DX, OFFSET CASE5_

```

```

case6:                jmp      write

                    cmp      AX, 0FFFDh
                    jne      case7
                    mov      DX, OFFSET CASE6_
                    jmp      write

case7:                cmp      AX, 0FFFEh
                    jne      case0
                    mov      DX, OFFSET CASE7_
                    jmp      write

case0:                xor      DI, DI
                    xor      DX, DX
                    mov      DI, OFFSET CASE_
add      DI, OFFSET SETPRECISION
                    call     WRD_TO_HEX
                    mov      DX, OFFSET CASE_

write:                call     PRINTF

end_a:                pop      DI
                    pop      DX
                    pop      ES
                    pop      AX

                    retn

GET_ADDRESS          ENDP
;-----
GET_SIZE_MEM         PROC NEAR
                    push     SI
                    push     BX
                    push     ES
                    push     AX
                    push     DX

                    mov      SI, OFFSET SIZE_MEM
add      SI, OFFSET SETPR

                    mov      BX, ES:[03h] ;размер участка в параграфах -
перевести в 10 с/с

                    mov      AX, 10h
                    mul      BX                ;параграф
                    call     GET_WRD_DEC        ;перевод в 10 с с

                    mov      DX, OFFSET SIZE_MEM
                    call     PRINTF

                    pop      DX
                    pop      AX

```

```

                                pop     ES
                                pop     BX
                                pop     DX
                                retn
GET_SIZE_MEM      ENDP
;-----
GET_SYMBOLS      PROC NEAR
                                push    DX
                                push    DI
                                push    CX
                                push    AX
                                push    ES

                                mov     DX, OFFSET LAST
                                call    PRINTF

                                mov     DI, 0
                                mov     CX, 8
symbol:
                                xor     AX, AX
                                mov     AH, 02h
                                mov     DL, ES:[08h + DI] ;08h - там лежат
СИМВОЛЫ
                                int     21h
                                inc     DI
                                loop    symbol

                                pop     ES
                                pop     AX
                                pop     CX
                                pop     DI
                                pop     DX
                                retn
GET_SYMBOLS      ENDP
;-----
GET_MCB          PROC NEAR ;3 ЗАДАНИЕ - ВЫВОД
ЦЕПОЧКИ БЛОКОВ УПРАВЛЕНИЯ ПАМЯТЬЮ
                                push    DX
                                push    AX
                                push    ES
                                push    BX

                                mov     DX, OFFSET MCB
                                call    PRINTF

                                mov     AH, 52h ;доступ к
указателю на структуру
                                int     21h ;list of lists
                                mov     AX, ES:[BX - 02h] ;получение адреса
                                mov     ES, AX ;первого MCB
replay:

```

```

xor      AX, AX
mov      AL, ES:[00h]      ;type MCB
push     AX

call     START_
call     GET_ADRESS
call     GET_SIZE_MEM
call     GET_SYMBOLS

pop      AX
cmp      AX, 4Dh
jne      end_MCB

параграфах      mov      AX, ES:[03h]      ;размер участка в

mov      BX, ES      ;адрес начала

add      BX, AX      ;адрес конца
inc      BX      ;в BX - адрес

след элемента

mov      ES, BX
jmp      replay

end_MCB:

pop      BX
pop      ES
pop      AX
pop      DX
retn
ENDP

GET_MCB
;-----
FREE_MEMORY      PROC NEAR      ;освобождение памяти
push     AX
push     BX
push     DX

mov      DX, OFFSET FR_MEM
call     PRINTF

mov      AH, 4Ah
mov      BX,  OFFSET END_PR
int      21h

pop      DX
pop      BX
pop      AX
retn
ENDP

FREE_MEMORY
;-----
GIVE_ME_MEMORY      PROC NEAR      ;запрашивание памяти
push     AX
push     BX

```

```

        push DX

        mov     DX, OFFSET GIVE_MEM
        call    PRINTF

        mov     AH, 48h
        mov     BX, 1000h        ;1024*64/16 В 10
        int     21h

        pop     DX
        pop     BX
        pop     AX
        retn

GIVE_ME_MEMORY ENDP
;-----
MAIN:
        call    GET_A_MEM
        call    FREE_MEMORY
        call    GIVE_ME_MEMORY
        call    GET_E_MEM
        call    GET_MCB

        ;ВЫХОД В DOS
        sub     AL, AL
        mov     AH, 4Ch
        int     21h
END_PR    DB     0
CODE ENDS
        END     START

```

## ПРИЛОЖЕНИЕ Г

### Содержимое файлы LR3\_4.ASM

COMMENT @

Максимова Анастасия, группа 8383, 3 лабораторная - 4 часть

@

CODE

SEGMENT

ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING

ORG 100H

START:

JMP

MAIN

EOF

EQU

'\$'

SETPRECISION

EQU

40

SETPR

EQU

20

;ДАННЫЕ

PROG

DB

0DH, 0AH, 0AH, 'Programm: fourth. ',

EOF

AVAILABLE\_MEM DB

0DH, 0AH, '1. Amount of available memory:

bytes.',

EOF

;количество доступной памяти

FR\_MEM

DB

0DH, 0AH, '2. Free memory!',

EOF

;количество доступной памяти

GIVE\_MEM

DB

0DH, 0AH, '3. Program requests memory',

EOF

EXTENDED\_MEM DB

0DH, 0AH, 0AH, '4. Extended memory size:

kbytes.', EOF

;размер расширенной памяти

MCB

DB

0DH, 0AH, 0AH, '5. Memory management block chain:',

EOF

DELIMITER

DB

0DH, 0AH, '-----

-----',

EOF

```

TYPE1          DB  0DH, 0AH, 'MCB TYPE: 4Dh    ',
                EOF
TYPE2          DB  0DH, 0AH, 'MCB TYPE: 5Ah    ',
                EOF
                ;5Ah - last

CASE1_         DB  0DH, 0AH, 'FREE AREA - address:
                0000h',
                EOF
CASE2_         DB  0DH, 0AH, 'BELONGS DRIVER OS XMS UMB -
                address:0006h',
                EOF
CASE3_         DB  0DH, 0AH, 'TOP MEMORY DRIVER  - address:
                0007h',
                EOF
CASE4_         DB  0DH, 0AH, 'BELONGS MS DOS  - address:
                0008h',
                EOF
CASE5_         DB  0DH, 0AH, 'CONTROL UNIT 386MAX UMB - address:
                FFFAh',
                EOF
CASE6_         DB  0DH, 0AH, 'BLOCKED 386MAX - address:
                FFFDh',
                EOF
CASE7_         DB  0DH, 0AH, 'BELONGS 386MAX UMB - address:
                FFFEh',
                EOF
CASE_          DB  0DH, 0AH, 'Segment address PSP:
                h',
                EOF

SIZE_MEM       DB  0DH, 0AH, 'Memory size:      bytes.',
                EOF
LAST          DB  0DH, 0AH, 'Last eight bytes:  ',
                EOF

ERROR         DB  0DH, 0AH, 'A crash occurred while allocating
                memory!',
                EOF
ENDL          DB  0DH, 0AH, EOF

;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
        push    CX
        push    DX
        xor     AH,AH
        xor     DX,DX
        mov     CX,10
loop_bd:
        div     CX
        or      DL,30h           ; перевод в ascii
        mov     [SI],DL
        dec     SI
        xor     DX,DX
        cmp     AX,10
        jae     loop_bd
        cmp     AL,00h
        je      end_l
        or      AL,30h           ; перевод в ascii
        mov     [SI],AL

```

```

end_1:
    pop DX
    pop CX
    ret

BYTE_TO_DEC ENDP
;-----
GET_WRD_DEC      PROC NEAR      ;из BYTE_TO_DEC

    push CX
    push DX

    mov  CX, 10

loop_wd:
    div  CX
    or   DL, 30h      ; перевод в ascii
    mov  [SI], DL
    dec  SI
    xor  DX, DX

    cmp  AX, 10      ;если в целой части есть, что еще
    делить
    jae  loop_wd

    cmp  AL, 00h      ;если 0 заканчиваем
    je   end_wb

    or   AL, 30h      ; перевод в ascii
    mov  [SI], AL

end_wb:
    pop DX
    pop CX
    retn

GET_WRD_DEC      ENDP

```

;-----

```
PRINTF      PROC  NEAR
            push    AX
            mov     AH, 09h
            int     21h
            pop     AX
            retn
PRINTF      ENDP
```

;-----

```
GET_A_MEM      PROC  NEAR                                ;1 ЗАДАНИЕ - РАСПЕЧАТАТЬ КОЛИЧЕСТВО
                ДОСТУПНОЙ ПАМЯТИ В 10 С.С.
```

```
                push DX
                push AX
                push BX
                push  SI

                mov     DX, OFFSET  PROG
                call    PRINTF
                xor     DX, DX

                mov     SI, OFFSET AVAILABLE_MEM
                add     SI, OFFSET SETPRECISION

                mov     AH, 04Ah                        ;изменить размер блока память
                mov     BX, 0FFFFh                      ;заведомо большая память
                int     21h                              ;в BX будет лежать
                максимальный размер доступный для этого блока

                mov     AX, 10h
                mul     BX                                ;параграф
                call    GET_WRD_DEC                      ;перевод в 10 с с

                mov     DX, OFFSET AVAILABLE_MEM
```

```

call PRINTF

pop     SI
pop     BX
pop     AX
pop     DX

retn

GET_A_MEM      ENDP
;-----
GET_E_MEM      PROC NEAR      ;2 ЗАДАНИЕ - РАСПЕЧАТАТЬ РАЗМЕР
    РАСШИРЕННОЙ ПАМЯТИ В 10 С.С.
    push     AX
    push     BX
    push     DX
    push     SI

    mov      SI, OFFSET EXTENDED_MEM
    add      SI, OFFSET SETPRECISION

    xor      DX, DX
    mov      AL, 30h          ;запись адреса ячейки CMOS
    out      70h, AL
    in       AL, 71h          ;чтение младшего байта
    mov      BL, AL           ;размер расширенной памяти

    mov      AL, 31h          ;запись адреса ячейки CMOS
    out      70h, AL
    in       AL, 71h          ;чтение старшего байта
    mov      AH, AL           ;размер расширенной памяти
    mov      AL, BL

    call     GET_WRD_DEC      ;перевод в 10 с с

```

```

                                mov     DX, OFFSET EXTENDED_MEM
                                call    PRINTF

                                pop     SI
                                pop     DX
                                pop     BX
                                pop     AX

                                retn

GET_E_MEM                      ENDP
;-----
START_                          PROC NEAR
                                push    DX
                                push    AX
                                push    CX
                                push    SI

                                mov     DX, OFFSET DELIMITER
                                call    PRINTF

                                cmp     AX, 4Dh                ;печать типа
                                jne     cout
                                mov     DX, OFFSET TYPE1
                                jmp     exit

cout:
                                mov     DX, OFFSET TYPE2

exit:
                                call    PRINTF
                                pop     SI
                                pop     CX
                                pop     AX
                                pop     DX
                                retn

START_                          ENDP

```

```

;-----
GET_ADRESS      PROC  NEAR
                 push  AX
                 push  ES
                 push  DX
                 push  DI

                 mov    AX, ES:[01h]

                 cmp     AX, 0000h
                 jne     case2
                 mov     DX, OFFSET CASE1_
                 jmp     write

case2:
                 cmp     AX, 0006h
                 jne     case3
                 mov     DX, OFFSET CASE2_
                 jmp     write

case3:
                 cmp     AX, 0007h
                 jne     case4
                 mov     DX, OFFSET CASE3_
                 jmp     write

case4:
                 cmp     AX, 0008h
                 jne     case5
                 mov     DX, OFFSET CASE4_
                 jmp     write

case5:
                 cmp     AX, 0FFFAh
                 jne     case6
                 mov     DX, OFFSET CASE5_
                 jmp     write

case6:

```

```

        cmp     AX, 0FFFDh
        jne     case7
        mov     DX, OFFSET CASE6_
        jmp     write

case7:

        cmp     AX, 0FFFEh
        jne     case0
        mov     DX, OFFSET CASE7_
        jmp     write

case0:

        xor     DI, DI
        xor     DX, DX
        mov     DI, OFFSET CASE_
        add     DI, OFFSET SETPRECISION
        call    WRD_TO_HEX
        mov     DX, OFFSET CASE_

write:

        call    PRINTF

end_a:

        pop     DI
        pop     DX
        pop     ES
        pop     AX

        retn

GET_ADDRESS    ENDP
;-----
GET_SIZE_MEM  PROC NEAR
        push    SI
        push    BX
        push    ES
        push    AX
        push    DX

```



```

        mov     SI, OFFSET SIZE_MEM
add     SI, OFFSET SETPR

        mov     BX, ES:[03h] ;размер участка в параграфах -
перевести в 10 с/с

        mov     AX, 10h
        mul     BX             ;параграф
        call    GET_WRD_DEC    ;перевод в 10 с с

        mov     DX, OFFSET SIZE_MEM
        call    PRINTF

        pop     DX
        pop     AX
        pop     ES
        pop     BX
        pop     DX
        retn

GET_SIZE_MEM ENDP
;-----
GET_SYMBOLS PROC NEAR
        push    DX
        push    DI
        push    CX
        push    AX
        push    ES

        mov     DX, OFFSET LAST
        call    PRINTF

        mov     DI, 0
        mov     CX, 8

```

symbol:

```

        xor     AX, AX
        mov     AH, 02h
        mov     DL, ES:[08h + DI]    ;08h - там лежат символы
        int     21h
        inc     DI
        loop    symbol

        pop     ES
        pop     AX
        pop     CX
        pop     DI
        pop     DX
        retn

GET_SYMBOLS    ENDP
;-----
GET_MCB        PROC NEAR                ;3 ЗАДАНИЕ - ВЫВОД ЦЕПОЧКИ БЛОКОВ
        УПРАВЛЕНИЯ ПАМЯТЬЮ

        push    DX
        push    AX
        push    ES
        push    BX

        mov     DX, OFFSET MCB
        call    PRINTF

        mov     AH, 52h                ;доступ к указателю
        на структуру

        int     21h                    ;list of lists
        mov     AX, ES:[BX - 02h]    ;получение адреса
        mov     ES, AX                ;первого MCB

replay:
        xor     AX, AX
        mov     AL, ES:[00h]        ;type MCB
        push    AX

```

```

call START_
call GET_ADRESS
call GET_SIZE_MEM
call GET_SYMBOLS

pop      AX
cmp      AX, 4Dh
jne      end_MCB

mov      AX, ES:[03h]      ;размер участка в параграфах
mov      BX, ES            ;адрес начала
add      BX, AX            ;адрес конца
inc      BX               ;в BX - адрес след
элемента

mov      ES, BX
jmp      replay

end_MCB:

pop      BX
pop      ES
pop      AX
pop      DX
retn

GET_MCB      ENDP
;-----
FREE_MEMORY  PROC NEAR      ;освобождение памяти
push AX
push BX
push DX

mov      DX, OFFSET FR_MEM
call PRINTF

mov      AH, 4Ah

```

```

        mov     BX,  OFFSET END_PR
        int     21h

        pop     DX
        pop     BX
        pop     AX
        retn

FREE_MEMORY    ENDP
;-----
GIVE_ME_MEMORY PROC NEAR           ;запрашивание памяти
        push   AX
        push   BX
        push   DX

        mov     DX,  OFFSET GIVE_MEM
        call    PRINTF

        mov     AH,  48h
        mov     BX,  1000h          ;1024*64/16 в 10
        int     21h

        jnc     exit_              ;CF=0
        mov     DX,  OFFSET ERROR
        call    PRINTF

exit_:

        pop     DX
        pop     BX
        pop     AX
        retn

GIVE_ME_MEMORY ENDP
;-----
MAIN:
        call    GET_A_MEM
        call    GIVE_ME_MEMORY

```

```

    call FREE_MEMORY
    call    GET_E_MEM
    call GET_MCB

;ВЫХОД В DOS
    sub     AL, AL
    mov     AH, 4Ch
    int     21h
END_PR    DB    0
CODE      ENDS
END       START

```