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

МСВ имеет следующую структуру:

Смещение	Длина поля (байт)	Содержимое поля	
00h	1	тип МСВ:	
		5Ah, если последний в списке,	
		4Dh, если не последний	
01h	2	Сегментный адрес PSP владельца участка памяти,	
		либо	
		0000h - свободный участок,	
		0006h - участок принадлежит драйверу	
		OS XMS UMB	
		0007h - участок является исключенной верхней	
		памятью драйверов	
		0008h - участок принадлежит MS DOS	
		FFFAh - участок занят управляющим блоком	
		386MAX UMB	
		FFFDh - участок заблокирован 386MAX	
		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 МСВ

Адрес первого МСВ 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:ВХ будет указывать на список списков. Слово по адресу ES:[ВХ-2] и есть адрес самого первого МСВ.

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

# 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.
CET WAD DEC	Процедура перевода слова в
GET_WRD_DEC	десятичную систему счисления.
PRINTF	Процедура печати
	Процедура, используемая для
GET_A_MEM	определения количества доступной
GDT_TI_IVIENT	памяти и вывода этого значения на
	экран.
	Процедура, используемая для
GET E MEM	определения размера расширенной
GLT_L_IVILIVI	памяти и вывода этого значения на
	экран.
	Процедура, используемая для вывода
START_	типа МСВ и "разделения" блоков при
	выводе на экран.
	Процедура, необходимая для
GET_ADRESS	определения адреса МСВ 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 МСВ.
GET_MCB	Процедура для вызова START_, GET_ADRESS, GET_SIZE_MEM,
	GET_MCB для всех блоков МСВ.
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 A, тестирование модуля см. на рис. 2 и 3.

```
Z:\>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

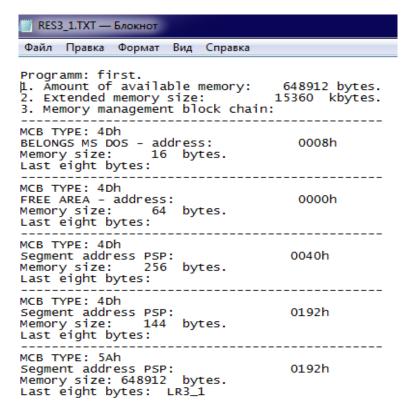


Рисунок 3 - Тестирование LR3\_1.COM

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

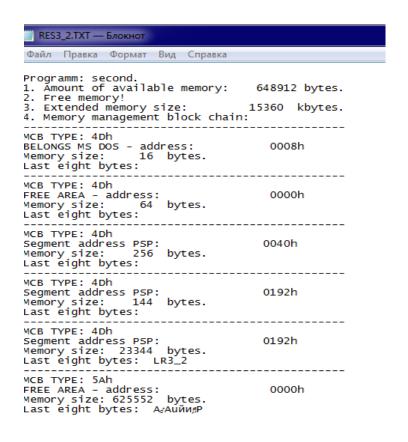


Рисунок 5 - Тестирование LR3\_2.COM

Программа была изменена так, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 4Аh прерывания 21Н. 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:\>exeZbin lr3_3.exe lr3_3.com

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

Рисунок 6 - Запуск LR3\_3.com

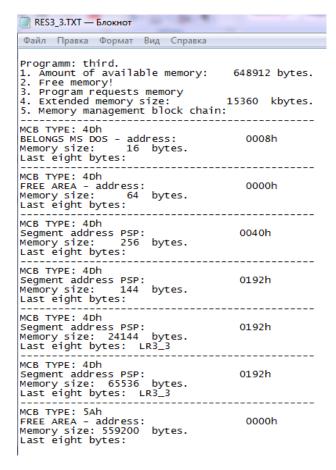


Рисунок 7 - Тестирование LR3\_3.com

Программа была изменена таким образом, чтобы запрос 64Кб памяти осуществлялся до ее освобождения. Проверяется флаг СF, 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 Γ, 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:\mathref{C:\mathref{Simple}} linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

LINK: warning L4021: no stack segment

C:\mathref{C:\mathref{Simple}} lr3_4.exe lr3_4.com

C:\mathref{C:\mathref{Simple}} lr3_4.com

C:\mathref{Simple} lr3_4.com

C:\mathref{Simple} lr3_4.com
```

Рисунок 8 - Запуск LR3\_4.COM

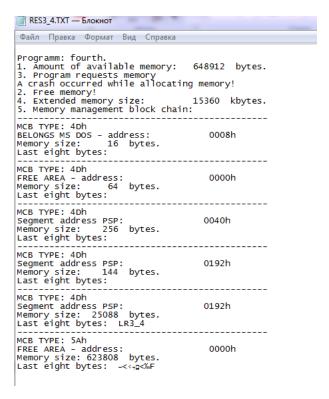


Рисунок 9 - Тестирование LR3\_4.COM

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

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

- Первая программа занимает 649 056 байт, то есть все доступную ей память.
- Вторая программа занимает 23 488 байта, так как неиспользуемая память была освобождена.
- Третья программа занимает 89 824 (24144 + 65536 + 144) байт, из них 65536 были запрошены дополнительно, а 24288 непосредственно используются программой.
- Четвертая программа занимает необходимые 25 232 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
                      ODH, OAH, OAH, 'Programm: first.',
                 DB
   EOF
AVAILABLE MEM DB
                 0DH, '1. Amount of available memory:
                                                              bytes.',
                 ;количество доступной памяти
   EOF
                 ODH, OAH, OAH, '2.
EXTENDED MEM DB
                                             Extended memory size:
   kbytes.', EOF
                        ;размер расширенной памяти
MCB
                  DB
                      ODH, OAH, '3. Memory management block chain:',
  EOF
DELIMITER
                 DB
                     ODH, OAH, '-----
  ----',
               EOF
TYPE1
                                     'MCB
                                            TYPE:
                DB
                        0DH,
                               ØΑH,
                                                    4Dh
   EOF
TYPE2
                DB
                        0DH,
                              OAH, 'MCB
                                            TYPE:
                                                    5Ah
   EOF
               ;5Ah - last
```

```
DB 0DH, 0AH, 'FREE AREA - address:
CASE1
  0000h',
              EOF
              DB 0DH, 0AH, 'BELONGS DRIVER OS XMS UMB -
CASE2
                       EOF
  address:0006h',
                        OAH, 'TOP MEMORY DRIVER - address:
CASE3
               DB ØDH,
  0007h',
                EOF
              DB 0DH, 0AH, 'BELONGS MS DOS - address:
CASE4_
  0008h',
                EOF
               DB ODH, OAH, 'CONTROL UNIT 386MAX UMB - address:
CASE5
  FFFAh',
               EOF
              DB 0DH, 0AH, 'BLOCKED 386MAX - address:
CASE6
  FFFDh',
               EOF
              DB 0DH, 0AH, 'BELONGS 386MAX UMB - address:
CASE7
 FFFEh',
              EOF
                                   'Segment address PSP:
              DB 0DH, 0AH,
CASE
  h',
              EOF
          DB
              ODH, OAH, 'Memory size:
                                                     bytes.',
SIZE MEM
  EOF
LAST
              DB
                   ODH, OAH, 'Last eight bytes:
  EOF
;ПРОЦЕДУРЫ
TETR TO HEX
             PROC NEAR
               and AL, 0Fh
                   AL, 09
               cmp
                    NEXT
               jbe
```

add

AL, 07

```
NEXT:
               add AL, 30h
               ret
         ENDP
TETR_TO_HEX
;-----
BYTE_TO_HEX
              PROC NEAR
       ;байт в AL переводится в два символа шестн. числа в АХ
               push
                      \mathsf{CX}
               mov AH, AL
               call TETR_TO_HEX
               xchg AL, AH
               mov CL, 4
               shr
                    AL, CL
                     TETR_TO_HEX
               call
           ;в AL - старшая цифра
                      \mathsf{CX}
               pop
           ;в АН - младшая
               ret
BYTE_TO_HEX
               ENDP
;-----
WRD_TO_HEX
               PROC NEAR
       ;перевод в 16 с/с 16-ти разрядного числа
               ;в АХ - число, DI - адрес последнего символа
               push BX
                        BH, AH
               mov
               call BYTE_TO_HEX
                        [DI], AH
               mov
               dec
                        DΙ
               mov
                       [DI], AL
                       DI
               dec
```

AL, BH

call BYTE TO HEX

mov

```
[DI], AH
                    mov
                                DΙ
                    dec
                                [DI], AL
                    mov
                                BX
                    pop
                    ret
WRD_TO_HEX
                    ENDP
BYTE_TO_DEC
                    PROC NEAR
                    push CX
                    push DX
                    xor AH, AH
                    xor DX,DX
                         CX,10
                    mov
loop_bd:
                    div
                               CX
                    or
                               DL,30h
   ; перевод в ascii
                         [SI],DL
                    mov
                    dec
                          SI
                    xor
                         DX,DX
                         AX,10
                    cmp
                    jae
                          loop_bd
                          AL,00h
                    cmp
                    je
                               \mathsf{end} \mathsf{\_1}
                               AL,30h
                    or
                         [SI],AL
                    mov
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
                        DL, 30h
                or
  ; перевод в ascii
                mov [SI], DL
                dec
                    SI
                xor DX, DX
                    AX, 10
                cmp
                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 ВХ 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
                     SI
                push
                          SI, OFFSET EXTENDED_MEM
                mov
                      SI, OFFSET SETPRECISION
               add
                         DX, DX
                xor
                         AL, 30h
                                       ;запись адреса ячейки CMOS
                mov
                         70h, AL
                out
                       AL, 71h ;чтение младшего байта
                in
                       BL, AL
                mov
                                        ;размер расширенной памяти
                       AL, 31h
                                        ;запись адреса ячейки CMOS
                mov
                out
                       70h, AL
                       AL, 71h
                in
                                       ;чтение старшего байта
                       AH, AL
                                       ;размер расширенной памяти
                mov
                       Al, BL
                mov
                       GET_WRD_DEC ;перевод в 10 с с
                call
                mov
                       DX, OFFSET EXTENDED_MEM
                call PRINTF
                       SI
                pop
                       DX
                pop
                       BX
                pop
```

AX

pop

retn GET\_E\_MEM **ENDP** ;-----START\_ PROC NEAR push DX push AX push CX push SI mov DX, OFFSET DELIMITER call PRINTF AX, 4Dh ;печать типа cmpcout jne DX, OFFSET TYPE1 mov exit jmp cout: mov DX, OFFSET TYPE2 exit: call PRINTF SI pop pop  $\mathsf{CX}$ AX pop DX pop retn START\_ **ENDP** GET\_ADRESS PROC NEAR push AX push ES

mov AX, ES:[01h]

push DX

push DI

AX, 0000h cmpjne case2 DX, OFFSET CASE1\_ mov write jmp case2: AX, 0006h cmpjne case3 DX, OFFSET CASE2\_ movwrite jmp case3: AX, 0007h cmpjne case4 mov DX, OFFSET CASE3\_ write jmp case4: AX, 0008h cmpjne case5 DX, OFFSET CASE4\_ mov jmp write case5: cmpAX, 0FFFAh jne case6 DX, OFFSET CASE5\_ movwrite jmp case6: AX, 0FFFDh cmpjne case7 DX, OFFSET CASE6\_ mov write jmp case7: AX, 0FFFEh cmpjne case0 DX, OFFSET CASE7\_ mov

```
write
                jmp
case0:
                        DI, DI
                xor
                       DX, DX
               xor
                       DI, OFFSET CASE_
                mov
              add DI, OFFSET SETPRECISION
                call WRD_TO_HEX
                       DX, OFFSET CASE_
                mov
write:
                call PRINTF
end_a:
                   DI
               pop
                   DX
                pop
                pop
                   ES
                pop AX
                retn
GET ADRESS
               ENDP
;-----
GET_SIZE_MEM PROC NEAR
                push SI
                push BX
               push ES
               push AX
               push DX
                        SI, OFFSET SIZE_MEM
               mov
              add SI, OFFSET SETPR
                      BX, ES:[03h] ;размер участка в параграфах -
               mov
  перевести в 10 с/с
                       AX, 10h
                mov
                        ВХ
               mul
                                          ;параграф
                      GET WRD DEC ;перевод в 10 с с
               call
```

```
mov DX, OFFSET SIZE_MEM
               call PRINTF
                      DX
               pop
               pop
                       AX
               pop
                       ES
                      BX
               pop
               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
                  DI, 0
               mov
                     CX, 8
               mov
symbol:
               xor AX, AX
                   AH, 02h
               mov
                  DL, ES:[08h + DI] ;08h - там лежат символы
               mov
               int
                   21h
                  DI
               inc
               loop symbol
                   ES
               pop
               pop
                     AX
```

```
CX
               pop
               pop
                      DI
                   DX
               pop
               retn
               ENDP
GET SYMBOLS
;-----
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
                   AX, ES:[BX - 02h] ;получение адреса
               mov
                   ES, AX
               mov
                                         ;первого МСВ
replay:
                       AX, AX
               xor
                       AL, ES:[00h] ;type MCB
               mov
               push AX
               call START_
               call GET_ADRESS
               call GET_SIZE_MEM
               call GET_SYMBOLS
                       AX
               pop
                       AX, 4Dh
               cmp
```

end MCB

jne

```
АХ, ES:[03h] ;размер участка в параграфах
               mov
                        BX, ES
                mov
                                             ;адрес начала
                      BX, AX
                add
                                        ;адрес конца
                         BX
                inc
                                             ;в ВХ - адрес след
  элемента
                        ES, BX
               mov
                jmp
                        replay
end_MCB:
                        BX
               pop
                        ES
               pop
                       AX
               pop
                        DX
               pop
                retn
GET_MCB
               ENDP
;-----
MAIN:
       call GET A MEM
       call GET_E_MEM
       call GET_MCB
      ;выход в DOS
       sub
              AL, AL
               AH, 4Ch
       mov
       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
                       ODH, OAH, OAH, 'Programm: second.
                  DB
   EOF
AVAILABLE MEM DB
                  ODH, '1. Amount of available memory:
                                                               bytes.',
   EOF
                  ;количество доступной памяти
FR MEM
                  DB ODH, '2. Free memory!',
                  ;количество доступной памяти
   EOF
EXTENDED MEM DB
                  ODH, OAH, OAH, '3. Extended memory size:
   kbytes.', EOF
                          ;размер расширенной памяти
MCB
               DB 0DH, 0AH, 0AH, '4. Memory management block chain:',
   EOF
                  DB 0DH, 0AH, '-----
DELIMITER
   ----',
                EOF
TYPE1
                 DB
                    ODH, OAH, 'MCB TYPE: 4Dh',
   EOF
```

```
TYPE2
              DB 0DH, 0AH, 'MCB TYPE: 5Ah ',
  EOF
              ;5Ah - last
CASE1
                 DB ODH, OAH, 'FREE AREA - address:
                  EOF
  0000h',
CASE2
                 DB ODH, OAH, 'BELONGS DRIVER OS XMS UMB -
  address:0006h',
                         EOF
CASE3_
                 DB ODH, OAH, 'TOP MEMORY DRIVER - address:
  0007h',
                  EOF
                 DB ODH, OAH, 'BELONGS MS DOS - address:
CASE4
  0008h',
                 EOF
CASE5
                 DB ODH, OAH, 'CONTROL UNIT 386MAX UMB - address:
  FFFAh',
                  EOF
CASE6
                 DB ODH, OAH, 'BLOCKED 386MAX - address:
                  EOF
  FFFDh',
                 DB ODH, OAH, 'BELONGS 386MAX UMB - address:
CASE7
  FFFEh',
                  EOF
                 DB 0DH, 0AH, 'Segment address PSP:
CASE
  h',
                EOF
SIZE_MEM DB 0DH, 0AH, 'Memory size: bytes.',
  EOF
                 DB ODH, OAH, 'Last eight bytes: ',
LAST
  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 АХ
               push
                     \mathsf{CX}
               mov AH, AL
               call
                    TETR_TO_HEX
                    AL, AH
               xchg
               mov CL, 4
               shr AL, CL
               call TETR_TO_HEX
           ;в AL - старшая цифра
               pop
                   CX
           ;в АН - младшая
               ret
BYTE_TO_HEX
               ENDP
;-----
               PROC NEAR
WRD_TO_HEX
      ;перевод в 16 с/с 16-ти разрядного числа
               ;в АХ - число, DI - адрес последнего символа
               push BX
               mov
                        BH, AH
               call BYTE_TO_HEX
                       [DI], AH
               mov
                       DΙ
               dec
                       [DI], AL
               mov
```

```
dec
                DI
                AL, BH
             mov
             call BYTE_TO_HEX
             mov
                   [DI], AH
                    DI
             dec
             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
                [SI], DL
             mov
                SI
             dec
             xor
                DX, DX
                AX, 10
             cmp
                 loop_bd
             jae
                AL, 00h
             cmp
                end l
             je
             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
                     DL, 30h ; перевод в ascii
              or
              mov [SI], DL
                 SI
              dec
              xor DX, DX
              стр АХ, 10 ; если в целой части есть, что еще
  делить
              jae loop_wd
              cmp AL, 00h
                 end wb
              je
              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
                   DX, OFFSET PROG
                mov
                call PRINTF
                xor
                        DX, DX
                         SI, OFFSET AVAILABLE_MEM
                mov
                     SI, OFFSET SETPRECISION
               add
                        AH, 04Ah
                                     ;изменить размер блока память
                mov
                        BX, 0FFFFh ;заведомо большая память
                mov
                int
                         21h
                                          ;в BX будет лежать
  максимальный размер доступный для этого блока
                        AX, 10h
                mov
                         BX
                                          ;параграф
                mul
                      GET_WRD_DEC ;перевод в 10 с с
                call
                      DX, OFFSET AVAILABLE_MEM
                mov
```

call PRINTF

AX

pop

pop

pop

pop

SI

BX

DX

retn GET\_A\_MEM **ENDP** ;-----GET\_E\_MEM PROC NEAR ;2 ЗАДАНИЕ - РАСПЕЧАТАТЬ РАЗМЕР РАСШИРЕННОЙ ПАМЯТИ В 10 С.С. push AXpush BX DX push SI push SI, OFFSET EXTENDED\_MEM mov add SI, OFFSET SETPRECISION DX, DX xor AL, 30h ;запись адреса ячейки CMOS mov 70h, AL out AL, 71h ;чтение младшего байта in BL, AL mov ;размер расширенной памяти AL, 31h ;запись адреса ячейки CMOS mov 70h, AL out in AL, 71h ;чтение старшего байта AH, AL ;размер расширенной памяти mov Al, BL movGET WRD DEC ;перевод в 10 с с call DX, OFFSET EXTENDED\_MEM mov call PRINTF pop SI DX pop

BX

AX

pop

pop

retn GET\_E\_MEM **ENDP** ;-----START\_ PROC NEAR push DX push AX push CX push SI mov DX, OFFSET DELIMITER call PRINTF cmpAX, 4Dh ;печать типа jne cout DX, OFFSET TYPE1 mov exit jmp cout: mov DX, OFFSET TYPE2 exit: call PRINTF SI pop  $\mathsf{CX}$ pop AX pop DX pop retn START\_ **ENDP** ;-----GET\_ADRESS PROC NEAR push AX push ES push DX push DI

AX, ES:[01h] mov AX, 0000h cmpcase2 jne DX, OFFSET CASE1\_ mov write jmp case2: AX, 0006h cmpjne case3 DX, OFFSET CASE2\_ mov write jmp case3: cmpAX, 0007h case4 jne DX, OFFSET CASE3\_ mov write jmp case4: AX, 0008h cmpjne case5 DX, OFFSET CASE4\_ mov write jmp case5: AX, 0FFFAh cmpjne case6 DX, OFFSET CASE5\_ movwrite jmp case6: AX, 0FFFDh cmpjne case7 DX, OFFSET CASE6\_ mov write jmp case7:

cmp

AX, ØFFFEh

```
jne
                        case0
                 mov
                        DX, OFFSET CASE7_
                          write
                 jmp
case0:
                          DI, DI
                 xor
                 xor
                          DX, DX
                          DI, OFFSET CASE_
                 mov
                       DI, OFFSET SETPRECISION
                add
                 call WRD_TO_HEX
                          DX, OFFSET CASE_
                 mov
write:
                 call PRINTF
end_a:
                 pop
                          DΙ
                 pop
                          DX
                      ES
                 pop
                 pop
                     AX
                 retn
GET_ADRESS
                 ENDP
;-----
GET_SIZE_MEM PROC NEAR
                 push SI
                 push BX
                 push ES
                 push AX
                 push DX
                          SI, OFFSET SIZE_MEM
                 mov
                       SI, OFFSET SETPR
                add
                        BX, ES:[03h] ;размер участка в параграфах -
                 mov
  перевести в 10 с/с
```

```
mov AX, 10h
                      \mathsf{BX}
                                   ;параграф
              mul
              call GET_WRD_DEC ;перевод в 10 с с
              mov DX, OFFSET SIZE_MEM
              call PRINTF
                     DX
              pop
              pop
                      AX
                  ES
              pop
              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
                  CX, 8
              mov
symbol:
              xor AX, AX
                      AH, 02h
              mov
                      DL, ES:[08h + DI] ;08h - там лежат символы
              mov
              int
                  21h
              inc DI
              loop symbol
```

```
ES
               pop
                     AX
               pop
                     CX
               pop
                       DI
               pop
               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
                   AX, ES:[BX - 02h] ;получение адреса
               mov
                  ES, AX
                                       ;первого МСВ
               mov
replay:
               xor AX, AX
                  AL, ES:[00h] ;type MCB
               mov
               push AX
               call START_
               call GET ADRESS
               call GET SIZE MEM
               call GET SYMBOLS
```

```
\mathsf{AX}
                  pop
                           AX, 4Dh
                  cmp
                  jne
                            end_MCB
                         AX, ES:[03h] ;размер участка в параграфах
                  mov
                            BX, ES
                  mov
                                                     ;адрес начала
                         BX, AX
                  add
                                                ;адрес конца
                            BX
                  inc
                                                     ;в ВХ - адрес след
   элемента
                            ES, BX
                  mov
                            replay
                  jmp
end_MCB:
                            BX
                  pop
                  pop
                            ES
                            AX
                  pop
                            DX
                  pop
                  retn
GET MCB
                  ENDP
;-----
FREE_MEMORY
                  PROC NEAR
                                          ;освобождение памяти
                  push AX
                  push BX
                  push DX
                  mov
                         DX, OFFSET FR_MEM
                  call PRINTF
                           AH, 4Ah
                  mov
                            BX, OFFSET END_PR
                  mov
                  int
                       21h
                         DX
                  pop
                            BX
                  pop
                  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
                  JMP
START:
                               MAIN
                               '$'
EOF
                     EOU
SETPRECISION
               EQU
                          40
               EQU
                       20
SETPR
;ДАННЫЕ
PROG
               DB
                    ODH, OAH, OAH, 'Programm: third.',
EOF
               DB
                    ODH, '1. Amount of available memory:
AVAILABLE MEM
bytes.',
                  EOF
                                    ;количество доступной памяти
                    DB ODH, '2. Free memory!',
FR MEM
EOF
               ;количество доступной памяти
               DB 0DH, '3. Program requests memory',
GIVE MEM
EOF
                    ODH, OAH, OAH, '4. Extended memory size:
EXTENDED MEM
               DB
kbytes.', EOF
                       ;размер расширенной памяти
MCB
               DB
                   ODH, OAH, '5. Memory management block chain:',
EOF
               DB 0DH, 0AH, '-----
DELIMITER
----',
             EOF
                  ODH, OAH, 'MCB TYPE: 4Dh',
TYPE1
              DB
EOF
                  ODH, OAH, 'MCB TYPE: 5Ah ',
TYPE2
              DB
             ;5Ah - last
EOF
                          ODH, OAH, 'FREE AREA - address:
CASE1
                     DB
0000h',
                  EOF
                    DB
                          ODH, OAH, 'BELONGS DRIVER OS XMS UMB -
CASE2
address:0006h',
                          EOF
                          ODH, OAH, 'TOP MEMORY DRIVER - address:
CASE3
                    DB
0007h',
                  EOF
                          ODH, OAH, 'BELONGS MS DOS - address:
CASE4
                    DB
0008h',
                  EOF
CASE5
                          ODH, OAH, 'CONTROL UNIT 386MAX UMB - address:
                    DB
FFFAh',
                  EOF
```

```
CASE6
                 DB
                      ODH, OAH, 'BLOCKED 386MAX - address:
FFFDh',
               EOF
CASE7
                 DB
                      ODH, OAH, 'BELONGS 386MAX UMB - address:
FFFEh',
               EOF
CASE_
             DB
                 ODH, OAH, 'Segment address PSP:
h',
            EOF
SIZE_MEM
                 ODH, OAH, 'Memory size: bytes.',
             DB
EOF
LAST
             DB
                 ODH, OAH, 'Last eight bytes: ',
EOF
;ПРОЦЕДУРЫ
;-----
                 PROC NEAR
TETR TO HEX
                 and
                       AL, 0Fh
                       AL, 09
                 cmp
                 ibe
                      NEXT
                 add AL, 07
NEXT:
             add
                 AL, 30h
                 ret
TETR_TO_HEX
            ENDP
;-----
                 PROC NEAR
BYTE TO HEX
        ;байт в AL переводится в два символа шестн. числа в АХ
                 push
                        \mathsf{CX}
                 mov
                        AH, AL
                        TETR_TO_HEX
                 call
                        AL, AH
                 xchg
                        CL, 4
                 mov
                 shr
                        AL, CL
                 call
                        TETR_TO_HEX
             ;в AL - старшая цифра
                        \mathsf{CX}
                 pop
             ;в АН - младшая
                 ret
BYTE TO HEX
                 ENDP
WRD_TO_HEX PROC NEAR
    ;перевод в 16 с/с 16-ти разрядного числа
                 ;в АХ - число, DI - адрес последнего символа
                 push BX
                          BH, AH
                 mov
                 call BYTE_TO_HEX
```

```
[DI], AH
                  mov
                  dec
                            DΙ
                  mov
                            [DI], AL
                  dec
                            DΙ
                            AL, BH
                  mov
                  call BYTE_TO_HEX
                            [DI], AH
                  mov
                  dec
                            DΙ
                            [DI], AL
                  mov
                            BX
                  pop
                  ret
WRD_TO_HEX
              ENDP
;-----
BYTE_TO_DEC
                  PROC NEAR
                  push CX
                  push DX
                  xor
                       AH, AH
                       DX,DX
                  xor
                  mov
                       CX,10
loop_bd:
                  div
                            \mathsf{CX}
                            DL,30h
                  or
                                      ; перевод в ascii
                       [SI],DL
                  mov
                  dec
                       SI
                  xor
                       DX,DX
                  cmp
                       AX,10
                  jae
                       loop_bd
                  cmp
                       AL,00h
                  je
                           end l
                  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
                       CX, 10
                  mov
loop_wd:
                  div
                            \mathsf{CX}
                            DL, 30h
                                    ; перевод в ascii
                  or
                       [SI], DL
                  mov
                       SI
                  dec
                  xor
                       DX, DX
                       AX, 10
                  cmp
```

```
jae loop_wd
                 cmp
                     AL, 00h
                         end_wb
                 je
                         AL, 30h
                 or
                 mov [SI], AL
end_wb:
                 pop DX
                 pop CX
                 retn
             ENDP
GET_WRD_DEC
;-----
             PROC NEAR
PRINTF
             push AX
             mov
                 AH, 09h
             int
                 21h
             pop
                  AX
             retn
PRINTF
             ENDP
;-----
                        ;1 ЗАДАНИЕ - РАСПЕЧАТАТЬ КОЛИЧЕСТВО
           PROC NEAR
GET_A_MEM
ДОСТУПНОЙ ПАМЯТИ В 10 С.С.
                 push DX
                 push AX
                 push BX
                 push SI
                 mov
                      DX, OFFSET PROG
                 call PRINTF
                 xor
                         DX, DX
                          SI, OFFSET AVAILABLE_MEM
                 mov
                add
                      SI, OFFSET SETPRECISION
                         AH, 04Ah
                                      ;изменить размер блока
                 mov
память
                          BX, 0FFFFh ;заведомо большая память
                 mov
                 int
                          21h
                                           ;в ВХ будет лежать
максимальный размер доступный для этого блока
                 mov
                          AX, 10h
                 mul
                          BX
                                           ;параграф
                 call GET_WRD_DEC ;перевод в 10 с с
                       DX, OFFSET AVAILABLE_MEM
                 mov
                 call PRINTF
                       SI
                 pop
```

```
BX
                   pop
                   pop
                       AX
                            DX
                   pop
                   retn
GET A MEM
              ENDP
;-----
GET E MEM
              PROC NEAR
                                ;2 ЗАДАНИЕ - РАСПЕЧАТАТЬ РАЗМЕР
РАСШИРЕННОЙ ПАМЯТИ В 10 С.С.
                   push
                         AX
                          BX
                   push
                         DX
                   push
                         SI
                   push
                            SI, OFFSET EXTENDED_MEM
                   mov
                        SI, OFFSET SETPRECISION
                 add
                            DX, DX
                   xor
                            AL, 30h
                   mov
                                       ;запись адреса ячейки
CMOS
                   out
                           70h, AL
                          AL, 71h
                                     ;чтение младшего байта
                   in
                          BL, AL
                                         ;размер расширенной
                   mov
памяти
                         AL, 31h
                   mov
                                          ;запись адреса ячейки
CMOS
                          70h, AL
                   out
                   in
                         AL, 71h
                                          ;чтение старшего байта
                         AH, AL
                                          ;размер расширенной
                   mov
памяти
                         Al, BL
                   mov
                   call
                         GET_WRD_DEC ;перевод в 10 с с
                          DX, OFFSET EXTENDED_MEM
                   mov
                   call PRINTF
                          SI
                   pop
                          DX
                   pop
                          BX
                   pop
                         AX
                   pop
                   retn
GET_E_MEM
              ENDP
;-----
START_
                   PROC NEAR
                   push
                          DX
                   push AX
                   push
                          \mathsf{CX}
                          SI
                   push
```

mov DX, OFFSET DELIMITER call PRINTF AX, 4Dh cmp;печать типа jne cout DX, OFFSET TYPE1 mov exit jmp cout: DX, OFFSET TYPE2 mov exit: call PRINTF SI pop CX pop AXpop pop DX retn START **ENDP** ;-----GET ADRESS PROC NEAR push AX push ES push DX push DI AX, ES:[01h] mov cmpAX, 0000h jne case2 mov DX, OFFSET CASE1\_ jmp write case2: cmpAX, 0006h jne case3 DX, OFFSET CASE2\_ mov jmp write case3: cmpAX, 0007h case4 jne DX, OFFSET CASE3\_ mov write jmp case4: AX, 0008h cmpjne case5 DX, OFFSET CASE4\_ mov write jmp case5: cmpAX, 0FFFAh

DX, OFFSET CASE5\_

case6

jne

mov

```
jmp
                              write
case6:
                    cmp
                              AX, 0FFFDh
                    jne
                           case7
                           DX, OFFSET CASE6_
                    mov
                    jmp
                              write
case7:
                              AX, 0FFFEh
                    cmp
                    jne
                           case0
                           DX, OFFSET CASE7_
                    mov
                              write
                    jmp
case0:
                              DI, DI
                    xor
                              DX, DX
                    xor
                              DI, OFFSET CASE_
                    mov
                          DI, OFFSET SETPRECISION
                   add
                    call WRD_TO_HEX
                              DX, OFFSET CASE
write:
                    call PRINTF
end a:
                    pop
                              DI
                              DX
                    pop
                    pop
                         ES
                    pop
                        AX
                    retn
GET_ADRESS
               ENDP
;-----
               PROC NEAR
GET SIZE MEM
                    push SI
                    push BX
                    push ES
                    push AX
                    push DX
                    mov
                              SI, OFFSET SIZE_MEM
                          SI, OFFSET SETPR
                   add
                           BX, ES:[03h] ;размер участка в параграфах -
                    mov
перевести в 10 с/с
                              AX, 10h
                    mov
                    mul
                              BX
                                                  ;параграф
                    call
                           GET_WRD_DEC ;перевод в 10 с с
                           DX, OFFSET SIZE_MEM
                    call PRINTF
                              DX
                    pop
                              AX
                    pop
```

```
ES
                  pop
                  pop
                           BX
                           DX
                  pop
                  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
                          CX, 8
                  mov
symbol:
                      AX, AX
                  xor
                           AH, 02h
                  mov
                           DL, ES:[08h + DI] ;08h - там лежат
                  mov
символы
                  int
                      21h
                  inc
                           DΙ
                  loop symbol
                  pop
                           ES
                          AX
                  pop
                  pop
                           \mathsf{CX}
                           DΙ
                  pop
                           DX
                  pop
                  retn
                  ENDP
GET_SYMBOLS
;-----
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
                      21h
                                             ;list of lists
                  int
                      АХ, ES:[BX - 02h] ;получение адреса
                  mov
                     ES, AX
                                             ;первого МСВ
                  mov
replay:
```

```
AX, AX
                 xor
                          AL, ES:[00h] ;type MCB
                 mov
                 push AX
                 call START_
                 call GET_ADRESS
                 call GET_SIZE_MEM
                 call GET_SYMBOLS
                          AX
                 pop
                          AX, 4Dh
                 cmp
                          end_MCB
                 jne
                      АХ, ES:[03h] ;размер участка в
                 mov
параграфах
                         BX, ES
                 mov
                                               ;адрес начала
                      BX, AX
                 add
                                           ;адрес конца
                 inc
                          BX
                                                ;в ВХ - адрес
след элемента
                 mov
                         ES, BX
                 jmp
                          replay
end_MCB:
                          BX
                 pop
                          ES
                 pop
                 pop
                          AX
                          DX
                 pop
                 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
                         BX, OFFSET END_PR
                 mov
                 int 21h
                        DX
                 pop
                          BX
                 pop
                          AX
                 pop
                 retn
FREE_MEMORY
                 ENDP
;-----
GIVE ME MEMORY PROC NEAR
                              ;запрашивание памяти
                 push AX
                 push BX
```

```
push DX
                 mov DX, OFFSET GIVE_MEM
                 call PRINTF
                     AH, 48h
                 mov
                         BX, 1000h ;1024*64/16 в 10
                 mov
                 int
                      21h
                 pop
                          DX
                          BX
                 pop
                          AX
                 pop
                 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
                 AL, AL
        sub
                 AH, 4Ch
        mov
                 21h
        int
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
                       ODH, OAH, OAH, 'Programm: fourth.
                  DB
   EOF
AVAILABLE MEM DB ODH, OAH, '1. Amount of available memory:
   bytes.',
                 EOF
                                 ;количество доступной памяти
FR MEM
                 DB ODH, OAH, '2. Free memory!',
                  ;количество доступной памяти
   EOF
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
              DB 0DH, 0AH, '5. Memory management block chain:',
MCB
   EOF
                  DB 0DH, 0AH, '-----
DELIMITER
   ----',
                EOF
```

```
TYPE1
               DB ODH, OAH, 'MCB TYPE: 4Dh ',
  EOF
TYPE2
              DB ODH, OAH, 'MCB TYPE: 5Ah ',
  EOF
              ;5Ah - last
CASE1
                DB ODH, OAH, 'FREE AREA - address:
  0000h',
                 EOF
CASE2
                DB ODH, OAH, 'BELONGS DRIVER OS XMS UMB -
  address:0006h',
                         EOF
                DB ODH, OAH, 'TOP MEMORY DRIVER - address:
CASE3
  0007h',
                EOF
CASE4
                DB ODH, OAH, 'BELONGS MS DOS - address:
  0008h',
                 EOF
CASE5
                DB ODH, OAH, 'CONTROL UNIT 386MAX UMB - address:
  FFFAh',
                 EOF
                DB ODH, OAH, 'BLOCKED 386MAX - address:
CASE6
  FFFDh',
                 EOF
                DB ODH, OAH, 'BELONGS 386MAX UMB - address:
CASE7
  FFFEh',
                EOF
CASE_
               DB ODH, OAH, 'Segment address PSP:
  h',
               EOF
SIZE_MEM DB 0DH, 0AH, 'Memory size: bytes.',
  EOF
LAST
                DB
                    ODH, OAH, 'Last eight bytes: ',
 EOF
                DB ODH, OAH, 'A crash occurred while allocating
ERROR
                    EOF
 memory!',
ENDL
                DB 0DH, 0AH, EOF
;ПРОЦЕДУРЫ
;-----
```

```
PROC NEAR
TETR TO HEX
              and
                  AL, 0Fh
                   AL, 09
              cmp
              jbe
                   NEXT
                 AL, 07
              add
              add AL, 30h
NEXT:
              ret
TETR TO HEX ENDP
;-----
BYTE TO HEX
             PROC NEAR
      ;байт в AL переводится в два символа шестн. числа в АХ
              push CX
              mov AH, AL
              call TETR_TO_HEX
              xchg AL, AH
              mov CL, 4
                   AL, CL
              shr
                   TETR_TO_HEX
              call
          ;в AL - старшая цифра
              pop
                    \mathsf{CX}
          ;в АН - младшая
              ret
BYTE_TO_HEX
              ENDP
;-----
WRD_TO_HEX
             PROC NEAR
      ;перевод в 16 с/с 16-ти разрядного числа
```

;в АХ - число, DI - адрес последнего символа

```
push BX
                         BH, AH
                mov
                call BYTE_TO_HEX
                mov
                         [DI], AH
                         DI
                dec
                mov
                        [DI], AL
                         DI
                dec
                        AL, BH
                mov
                call BYTE_TO_HEX
                         [DI], AH
                \text{mov}
                         DI
                dec
                       [DI], AL
                mov
                         BX
                pop
                ret
WRD_TO_HEX
                ENDP
;-----
                PROC NEAR
BYTE_TO_DEC
                push CX
                push DX
                xor AH, AH
                xor DX,DX
                mov CX, 10
loop_bd:
                       CX
                div
                or
                         DL,30h
                                       ; перевод в ascii
                    [SI],DL
                mov
                    SI
                dec
                    DX,DX
                xor
                    AX,10
                cmp
                jae
                    loop_bd
                    AL,00h
                cmp
                        end l
                je
                        AL,30h
                or
                                    ; перевод в ascii
                mov
                     [SI],AL
```

```
end 1:
                pop DX
                pop CX
                ret
BYTE_TO_DEC ENDP
GET_WRD_DEC PROC NEAR ;us BYTE_TO_DEC
                push CX
                push DX
                mov CX, 10
loop_wd:
                div
                        CX
                        DL, 30h
                or
                                  ; перевод в ascii
                mov [SI], DL
                     SI
                dec
                xor DX, DX
                    АХ, 10 ;если в целой части есть, что еще
                cmp
  делить
                     loop_wd
                jae
                    AL, 00h ;если 0 заканчиваем
                cmp
                je
                        end_wb
                    AL, 30h ; перевод в ascii
                or
                mov [SI], AL
end_wb:
                pop DX
                pop CX
                retn
```

GET WRD DEC ENDP

```
PRINTF
            PROC NEAR
                 AX
            push
                 AH, 09h
            mov
                 21h
            int
                   AX
            pop
            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
                     DX, DX
                 xor
                          SI, OFFSET AVAILABLE_MEM
                 mov
                       SI, OFFSET SETPRECISION
                add
                          АН, 04Ah ;изменить размер блока память
                 mov
                          BX, 0FFFFh ;заведомо большая память
                 mov
                          21h
                 int
                                             ;в ВХ будет лежать
  максимальный размер доступный для этого блока
                          AX, 10h
                 mov
                 mul
                          BX
                                             ;параграф
                        GET WRD DEC ;перевод в 10 с с
                 call
```

DX, OFFSET AVAILABLE MEM

mov

```
call PRINTF
                    SI
                pop
                         BX
                pop
                pop
                    AX
                        DX
                pop
                retn
GET_A_MEM
                ENDP
;-----
GET_E_MEM
                PROC NEAR
                           ;2 ЗАДАНИЕ - РАСПЕЧАТАТЬ РАЗМЕР
  РАСШИРЕННОЙ ПАМЯТИ В 10 С.С.
                    AX
                push
                push
                    BX
                    DX
                push
                    SI
                push
                         SI, OFFSET EXTENDED MEM
                mov
               add SI, OFFSET SETPRECISION
                        DX, DX
                xor
                        AL, 30h
                                  ;запись адреса ячейки CMOS
                mov
                        70h, AL
                out
                      AL, 71h ;чтение младшего байта
                in
                      BL, AL
                mov
                                      ;размер расширенной памяти
                      AL, 31h
                                      ;запись адреса ячейки CMOS
                mov
                      70h, AL
                out
                     AL, 71h
                                      ;чтение старшего байта
                in
                      AH, AL
                                      ;размер расширенной памяти
                mov
                      Al, BL
                mov
```

call GET WRD DEC ;перевод в 10 с с

```
mov DX, OFFSET EXTENDED_MEM
                call PRINTF
                       SI
                pop
                    DX
                 pop
                 pop
                      BX
                      AX
                 pop
                 retn
GET_E_MEM
                ENDP
START_
                PROC NEAR
                push DX
                push AX
                push CX
                push SI
                mov DX, OFFSET DELIMITER
                 call PRINTF
                        AX, 4Dh
                 cmp
                                    ;печать типа
                jne
                         cout
                      DX, OFFSET TYPE1
                 mov
                       exit
                 jmp
cout:
                mov
                       DX, OFFSET TYPE2
exit:
                call PRINTF
                     SI
                pop
                       CX
                pop
                       AX
                 pop
                      DX
                 pop
                 retn
START
                ENDP
```

```
GET_ADRESS
                    PROC NEAR
                    push AX
                    push ES
                    push DX
                    push DI
                            AX, ES:[01h]
                    mov
                               AX, 0000h
                    cmp
                    jne
                            case2
                            DX, OFFSET CASE1_
                    mov
                               write
                    jmp
case2:
                              AX, 0006h
                    cmp
                    jne
                            case3
                            DX, OFFSET CASE2_
                    mov
                               write
                    jmp
case3:
                    cmp
                               AX, 0007h
                    jne
                            case4
                    mov
                            DX, OFFSET CASE3_
                               write
                    jmp
case4:
                              AX, 0008h
                    cmp
                    jne
                            case5
                            DX, OFFSET CASE4_
                    mov
                               write
                    jmp
case5:
                              AX, 0FFFAh
                    cmp
                    jne
                            case6
                            DX, OFFSET CASE5_
                    mov
                               write
                    jmp
```

case6:

AX, 0FFFDh cmpjne case7 DX, OFFSET CASE6\_ mov write jmp case7: cmpAX, 0FFFEh case0 jne DX, OFFSET CASE7\_ mov write jmp case0: DI, DI xor DX, DX xor DI, OFFSET CASE\_ mov add DI, OFFSET SETPRECISION call WRD\_TO\_HEX DX, OFFSET CASE\_ mov write: call PRINTF end\_a: pop DI DX pop pop ES pop AX retn GET ADRESS **ENDP** GET\_SIZE\_MEM PROC NEAR push SI push BX push ES push AX push DX

```
SI, OFFSET SIZE_MEM
                mov
                      SI, OFFSET SETPR
               add
                       BX, ES:[03h] ;размер участка в параграфах -
                mov
  перевести в 10 с/с
                         AX, 10h
                mov
                mul
                          \mathsf{BX}
                                            ;параграф
                       GET_WRD_DEC ;перевод в 10 с с
                call
                       DX, OFFSET SIZE_MEM
                mov
                call PRINTF
                pop
                         DX
                pop
                         AX
                         ES
                pop
                pop
                          BX
                          DX
                pop
                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
                       DI, 0
                mov
                         CX, 8
                mov
symbol:
```

```
xor AX, AX
                      AH, 02h
               mov
                  DL, ES:[08h + DI] ;08h - там лежат символы
               mov
               int
                   21h
                   DI
               inc
               loop symbol
                      ES
               pop
               pop
                      AX
                  CX
               pop
               pop DI
               pop DX
               retn
GET SYMBOLS
               ENDP
;-----
              PROC NEAR
                         ;3 ЗАДАНИЕ - ВЫВОД ЦЕПОЧКИ БЛОКОВ
GET_MCB
  УПРАВЛЕНИЯ ПАМЯТЬЮ
               push DX
               push AX
               push ES
               push BX
               mov DX, OFFSET MCB
               call PRINTF
               mov AH, 52h
                                           ;доступ к указателю
  на структуру
                                       ;list of lists
               int 21h
                  AX, ES:[BX - 02h] ;получение адреса
               mov
                   ES, AX
               mov
                                       ;первого МСВ
replay:
                      AX, AX
               xor
                      AL, ES:[00h] ;type MCB
               mov
               push AX
```

```
call START_
                  call GET_ADRESS
                  call GET_SIZE_MEM
                  call GET SYMBOLS
                            AX
                  pop
                            AX, 4Dh
                  cmp
                            end_MCB
                  jne
                          АХ, ES:[03h] ;размер участка в параграфах
                  mov
                             BX, ES
                  mov
                                                     ;адрес начала
                          BX, AX
                                                ;адрес конца
                  add
                  inc
                             BX
                                                     ;в ВХ - адрес след
   элемента
                            ES, BX
                  mov
                            replay
                  jmp
end MCB:
                             BX
                  pop
                  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
```

AH, 4Ah

mov

```
BX, OFFSET END_PR
            mov
            int 21h
               DX
            pop
                    BX
            pop
            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
               BX, 1000h ;1024*64/16 в 10
            mov
            int 21h
            jnc exit_
                         ;CF=0
            mov DX, OFFSET ERROR
            call PRINTF
exit:
               DX
            pop
            pop BX
            pop AX
            retn
GIVE_ME_MEMORY
            ENDP
;-----
MAIN:
     call GET_A_MEM
```

call GIVE ME MEMORY

 ${\tt call \ FREE\_MEMORY}$ 

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