МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №3

по дисциплине «Операционные системы»

Тема: Исследование организации управления основной памятью

Студент гр. 8381 _____ Муковский Д.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованный в компьютере и способ организации, принятый в ОС. В лабораторной работе рассматривается нестраничная память и способ управления динамическими разделами. Для реализации управления памятью в этом случае строится список занятых и свободных участков памяти. Функции ядра, обеспечивающие управление основной памятью, просматривают и преобразуют этот список.

В лабораторной работе исследуются структуры данных и работа 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

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

Адрес первого MCB хранится во внутренней структуре MS DOS, называемой "List of Lists" (список списков). Доступ к указателю на эту структуру можно получить, используя функцию 52h "Get List of Lists" int 21h. В результате выполнения этой функции ES: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. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта. Размер расширенной памяти в Кбайтах можно определить, обращаясь к ячейкам CMOS следующим образом:

```
mov  AL,30h ; запись адреса ячейки CMOS
out 70h,AL
in AL,71h  ; чтение младшего байта
mov BL,AL  ; размера расширенной памяти
mov AL,31h ; запись адреса ячейки CMOS
out 70h,AL
in AL,71h  ; чтение старшего байта размера расширенной памяти
```

Выполнение работы.

Написан текст исходного .COM модуля, который выбирает и распечатывает следующую информацию:

- Количество доступной памяти (в килобайтах)
- Размер расширенной памяти (в килобайтах)
- Выводит цепочку блоков управления памятью

Полученный исходный модуль был отлажен. Результаты выполнения программы представлены на рис. 2.

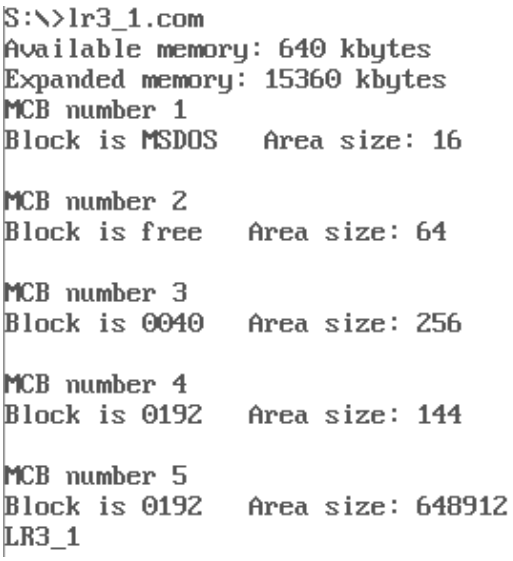

Рисунок 2 – Вывод программы lr3_1.com

Программа занимает всю доступную память.

Далее в программу была внесена процедура, которая освобождала всю память, которую программа не занимает. Была использована функция 4Ah прерывания 21h. Результат выполнения программы представлен на рис. 3.

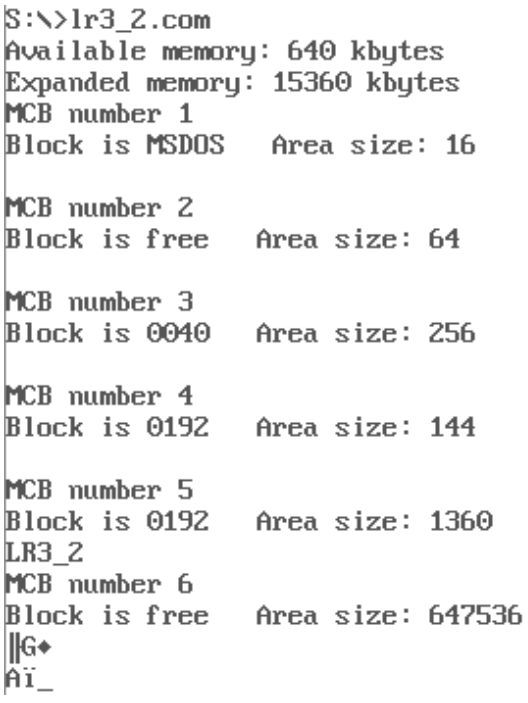

Рисунок 3 – Вывод программы lr3_2.com

В данном случае была высвобождена память. Как видно из скриншота, освобожденная память относится к шестому блоку управления памятью, который является свободным.

Далее в программу была внесена процедура, которая запрашивает 64Кб памяти (функцией 48h прерывания 21h) после освобождения памяти, которую она не занимает. Результат выполнения программы представлен на рис. 4.

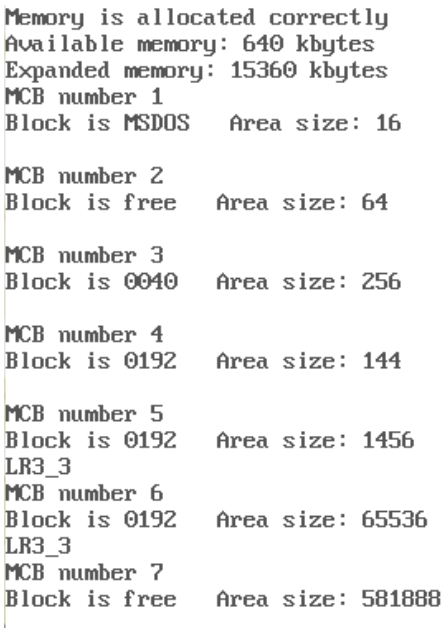

Рисунок 4 – Вывод программы lr3_3.com

В данном случае мы сначала выделяем всю доступную память, потом освобождаем то, что не нужно. Затем запрашиваем блок памяти 64 Кб, в итоге система выделяет нам ещё 65536 б памяти и программа выводит в консоль, что память выделена корректно. Сегментный адрес PSP владельца участка памяти 5 и 6 блока совпадают.

Далее программа была изменена таким образом, чтобы сначала она запрашивала дополнительно 64Кб, а затем освобождала память. Результат выполнения программы представлен на рис. 5.

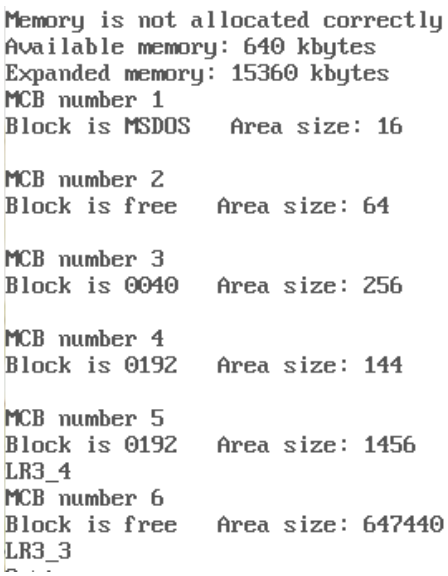

Рисунок 5 – Вывод программы lr3_4.com

В результате возникает ошибка и дополнительный блок памяти в 64Кб не выделяется, о чем нам сообщает программа первой строчкой. Это происходит так как вся доступная память уже была выделена.

Контрольные вопросы

- **Что означает «доступный объём памяти»?**

  Максимальный объем памяти, который может использовать программа.

- **Где MCB блок Вашей программы в списке?**

  Принадлежность MCB можно определить по сегментной компоненте адреса владельца блок.

  - В lr3_1.com, lr3_2.com и lr3_4.com MCB блок программы четвертый и пятый.
  - В lr3_4.com MCB блок программы четвертый, пятый и шестой, так как во время работы программы выделяется дополнительно 64Кб.

- **Какой размер памяти занимает программа в каждом случае?**

  - Lr3_1.com

    Программа занимает (648912Б + 144Б) 649056 байт.

- Lr3_2.com

  Программа занимает (1360Б + 144Б) 1504 байт после освобождения памяти.
- Lr3_3.com

  Программа занимает (1376Б + 144Б) 1520 байт после освобождения памяти и дополнительно 65536 байт после выделения.
- Lr3_4.com

  Программа занимает (1376Б + 144Б) 1520 байт.

Вывод.

В ходе выполнения данной лабораторной работы был изучен список блоков управления памятью, а также методы выделения и освобождения памяти для программы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR3_1.ASM

```
LAB SEGMENT
     ASSUME CS:LAB, DS:LAB, ES:NOTHING, SS:NOTHING
     ORG 100H
MAIN: JMP BEGIN

;данные
AVAILABLE_MEMORY db 'Available memory: $'
EXPENDED_MEMORY  db 'Expanded memory: $'
MCB_NUM_INFO     db  'MCB number $'
AREA_SIZE_INFO   db  '   Area size: $'
END_LINE         db  0Dh, 0Ah, "$"
KBYTES                db ' kbytes', 13,10,'$'
BLOCK                 db  0Dh, 0Ah, 'Block is $'
FREE            db 'free$'
XMS             db 'OS XMS UMB$'
DRIVER_TOP           db 'Excluded top memory of driver$'
DOS              db 'MSDOS$'
OCCUP_386        db '386MAX UMB$'
BLOCK_386        db '386MAX$'
BELONG_386            db '386MAX UMB$'

;-------------------------------------------------------------------------
-----------
PRINT PROC NEAR
           PUSH AX
           MOV AH, 09H
           INT 21H
           POP AX
           RET
PRINT ENDP
;-------------------------------------------------------------------------
-----------
PRINT_KBYTES PROC NEAR
           PUSH DX
           MOV DX, OFFSET KBYTES
           CALL PRINT
           POP DX
           RET
PRINT_KBYTES ENDP
;-------------------------------------------------------------------------
-----------
;ВЫВОДИТ СЛОВО(AX) В 10 С/С
DEC_WORD_PRINT PROC
           PUSH AX
           PUSH CX
           PUSH DX
           PUSH BX
           MOV BX, 10
           XOR CX, CX
     ;В ЦИКЛЕ ДЕЛИМ ЧИСЛО(AX) НА 10(BX) И ОСТАТКИ ОТ ДЕЛЕНИЯ(DX)
ЗАНОСИМ В СТЕК
     GET_NUMBERS:
           DIV BX
           PUSH DX
```

```
            XOR DX, DX
            INC CX
            CMP AX, 0
            JNZ GET_NUMBERS

      ;В ЦИКЛЕ ДОСТАЕМ ИЗ СТЕКА ЧИСЛА В 10 С/С И ВЫВОДИМ
      WRITING:
            POP DX
            OR DL, 48 ;СДВИГ В ASCII ДО ЦИФР
            MOV AH, 2
            INT 21H
            LOOP WRITING

            POP BX
            POP DX
            POP CX
            POP AX
            RET
DEC_WORD_PRINT ENDP
;-------------------------------------------------------------------
-----------
;ВЫВОДИТ БАЙТ(AL) В 16 С/С
HEX_BYTE_PRINT PROC NEAR
            PUSH AX
            PUSH BX
            PUSH DX

            MOV AH, 0
            MOV BL, 10H
            DIV BL
            MOV DX, AX ;В DL - ПЕРВАЯ ЦИФРА В DH - ВТОРАЯ
            MOV AH, 02H
            CMP DL, 0AH
            JL PRINT_1 ;ЕСЛИ В DL - ЦИФРА
            ADD DL, 7   ;СДВИГ В ASCII С ЦИФР ДО БУКВ
      PRINT_1:
            ADD DL, 48
            INT 21H
            MOV DL, DH
            CMP DL, 0AH
            JL PRINT_2
            ADD DL, 7
      PRINT_2:
            ADD DL, 48
            INT 21H;
            POP DX
            POP BX
            POP AX
            RET
      HEX_BYTE_PRINT ENDP
;-------------------------------------------------------------------
-----------
;ВЫВОДИТ СЛОВО(AX) В 16 С/С
HEX_WORD_PRINT PROC NEAR
            PUSH AX
            PUSH DX
            MOV DX,AX
            MOV AL,DH
```

```
          CALL HEX_BYTE_PRINT
          MOV AL,DL
          CALL HEX_BYTE_PRINT
          POP DX
          POP AX
          RET
HEX_WORD_PRINT ENDP
;-------------------------------------------------------------------------
-----------


BEGIN:
          PUSH  AX
          PUSH  BX
          PUSH  CX
          PUSH  DX
          PUSH  ES
          PUSH  SI

     PRINT_AVAILABLE_MEMORY:
          XOR     AX, AX
          INT   12H

          MOV     DX, OFFSET AVAILABLE_MEMORY
          CALL    PRINT
          XOR        DX, DX
          CALL    DEC_WORD_PRINT
          CALL    PRINT_KBYTES

     PRINT_EXPENDED_MEMORY:
          MOV   AL, 30H
          OUT   70H, AL
          IN         AL, 71H
          MOV   BL, AL
          MOV   AL, 31H
          OUT   70H, AL
          IN         AL, 71H

          MOV   AH, AL
          MOV   AL, BL

        MOV     DX, OFFSET EXPENDED_MEMORY
          CALL  PRINT
          XOR        DX, DX
          CALL  DEC_WORD_PRINT
        CALL    PRINT_KBYTES


     PRINT_MCB:
          MOV   AH, 52H
          INT   21H
          MOV   AX, ES:[BX-2] ;АДРЕСС ПЕРВОГО MCB
          MOV   ES, AX
          XOR   CX, CX

     NEXT_MCB:
          INC   CX
          MOV   DX, OFFSET MCB_NUM_INFO
```

```
      PUSH CX
      CALL PRINT
      MOV  AX, CX
      XOR  DX, DX
      CALL DEC_WORD_PRINT ;ВЫВОДИТ НОМЕР ТЕКУЩЕГО MCB
START:
      MOV  DX, OFFSET BLOCK
      CALL PRINT
      XOR  AX, AX
      MOV  AL, ES:[0H]
      PUSH AX
      MOV  AX, ES:[1H]

      CMP  AX, 0H
      JE         FREE_PRINT
      CMP  AX, 6H
      JE         XMS_PRINT
      CMP  AX, 7H
      JE         DRIVER_TOP_PRINT
      CMP  AX, 8H
      JE         DOS_PRINT
      CMP  AX, 0FFFAH
      JE         OCCUP_386_PRINT
      CMP  AX, 0FFFDH
      JE         BLOCK_386_PRINT
      CMP  AX, 0FFFEH
      JE         BELONG_386_PRINT

      XOR  DX, DX
      CALL HEX_WORD_PRINT
      JMP  AREA_SIZE_START

FREE_PRINT:
      MOV  DX, OFFSET FREE
      JMP  PRINTING
XMS_PRINT:
      MOV  DX, OFFSET XMS
      JMP  PRINTING
DRIVER_TOP_PRINT:
      MOV  DX, OFFSET DRIVER_TOP
      JMP  PRINTING
DOS_PRINT:
      MOV  DX, OFFSET DOS
      JMP  PRINTING
OCCUP_386_PRINT:
      MOV  DX, OFFSET OCCUP_386
      JMP  PRINTING
BLOCK_386_PRINT:
      MOV  DX, OFFSET BLOCK_386
      JMP  PRINTING
BELONG_386_PRINT:
      MOV  DX, OFFSET BELONG_386
PRINTING:
      CALL PRINT

AREA_SIZE_START:
      MOV  DX, OFFSET AREA_SIZE_INFO
      CALL PRINT
```

```
        MOV   AX, ES:[3H]
        MOV   BX, 10H
        MUL   BX
        CALL  DEC_WORD_PRINT

        MOV   CX, 8
        XOR   SI, SI
        MOV   DX, OFFSET END_LINE
        CALL  PRINT

    LAST_BYTES_START:
        MOV     DL, ES:[SI + 8H]
        MOV     AH, 02H
        INT     21H
        INC     SI
        LOOP    LAST_BYTES_START

        MOV     AX, ES:[3H]
        MOV     BX, ES
        ADD     BX, AX
        INC     BX
        MOV     ES, BX
        POP     AX
        POP     CX
        CMP     AL, 5AH
        JE      EXIT
        MOV     DX, OFFSET END_LINE
        CALL    PRINT
        JMP     NEXT_MCB

    EXIT:
        POP       SI
        POP  ES
        POP       DX
        POP  CX
        POP       BX
        POP       AX
        XOR       AL, AL
        MOV  AH, 4CH
        INT       21H
        RET


LAB  ENDS
        END  MAIN
```

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR3_2.ASM

```
LAB SEGMENT
                    ASSUME CS:LAB, DS:LAB, ES:NOTHING, SS:NOTHING
                    ORG 100H
MAIN: JMP BEGIN

;данные
AVAILABLE_MEMORY db 'Available memory: $'
EXPENDED_MEMORY  db 'Expanded memory: $'
MCB_NUM_INFO     db  'MCB number $'
AREA_SIZE_INFO       db  '   Area size: $'
END_LINE                 db  0Dh, 0Ah, "$"
KBYTES                        db ' kbytes', 13,10,'$'
BLOCK                         db  0Dh, 0Ah, 'Block is $'
FREE                         db 'free$'
XMS                          db 'OS XMS UMB$'
DRIVER_TOP               db 'Excluded top memory of driver$'
DOS                           db 'MSDOS$'
OCCUP_386                db '386MAX UMB$'
BLOCK_386                db '386MAX$'
BELONG_386               db '386MAX UMB$'
last db ?



;-------------------------------------------------------------------
-------------
PRINT PROC NEAR
                         PUSH AX
                         MOV AH, 09H
                         INT 21H
                         POP AX
                         RET
PRINT ENDP
;-------------------------------------------------------------------
-------------
PRINT_KBYTES PROC NEAR
                         PUSH DX
```

```
                        MOV DX, OFFSET KBYTES
                        CALL PRINT
                        POP DX
                        RET
PRINT_KBYTES ENDP
;-------------------------------------------------------------------
-------------
;ВЫВОДИТ СЛОВО(AX) В 10 С/С
DEC_WORD_PRINT PROC
                        PUSH AX
                        PUSH CX
                        PUSH DX
                        PUSH BX
                        MOV BX, 10
                        XOR CX, CX
                    ;В ЦИКЛЕ ДЕЛИМ ЧИСЛО(AX) НА 10(BX) И ОСТАТКИ ОТ
ДЕЛЕНИЯ(DX) ЗАНОСИМ В СТЕК
                    GET_NUMBERS:
                        DIV BX
                        PUSH DX
                        XOR DX, DX
                        INC CX
                        CMP AX, 0
                        JNZ GET_NUMBERS

                    ;В ЦИКЛЕ ДОСТАЕМ ИЗ СТЕКА ЧИСЛА В 10 С/С И
ВЫВОДИМ
                    WRITING:
                        POP DX
                        OR DL, 48 ;СДВИГ В ASCII ДО ЦИФР
                        MOV AH, 2
                        INT 21H
                        LOOP WRITING

                        POP BX
                        POP DX
                        POP CX
                        POP AX
                        RET
```

```
DEC_WORD_PRINT ENDP
;-------------------------------------------------------------------
-------------
;ВЫВОДИТ БАЙТ(AL) В 16 С/С
HEX_BYTE_PRINT PROC NEAR
                        PUSH AX
                        PUSH BX
                        PUSH DX

                        MOV AH, 0
                        MOV BL, 10H
                        DIV BL
                        MOV DX, AX ;В DL - ПЕРВАЯ ЦИФРА В DH -
ВТОРАЯ
                        MOV AH, 02H
                        CMP DL, 0AH
                        JL PRINT_1 ;ЕСЛИ В DL - ЦИФРА
                        ADD DL, 7   ;СДВИГ В ASCII С ЦИФР ДО БУКВ
                    PRINT_1:
                        ADD DL, 48
                        INT 21H
                        MOV DL, DH
                        CMP DL, 0AH
                        JL PRINT_2
                        ADD DL, 7
                    PRINT_2:
                        ADD DL, 48
                        INT 21H;
                        POP DX
                        POP BX
                        POP AX
                        RET
                    HEX_BYTE_PRINT ENDP
;-------------------------------------------------------------------
-------------
;ВЫВОДИТ СЛОВО(AX) В 16 С/С
HEX_WORD_PRINT PROC NEAR
                        PUSH AX
                        PUSH DX
```

```
                        MOV DX,AX
                        MOV AL,DH
                        CALL HEX_BYTE_PRINT
                        MOV AL,DL
                        CALL HEX_BYTE_PRINT
                        POP DX
                        POP AX
                        RET
HEX_WORD_PRINT ENDP
;-------------------------------------------------------------------
-------------

FREE_MEMORY         PROC
                        PUSH  AX
                        PUSH  BX
                        PUSH  CX

                        MOV   BX, OFFSET PROGRAM_END
                        ADD   BX, 100H
                        MOV   CL, 4
                        SHR   BX, CL
                        ADD   BX, 17
                        MOV   AH, 4AH
                        INT   21H

                        POP   CX
                        POP   BX
                        POP   AX
                    RET
FREE_MEMORY          ENDP
;-------------------------------------------------------------------
-------------
BEGIN:
                        PUSH  AX
                        PUSH  BX
                        PUSH  CX
                        PUSH  DX
                        PUSH  ES
                        PUSH  SI
```

```
                call FREE_MEMORY

            PRINT_AVAILABLE_MEMORY:
                XOR     AX, AX
                INT  12H

                MOV     DX, OFFSET AVAILABLE_MEMORY
                CALL    PRINT
                XOR        DX, DX
                CALL    DEC_WORD_PRINT
                CALL    PRINT_KBYTES

            PRINT_EXPENDED_MEMORY:
                MOV  AL, 30H
                OUT  70H, AL
                IN         AL, 71H
                MOV  BL, AL
                MOV  AL, 31H
                OUT  70H, AL
                IN         AL, 71H

                MOV  AH, AL
                MOV  AL, BL

MOV         DX, OFFSET EXPENDED_MEMORY
                CALL  PRINT
                XOR        DX, DX
                CALL  DEC_WORD_PRINT
CALL    PRINT_KBYTES


            PRINT_MCB:
                MOV  AH, 52H
                INT  21H
                MOV  AX, ES:[BX-2] ;АДРЕСС ПЕРВОГО МСВ
                MOV  ES, AX
                XOR  CX, CX
```

```
NEXT_MCB:
    INC  CX
    MOV  DX, OFFSET MCB_NUM_INFO
    PUSH CX
    CALL PRINT
    MOV  AX, CX
    XOR  DX, DX
    CALL DEC_WORD_PRINT ;ВЫВОДИТ НОМЕР ТЕКУЩЕГО
MCB
START:
    MOV  DX, OFFSET BLOCK
    CALL PRINT
    XOR  AX, AX
    MOV  AL, ES:[0H]
    PUSH AX
    MOV  AX, ES:[1H]

    CMP  AX, 0H
    JE       FREE_PRINT
    CMP  AX, 6H
    JE       XMS_PRINT
    CMP  AX, 7H
    JE       DRIVER_TOP_PRINT
    CMP  AX, 8H
    JE       DOS_PRINT
    CMP  AX, 0FFFAH
    JE       OCCUP_386_PRINT
    CMP  AX, 0FFFDH
    JE       BLOCK_386_PRINT
    CMP  AX, 0FFFEH
    JE       BELONG_386_PRINT

    XOR  DX, DX
    CALL HEX_WORD_PRINT
    JMP  AREA_SIZE_START

FREE_PRINT:
    MOV  DX, OFFSET FREE
    JMP  PRINTING
```

```
XMS_PRINT:
    MOV   DX, OFFSET XMS
    JMP   PRINTING
DRIVER_TOP_PRINT:
    MOV   DX, OFFSET DRIVER_TOP
    JMP   PRINTING
DOS_PRINT:
    MOV   DX, OFFSET DOS
    JMP   PRINTING
OCCUP_386_PRINT:
    MOV   DX, OFFSET OCCUP_386
    JMP   PRINTING
BLOCK_386_PRINT:
    MOV   DX, OFFSET BLOCK_386
    JMP   PRINTING
BELONG_386_PRINT:
    MOV   DX, OFFSET BELONG_386
PRINTING:
    CALL  PRINT

AREA_SIZE_START:
    MOV   DX, OFFSET AREA_SIZE_INFO
    CALL  PRINT
    MOV   AX, ES:[3H]
    MOV   BX, 10H
    MUL   BX
    CALL  DEC_WORD_PRINT

    MOV   CX, 8
    XOR   SI, SI
    MOV   DX, OFFSET END_LINE
    CALL  PRINT

LAST_BYTES_START:
    MOV     DL, ES:[SI + 8H]
    MOV     AH, 02H
    INT     21H
    INC     SI
    LOOP    LAST_BYTES_START
```

```
                MOV     AX, ES:[3H]
                MOV     BX, ES
                ADD     BX, AX
                INC     BX
                MOV     ES, BX
                POP     AX
                POP     CX
                CMP     AL, 5AH
                JE      EXIT
                MOV     DX, OFFSET END_LINE
                CALL    PRINT
                JMP     NEXT_MCB

            EXIT:
                POP       SI
                POP  ES
                POP       DX
                POP  CX
                POP       BX
                POP       AX
                XOR       AL, AL
                MOV  AH, 4CH
                INT       21H
                RET

            PROGRAM_END:

LAB         ENDS
                END  MAIN
```

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR3_3.ASM

```
LAB SEGMENT
                    ASSUME CS:LAB, DS:LAB, ES:NOTHING, SS:NOTHING
                    ORG 100H
MAIN: JMP BEGIN

;данные
AVAILABLE_MEMORY db 'Available memory: $'
EXPENDED_MEMORY  db 'Expanded memory: $'
MCB_NUM_INFO     db  'MCB number $'
AREA_SIZE_INFO       db  '   Area size: $'
END_LINE                 db  0Dh, 0Ah, "$"
KBYTES                        db ' kbytes', 13,10,'$'
BLOCK                         db  0Dh, 0Ah, 'Block is $'
FREE                         db 'free$'
XMS                          db 'OS XMS UMB$'
DRIVER_TOP               db 'Excluded top memory of driver$'
DOS                           db 'MSDOS$'
OCCUP_386                db '386MAX UMB$'
BLOCK_386                db '386MAX$'
BELONG_386               db '386MAX UMB$'

MEMORY_SUCCESS   db 'Memory is allocated correctly',13,10,'$'
MEMORY_FAIL      db 'Memory is not allocated correctly',13,10,'$'



;-------------------------------------------------------------------
-------------
PRINT PROC NEAR
                        PUSH AX
                        MOV AH, 09H
                        INT 21H
                        POP AX
                        RET
PRINT ENDP
;-------------------------------------------------------------------
-------------
```

```
PRINT_KBYTES PROC NEAR
                        PUSH DX
                        MOV DX, OFFSET KBYTES
                        CALL PRINT
                        POP DX
                        RET
PRINT_KBYTES ENDP
;-------------------------------------------------------------------
-------------
;ВЫВОДИТ СЛОВО(AX) В 10 С/С
DEC_WORD_PRINT PROC
                        PUSH AX
                        PUSH CX
                        PUSH DX
                        PUSH BX
                        MOV BX, 10
                        XOR CX, CX
                    ;В ЦИКЛЕ ДЕЛИМ ЧИСЛО(AX) НА 10(BX) И ОСТАТКИ ОТ
ДЕЛЕНИЯ(DX) ЗАНОСИМ В СТЕК
                    GET_NUMBERS:
                        DIV BX
                        PUSH DX
                        XOR DX, DX
                        INC CX
                        CMP AX, 0
                        JNZ GET_NUMBERS

                    ;В ЦИКЛЕ ДОСТАЕМ ИЗ СТЕКА ЧИСЛА В 10 С/С И
ВЫВОДИМ
                    WRITING:
                        POP DX
                        OR DL, 48 ;СДВИГ В ASCII ДО ЦИФР
                        MOV AH, 2
                        INT 21H
                        LOOP WRITING

                        POP BX
                        POP DX
                        POP CX
```

```
                        POP AX
                        RET
DEC_WORD_PRINT ENDP
;-------------------------------------------------------------------
-------------
;ВЫВОДИТ БАЙТ(AL) В 16 С/С
HEX_BYTE_PRINT PROC NEAR
                        PUSH AX
                        PUSH BX
                        PUSH DX

                        MOV AH, 0
                        MOV BL, 10H
                        DIV BL
                        MOV DX, AX ;В DL - ПЕРВАЯ ЦИФРА В DH -
ВТОРАЯ
                        MOV AH, 02H
                        CMP DL, 0AH
                        JL PRINT_1 ;ЕСЛИ В DL - ЦИФРА
                        ADD DL, 7   ;СДВИГ В ASCII С ЦИФР ДО БУКВ
                    PRINT_1:
                        ADD DL, 48
                        INT 21H
                        MOV DL, DH
                        CMP DL, 0AH
                        JL PRINT_2
                        ADD DL, 7
                    PRINT_2:
                        ADD DL, 48
                        INT 21H;
                        POP DX
                        POP BX
                        POP AX
                        RET
                    HEX_BYTE_PRINT ENDP
;-------------------------------------------------------------------
-------------
;ВЫВОДИТ СЛОВО(AX) В 16 С/С
HEX_WORD_PRINT PROC NEAR
```

```
                        PUSH AX
                        PUSH DX
                        MOV DX,AX
                        MOV AL,DH
                        CALL HEX_BYTE_PRINT
                        MOV AL,DL
                        CALL HEX_BYTE_PRINT
                        POP DX
                        POP AX
                        RET
HEX_WORD_PRINT ENDP
;-------------------------------------------------------------------
-------------

FREE_MEMORY         PROC
                        PUSH  AX
                        PUSH  BX
                        PUSH  CX

                        MOV   BX, OFFSET PROGRAM_END
                        ADD   BX, 100H
                        MOV   CL, 4
                        SHR   BX, CL
                        ADD   BX, 17
                        MOV   AH, 4AH
                        INT   21H

                        POP   CX
                        POP   BX
                        POP   AX
                    RET
FREE_MEMORY          ENDP
;-------------------------------------------------------------------
-------------
ADD_MEMORY     PROC
                        PUSH  AX
                        PUSH  BX
                        PUSH    DX
```

```
                        MOV   BX, 1000H
                        MOV   AH, 48H
                        INT   21H

                        JC MEM_FAIL

                        MEM_SUCCESS:
                              MOV DX,OFFSET MEMORY_SUCCESS
                              CALL PRINT
                              JMP FUNC_END

                        MEM_FAIL:
                              MOV DX,OFFSET MEMORY_FAIL
                              CALL PRINT




                        FUNC_END:

                        POP     DX
                        POP   BX
                        POP   AX
                    RETADD_MEMORY      ENDP
;-------------------------------------------------------------------
-------------

BEGIN:
                        PUSH  AX
                        PUSH  BX
                        PUSH  CX
                        PUSH  DX
                        PUSH  ES
                        PUSH  SI

                        call FREE_MEMORY
                        call ADD_MEMORY

                    PRINT_AVAILABLE_MEMORY:
```

```
        XOR     AX, AX
        INT  12H

        MOV     DX, OFFSET AVAILABLE_MEMORY
        CALL    PRINT
        XOR        DX, DX
        CALL    DEC_WORD_PRINT
        CALL    PRINT_KBYTES

    PRINT_EXPENDED_MEMORY:
        MOV  AL, 30H
        OUT  70H, AL
        IN         AL, 71H
        MOV  BL, AL
        MOV  AL, 31H
        OUT  70H, AL
        IN         AL, 71H

        MOV  AH, AL
        MOV  AL, BL

MOV         DX, OFFSET EXPENDED_MEMORY
        CALL  PRINT
        XOR        DX, DX
        CALL  DEC_WORD_PRINT
CALL    PRINT_KBYTES


    PRINT_MCB:
        MOV  AH, 52H
        INT  21H
        MOV  AX, ES:[BX-2] ;АДРЕСС ПЕРВОГО MCB
        MOV  ES, AX
        XOR  CX, CX

    NEXT_MCB:
        INC  CX
        MOV  DX, OFFSET MCB_NUM_INFO
        PUSH CX
```

```
        CALL  PRINT
        MOV   AX, CX
        XOR   DX, DX
        CALL  DEC_WORD_PRINT ;ВЫВОДИТ НОМЕР ТЕКУЩЕГО
MCB
START:
        MOV   DX, OFFSET BLOCK
        CALL  PRINT
        XOR   AX, AX
        MOV   AL, ES:[0H]
        PUSH  AX
        MOV   AX, ES:[1H]

        CMP   AX, 0H
        JE          FREE_PRINT
        CMP   AX, 6H
        JE          XMS_PRINT
        CMP   AX, 7H
        JE          DRIVER_TOP_PRINT
        CMP   AX, 8H
        JE          DOS_PRINT
        CMP   AX, 0FFFAH
        JE          OCCUP_386_PRINT
        CMP   AX, 0FFFDH
        JE          BLOCK_386_PRINT
        CMP   AX, 0FFFEH
        JE          BELONG_386_PRINT

        XOR   DX, DX
        CALL  HEX_WORD_PRINT
        JMP   AREA_SIZE_START

FREE_PRINT:
        MOV   DX, OFFSET FREE
        JMP   PRINTING
XMS_PRINT:
        MOV   DX, OFFSET XMS
        JMP   PRINTING
DRIVER_TOP_PRINT:
```

```
    MOV   DX, OFFSET DRIVER_TOP
    JMP   PRINTING
DOS_PRINT:
    MOV   DX, OFFSET DOS
    JMP   PRINTING
OCCUP_386_PRINT:
    MOV   DX, OFFSET OCCUP_386
    JMP   PRINTING
BLOCK_386_PRINT:
    MOV   DX, OFFSET BLOCK_386
    JMP   PRINTING
BELONG_386_PRINT:
    MOV   DX, OFFSET BELONG_386
PRINTING:
    CALL  PRINT

AREA_SIZE_START:
    MOV   DX, OFFSET AREA_SIZE_INFO
    CALL  PRINT
    MOV   AX, ES:[3H]
    MOV   BX, 10H
    MUL   BX
    CALL  DEC_WORD_PRINT

    MOV   CX, 8
    XOR   SI, SI
    MOV   DX, OFFSET END_LINE
    CALL  PRINT

LAST_BYTES_START:
    MOV     DL, ES:[SI + 8H]
    MOV     AH, 02H
    INT     21H
    INC     SI
    LOOP    LAST_BYTES_START

    MOV     AX, ES:[3H]
    MOV     BX, ES
    ADD     BX, AX
```

```
        INC     BX
        MOV     ES, BX
        POP     AX
        POP     CX
        CMP     AL, 5AH
        JE      EXIT
        MOV     DX, OFFSET END_LINE
        CALL    PRINT
        JMP     NEXT_MCB

    EXIT:
        POP       SI
        POP  ES
        POP       DX
        POP  CX
        POP       BX
        POP       AX
        XOR       AL, AL
        MOV  AH, 4CH
        INT       21H
        RET

    PROGRAM_END:

LAB     ENDS
        END  MAIN
```

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR3_4.ASM

```
LAB SEGMENT
     ASSUME CS:LAB, DS:LAB, ES:NOTHING, SS:NOTHING
     ORG 100H
MAIN: JMP BEGIN

;данные
AVAILABLE_MEMORY db 'Available memory: $'
EXPENDED_MEMORY  db 'Expanded memory: $'
MCB_NUM_INFO     db  'MCB number $'
AREA_SIZE_INFO   db  '   Area size: $'
END_LINE         db  0Dh, 0Ah, "$"
KBYTES                db ' kbytes', 13,10,'$'
BLOCK                 db  0Dh, 0Ah, 'Block is $'
FREE           db 'free$'
XMS            db 'OS XMS UMB$'
DRIVER_TOP          db 'Excluded top memory of driver$'
DOS             db 'MSDOS$'
OCCUP_386       db '386MAX UMB$'
BLOCK_386       db '386MAX$'
BELONG_386           db '386MAX UMB$'

MEMORY_SUCCESS   db 'Memory is allocated correctly',13,10,'$'
MEMORY_FAIL      db 'Memory is not allocated correctly',13,10,'$'


;-------------------------------------------------------------------------
-----------
PRINT PROC NEAR
          PUSH AX
          MOV AH, 09H
          INT 21H
          POP AX
          RET
PRINT ENDP
;-------------------------------------------------------------------------
-----------
PRINT_KBYTES PROC NEAR
          PUSH DX
          MOV DX, OFFSET KBYTES
          CALL PRINT
          POP DX
          RET
PRINT_KBYTES ENDP
;-------------------------------------------------------------------------
-----------
;ВЫВОДИТ СЛОВО(AX) В 10 С/С
DEC_WORD_PRINT PROC
          PUSH AX
          PUSH CX
          PUSH DX
          PUSH BX
          MOV BX, 10
          XOR CX, CX
     ;В ЦИКЛЕ ДЕЛИМ ЧИСЛО(AX) НА 10(BX) И ОСТАТКИ ОТ ДЕЛЕНИЯ(DX) ЗАНОСИМ
В СТЕК
     GET_NUMBERS:
```

```
            DIV BX
            PUSH DX
            XOR DX, DX
            INC CX
            CMP AX, 0
            JNZ GET_NUMBERS

      ;В ЦИКЛЕ ДОСТАЕМ ИЗ СТЕКА ЧИСЛА В 10 С/С И ВЫВОДИМ
      WRITING:
            POP DX
            OR DL, 48 ;СДВИГ В ASCII ДО ЦИФР
            MOV AH, 2
            INT 21H
            LOOP WRITING

            POP BX
            POP DX
            POP CX
            POP AX
            RET
DEC_WORD_PRINT ENDP
;-------------------------------------------------------------------------
-----------
;ВЫВОДИТ БАЙТ(AL) В 16 С/С
HEX_BYTE_PRINT PROC NEAR
            PUSH AX
            PUSH BX
            PUSH DX

            MOV AH, 0
            MOV BL, 10H
            DIV BL
            MOV DX, AX ;В DL - ПЕРВАЯ ЦИФРА В DH - ВТОРАЯ
            MOV AH, 02H
            CMP DL, 0AH
            JL PRINT_1 ;ЕСЛИ В DL - ЦИФРА
            ADD DL, 7   ;СДВИГ В ASCII С ЦИФР ДО БУКВ
      PRINT_1:
            ADD DL, 48
            INT 21H
            MOV DL, DH
            CMP DL, 0AH
            JL PRINT_2
            ADD DL, 7
      PRINT_2:
            ADD DL, 48
            INT 21H;
            POP DX
            POP BX
            POP AX
            RET
      HEX_BYTE_PRINT ENDP
;-------------------------------------------------------------------------
-----------
;ВЫВОДИТ СЛОВО(AX) В 16 С/С
HEX_WORD_PRINT PROC NEAR
            PUSH AX
            PUSH DX
```

```
            MOV DX,AX
            MOV AL,DH
            CALL HEX_BYTE_PRINT
            MOV AL,DL
            CALL HEX_BYTE_PRINT
            POP DX
            POP AX
            RET
HEX_WORD_PRINT ENDP
;-------------------------------------------------------------------------
-----------

FREE_MEMORY     PROC
            PUSH  AX
            PUSH  BX
            PUSH  CX

            MOV   BX, OFFSET PROGRAM_END
            ADD   BX, 100H
            MOV   CL, 4
            SHR   BX, CL
            ADD   BX, 17
            MOV   AH, 4AH
            INT   21H

            POP   CX
            POP   BX
            POP   AX
     RET
FREE_MEMORY      ENDP
;-------------------------------------------------------------------------
-----------
ADD_MEMORY     PROC
            PUSH  AX
            PUSH  BX
            PUSH    DX

            MOV   BX, 1000H
            MOV   AH, 48H
            INT   21H

            JC MEM_FAIL

            MEM_SUCCESS:
                  MOV DX,OFFSET MEMORY_SUCCESS
                  CALL PRINT
                  JMP FUNC_END

            MEM_FAIL:
                  MOV DX,OFFSET MEMORY_FAIL
                  CALL PRINT



            FUNC_END:

            POP     DX
            POP   BX
```

```
          POP  AX
     RETADD_MEMORY      ENDP
;-------------------------------------------------------------------------
-----------

BEGIN:
          PUSH AX
          PUSH BX
          PUSH CX
          PUSH DX
          PUSH ES
          PUSH SI

          call ADD_MEMORY
          call FREE_MEMORY

     PRINT_AVAILABLE_MEMORY:
          XOR     AX, AX
          INT  12H

          MOV     DX, OFFSET AVAILABLE_MEMORY
          CALL    PRINT
          XOR        DX, DX
          CALL    DEC_WORD_PRINT
          CALL    PRINT_KBYTES

     PRINT_EXPENDED_MEMORY:
          MOV  AL, 30H
          OUT  70H, AL
          IN         AL, 71H
          MOV  BL, AL
          MOV  AL, 31H
          OUT  70H, AL
          IN         AL, 71H

          MOV  AH, AL
          MOV  AL, BL

        MOV     DX, OFFSET EXPENDED_MEMORY
          CALL PRINT
          XOR        DX, DX
          CALL DEC_WORD_PRINT
        CALL    PRINT_KBYTES


     PRINT_MCB:
          MOV  AH, 52H
          INT  21H
          MOV  AX, ES:[BX-2] ;АДРЕСС ПЕРВОГО MCB
          MOV  ES, AX
          XOR  CX, CX

     NEXT_MCB:
          INC  CX
          MOV  DX, OFFSET MCB_NUM_INFO
          PUSH CX
          CALL PRINT
          MOV  AX, CX
```

```
      XOR  DX, DX
      CALL DEC_WORD_PRINT ;ВЫВОДИТ НОМЕР ТЕКУЩЕГО MCB
START:
      MOV  DX, OFFSET BLOCK
      CALL PRINT
      XOR  AX, AX
      MOV  AL, ES:[0H]
      PUSH AX
      MOV  AX, ES:[1H]

      CMP  AX, 0H
      JE         FREE_PRINT
      CMP  AX, 6H
      JE         XMS_PRINT
      CMP  AX, 7H
      JE         DRIVER_TOP_PRINT
      CMP  AX, 8H
      JE         DOS_PRINT
      CMP  AX, 0FFFAH
      JE         OCCUP_386_PRINT
      CMP  AX, 0FFFDH
      JE         BLOCK_386_PRINT
      CMP  AX, 0FFFEH
      JE         BELONG_386_PRINT

      XOR  DX, DX
      CALL HEX_WORD_PRINT
      JMP  AREA_SIZE_START

FREE_PRINT:
      MOV  DX, OFFSET FREE
      JMP  PRINTING
XMS_PRINT:
      MOV  DX, OFFSET XMS
      JMP  PRINTING
DRIVER_TOP_PRINT:
      MOV  DX, OFFSET DRIVER_TOP
      JMP  PRINTING
DOS_PRINT:
      MOV  DX, OFFSET DOS
      JMP  PRINTING
OCCUP_386_PRINT:
      MOV  DX, OFFSET OCCUP_386
      JMP  PRINTING
BLOCK_386_PRINT:
      MOV  DX, OFFSET BLOCK_386
      JMP  PRINTING
BELONG_386_PRINT:
      MOV  DX, OFFSET BELONG_386
PRINTING:
      CALL PRINT

AREA_SIZE_START:
      MOV  DX, OFFSET AREA_SIZE_INFO
      CALL PRINT
      MOV  AX, ES:[3H]
      MOV  BX, 10H
      MUL  BX
```

```
        CALL  DEC_WORD_PRINT

        MOV   CX, 8
        XOR   SI, SI
        MOV   DX, OFFSET END_LINE
        CALL  PRINT

    LAST_BYTES_START:
        MOV     DL, ES:[SI + 8H]
        MOV     AH, 02H
        INT     21H
        INC     SI
        LOOP    LAST_BYTES_START

        MOV     AX, ES:[3H]
        MOV     BX, ES
        ADD     BX, AX
        INC     BX
        MOV     ES, BX
        POP     AX
        POP     CX
        CMP     AL, 5AH
        JE      EXIT
        MOV     DX, OFFSET END_LINE
        CALL    PRINT
        JMP     NEXT_MCB

    EXIT:
        POP         SI
        POP   ES
        POP         DX
        POP   CX
        POP         BX
        POP         AX
        XOR         AL, AL
        MOV   AH, 4CH
        INT         21H
        RET

    PROGRAM_END:

LAB ENDS
        END   MAIN
```