МИНОБРНАУКИ РОССИИ САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Операционные системы»

Тема: Исследование организации управления основной памятью

Студентка гр. 8381	Ивлева О.А	
Преподаватель	Ефремов М.	A

Санкт-Петербург 2020

Цель работы.

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

Основные теоретические положения.

Учет занятой и свободной памяти ведется при помощи списка блоков управления памятью MCB (Memory Control Block). MCB занимает 16 байт (параграф) и располагается всегда с адреса кратного 16 (адрес сегмента ОП) и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет. Структура МСВ:

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

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

Адрес первого МСВ хранится во внутренней структуре MS DOS, называемой "List of Lists" (список списков). Доступ к указателю на эту структуру можно получить, используя функцию f52h "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 МСВ.

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.

Выполнение работы.

Сборка и отладка производились на базе эмулятора DOSBox 0.74-3.

Был написан текст исходного .COM модуля с именем lr31.com, который выводит блоки памяти. Наша программа (0192) до очистки памяти занимает 144+648912 байт.

```
D:\>lr31.com
Avl mem: 640 kb
Ext mem: 15360 kb
MCB num: 1
Block is MS DOS Area size: 16
MCB num: 2
Block is free Area size: 64
MCB num: 3
Block is 6040 Area size: 256
MCB num: 4
Block is 0192 Area size: 144
MCB num: 5
Block is 0192 Area size: 648912
LR31
D:\>_
```

Рисунок 1 – lr31.com

Был написан текст исходного .COM модуля с именем lr32.com, который выводит блоки памяти. Наша программа (0192) с очисткой памяти занимает 144 + 1104 байт. Также присутствует 6-ой блок освобожденной памяти (free), который занимает 647792 байта.

```
\>lr32.com
 vl mem: 640 kb
 xt mem: 15360 kb
Block is MS DOS
                      Area size: 16
MCB num: 2
Block is free
                    Area size: 64
MCB num: 3
Block is 0040
                   Area size: 256
MCB num: 4
Block is 0192
                   Area size: 144
MCB num: 5
Block is 0192
                   Area size: 1104
MCB num: 6
Block is free
                    Area size: 647792
```

Рисунок 2 - lr32.com

Был написан текст исходного .COM модуля с именем lr32.com, который выводит блоки памяти. В нем программа сначала освобождает память, а потом запрашивает 64 кбайт. Наша программа (0192) занимает 144 + 1120 + 65536 байт. Также присутствует 7-ой блок освобожденной памяти (free), который занимает 582224 байта.

```
vl mem: 640 kb
 xt mem: 15360 kb
MCB num: 1
Block is MS DOS
                      Area size: 16
 Block is free
                    Area size: 64
MCB num: 3
Block is 0040
                   Area size: 256
1CB num: 4
Block is 0192
                   Area size: 144
 1CB num: 5
Block is 0192
                   Area size: 1120
MCB num: 6
Block is 0192
                   Area size: 65536
 KB num: 7
                    Area size: 582224
```

Рисунок 3 –lr33.com

Был написан текст исходного .COM модуля с именем lr34.com, который выводит блоки памяти. В нем программа сначала запрашивает дополнительные 64кб, которые не могут выделиться, так как программа занимает всю свободную память, а затем происходит освобождение памяти (6 блок).

```
D:\>lr34.com
Avl mem: 640 kb
Ext mem: 15360 kb
MCB num: 1
Block is MS DOS Area size: 16

MCB num: 2
Block is free Area size: 64

MCB num: 3
Block is 6040 Area size: 256

MCB num: 4
Block is 0192 Area size: 144

MCB num: 5
Block is 0192 Area size: 1120
LR34
MCB num: 6
Block is free Area size: 647776
LR33
D:\>_
```

Рисунок 4 –lr34.com

Контрольные вопросы

1. Что означает «доступный объем» памяти?

Максимальный объем памяти, который доступен программе.

2. Где МСВ блок вашей программы в списке?

(0192)

Lr31, lr32, lr34 – 4 и 5 блок.

Lr33 – 4, 5, 6 блоки

3. Какой размер памяти занимает программа в каждом случае?

Lr31: 144+648912

Lr32: 144 + 1104

Lr33: 144 + 1120 + 65536

Lr34: 144 + 1120

Выводы.

В ходе выполнения лабораторной работы были изучены методы выделения и освобождения памяти для программы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR31.ASM

```
LAB segment
           ASSUME CS: LAB, DS: LAB, ES: NOTHING, SS: NOTHING
           org 100h
START: jmp BEGIN
STR_AVL_MEMORY
                       db "Avl mem: $"
STR_EXT_MEMORY
                       db "Ext mem: $"
STR MCB NUM
                       db "MCB num: $"
                       db " Area size: $"
STR AREA SIZE
END LINE
                       db 0Dh, 0Ah, "$"
                    db " kb", 0Dh, 0Ah, "$"
KBYTES
                            db " driver's top memory$"
STR_TM
                       db " MS DOS$"
STR_DOS
                       db 0Dh, 0Ah, "Block is $"
STR INFO
                       db "free$"
STR FREE
                       db " OS XMS UMB$"
STR XMS
                       db " busy by 386MAX UMB$"
STR_386CB
                       db " blocked by 386MAX$"
STR_386B
                       db " 386MAX UMB$"
STR_386
WRITE
           PROC near
           push AX
           mov
                 AH, 09h
           int
                       21h
                 AX
           pop
      ret
WRITE
           ENDP
KBYTES_PRINT
                PROC
                        near
                DX
        push
                DX, offset KBYTES
        mov
                WRITE
        call
                DX
        pop
    ret
KBYTES_PRINT
                ENDP
DEC_WORD_PRINT
                 PROC
           push AX
           push
                 CX
           push
                 DX
           push
                 BX
                 BX, 10
           mov
                 CX, CX
           xor
     NUM:
           div
                 BX
           push DX
```

xor

inc

DX, DX

 CX

```
AX, 0h
              cmp
              jnz
                     NUM
       PRINT_NUM:
                     \mathsf{DX}
              pop
                            DL, 30h
              or
              mov
                     AH, 02h
              int
                     21h
              loop
                     PRINT_NUM
              pop
                     BX
                     \mathsf{DX}
              pop
                     \mathsf{CX}
              pop
                     AX
              pop
       ret
DEC_WORD_PRINT
                     ENDP
HEX_BYTE_PRINT
                     PROC
                     AX
              push
                     \mathsf{BX}
              push
              push
                    DX
                     AH, 0
              mov
                     BL, 10h
              mov
              div
                     BL
                     DX, AX
              mov
                     AH, 02h
              mov
              cmp
                     DL, 0Ah
              jl
                            PRINT
              add
                     DL, 07h
       PRINT:
              add
                     DL, '0'
                     21h;
              int
              mov
                     DL, DH
                     DL, 0Ah
              cmp
              jl
                            PRINT_EXT
              add
                     DL, 07h
       PRINT_EXT:
                     DL, '0'
              add
              int
                     21h;
                     \mathsf{DX}
              pop
              pop
                     BX
              pop
                     AX
       ret
HEX_BYTE_PRINT
                     ENDP
HEX_WORD_PRINT
                     PROC
              push AX
              push AX
              mov AL, AH
```

```
call HEX_BYTE_PRINT
            pop AX
            call HEX_BYTE_PRINT
            pop AX
      ret
HEX_WORD_PRINT
                  ENDP
PRINT_AVL_MEMORY PROC NEAR
          push
                  ΑX
          push
                  BX
          push
                  DX
          push
                  SI
          xor
                  AX, AX
                  12h
            int
                  DX, offset STR_AVL_MEMORY
          mov
          call
                  WRITE
                        DX, DX
            xor
                  DEC_WORD_PRINT
          call
        call
                KBYTES_PRINT
                  SI
          pop
          pop
                  DX
                  BX
          pop
                  AX
          pop
      ret
PRINT_AVL_MEMORY ENDP
PRINT_EXT_MEMORY PROC NEAR
            push AX
            push BX
            push DX
            push SI
                  AL, 30h
            mov
                  70h, AL
            out
            in
                       AL, 71h
                  BL, AL
            mov
                  AL, 31h
            mov
                  70h, AL
            out
                        AL, 71h
            in
                  AH, AL
            mov
            mov
                  AL, BL
        mov
                  DX, offset STR_EXT_MEMORY
            call WRITE
            xor
                        DX, DX
            call DEC_WORD_PRINT
                KBYTES_PRINT
        call
                  SI
            pop
                  DX
            pop
```

```
BX
            pop
                   AX
            pop
      ret
PRINT_EXT_MEMORY ENDP
PRINT_MCB
                   PROC
            push AX
            push
                   BX
            push
                  \mathsf{CX}
            push DX
            push
                  ES
            push
                  SI
                   AH, 52h
            mov
            int
                   21h
                   AX, ES:[BX-2]
            mov
            mov
                   ES, AX
                   CX, CX
            xor
      NEXT_MCB:
            inc
                   \mathsf{CX}
            mov
                   DX, offset STR_MCB_NUM
            push CX
            call
                  WRITE
            mov
                   AX, CX
                   DX, DX
            xor
            call DEC_WORD_PRINT
      OWNER_START:
            mov
                   DX, offset STR_INFO
            call
                  WRITE
            xor
                   AX, AX
                   AL, ES:[0h]
            mov
                  AX
            push
                   AX, ES:[1h]
            mov
            cmp
                   AX, 0h
                         PRINT_FREE
            je
            cmp
                   AX, 6h
                         PRINT_XMS
            je
                   AX, 7h
            cmp
                         PRINT_TM
            jе
            cmp
                   AX, 8h
                         PRINT_DOS
            jе
                   AX, 0FFFAh
            cmp
                         PRINT_386CB
            je
                   AX, 0FFFDh
            cmp
            je
                         PRINT_386B
            cmp
                   AX, 0FFFEh
            je
                         PRINT_386
                   DX, DX
            xor
            call HEX_WORD_PRINT
            jmp
                   AREA_SIZE_START
```

```
mov
            DX, offset STR_FREE
            OWNER_END
      jmp
PRINT_XMS:
            DX, offset STR_XMS
      \text{mov} \\
            OWNER_END
      jmp
PRINT_TM:
            DX, offset STR_TM
      mov
      jmp
            OWNER_END
PRINT_DOS:
      mov
            DX, offset STR_DOS
            OWNER_END
      jmp
PRINT_386CB:
            DX, offset STR_386CB
      mov
            OWNER_END
      jmp
PRINT_386B:
            DX, offset STR_386B
      \text{mov}
      jmp
            OWNER_END
PRINT_386:
            DX, offset STR_386
      mov
OWNER_END:
      call
            WRITE
AREA_SIZE_START:
      mov
            DX, offset STR_AREA_SIZE
      call
            WRITE
            AX, ES:[3h]
      {\sf mov}
      mov
            BX, 10h
            BX
      mul
      call
            DEC_WORD_PRINT
      mov
            CX, 8
      xor
            SI, SI
            DX, offset END_LINE
      mov
      call WRITE
LAST_BYTES_START:
               DL, ES:[SI + 8h]
      mov
      mov
               AH, 02h
      int
               21h
      inc
               SI
               LAST_BYTES_START
      loop
      mov
               AX, ES:[3h]
               BX, ES
      mov
      add
               BX, AX
      inc
               BX
               ES, BX
      mov
               ΑX
      pop
      pop
               CX
               AL, 5Ah
      cmp
               ENDING
      jе
               DX, offset END_LINE
      mov
      call
               WRITE
      jmp
               NEXT_MCB
```

```
ENDING:
             pop
                      SI
             pop
                      ES
                      \mathsf{DX}
             pop
             pop
                      \mathsf{CX}
                      ВХ
             pop
                      \mathsf{AX}
             pop
      ret
PRINT_MCB
                    ENDP
BEGIN:
         call PRINT_AVL_MEMORY
             call PRINT_EXT_MEMORY
                  PRINT_MCB
         call
                  AL, AL
         xor
                    AH, 4Ch
           mov
                    21h
           int
      PROG_END:
```

ends

START

LAB end

приложение Б

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR32.ASM

```
LAB segment
```

```
ASSUME CS: LAB, DS: LAB, ES: NOTHING, SS: NOTHING org 100h
```

START: jmp BEGIN

```
STR_AVL_MEMORY
                      db "Avl mem: $"
                      db "Ext mem: $"
STR EXT MEMORY
                          "MCB num: $"
STR_MCB_NUM
                      db
                      db " Area size: $"
STR_AREA_SIZE
                      db 0Dh, 0Ah, "$"
END_LINE
                   db " kb", 0Dh, 0Ah, "$"
KBYTES
                            db " driver's top memory$"
STR_TM
STR DOS
                      db " MS DOS$"
STR_INFO
                      db 0Dh, 0Ah, "Block is $"
                      db "free$"
STR_FREE
                      db " OS XMS UMB$"
STR_XMS
STR_386CB
                      db "busy by 386MAX UMB$"
                      db " blocked by 386MAX$"
STR 386B
                      db " 386MAX UMB$"
STR_386
```

WRITE PROC near push AX mov AH, 09h int 21h

pop AX

ret

WRITE ENDP

```
KBYTES_PRINT PROC near
```

push DX

mov DX, offset KBYTES

call WRITE pop DX

ret

KBYTES_PRINT ENDP

```
DEC_WORD_PRINT PROC; IN: AX
```

push AX
push CX
push DX
push BX

mov BX, 10

```
CX, CX
             xor
      NUM:
             div
                    ВХ
              push
                    DX
             xor
                    DX, DX
              inc
                    \mathsf{CX}
                    AX, 0h
              cmp
              jnz
                    NUM
       PRINT_NUM:
             pop
                    DX
                           DL, 30h
             or
             \text{mov}
                    AH, 02h
              int
                    21h
              loop
                    PRINT_NUM
              pop
                    BX
                    DX
              pop
              pop
                    \mathsf{CX}
              pop
                    AX
       ret
DEC_WORD_PRINT
                    ENDP
                    PROC
HEX_BYTE_PRINT
              push AX
                    \mathsf{BX}
              push
              push
                    DX
                    AH, 0
             mov
             mov
                    BL, 10h
                    BL
             div
                    DX, AX
             mov
                    AH, 02h
             mov
                    DL, 0Ah
              cmp
              jl
                           PRINT
              add
                    DL, 07h
      PRINT:
                    DL, '0'
              add
              int
                    21h;
                    DL, DH
             \text{mov}
                    DL, 0Ah
              cmp
                           PRINT_EXT
              jl
              add
                    DL, 07h
       PRINT_EXT:
                    DL, '0'
              add
```

```
int
                  21h;
                  DX
            pop
                  ВХ
            pop
                  ΑX
            pop
      ret
                  ENDP
HEX_BYTE_PRINT
HEX_WORD_PRINT
                  PROC
            push AX
            push AX
            mov AL, AH
            call HEX_BYTE_PRINT
            pop AX
            call HEX_BYTE_PRINT
            pop AX
      ret
HEX_WORD_PRINT
                  ENDP
FREE_MEM
            PROC
            push
                  AX
            push
                  BX
            push
                  DX
                  BX, offset PROG_END
            mov
                  BX, 100h
            add
            shr
                  BX, 1
            shr
                  BX, 1
                  BX, 1
            shr
                  BX, 1 ; to paragraph
            shr
                  AH, 4Ah
            mov
            int
                  21h
                  DX
            pop
                  BX
            pop
            pop
                  ΑX
      ret
FREE_MEM
             ENDP
PRINT_AVL_MEMORY PROC NEAR
          push
                   AX
          push
                   BX
                   \mathsf{DX}
          push
          push
                   SI
```

AX, AX

xor

```
int 12h
                  DX, offset STR_AVL_MEMORY
          mov
          call
                  WRITE
                        DX, DX
            xor
                  DEC_WORD_PRINT
          call
        call
                KBYTES_PRINT
                  SI
          pop
                  DX
          pop
                  BX
          pop
                  \mathsf{AX}
          pop
      ret
PRINT_AVL_MEMORY ENDP
PRINT_EXT_MEMORY PROC NEAR
            push AX
            push BX
            push DX
            push SI
                  AL, 30h
            mov
                  70h, AL
            out
            in
                        AL, 71h
                  BL, AL
            mov
                  AL, 31h
            mov
                  70h, AL
            out
                       AL, 71h
            in
                  AH, AL
            mov
                  AL, BL
            mov
                  DX, offset STR_EXT_MEMORY
        mov
            call WRITE
                        DX, DX
            xor
            call DEC_WORD_PRINT
                KBYTES PRINT
        call
            pop
                  SI
                  \mathsf{DX}
            pop
            pop
                  ВХ
            pop
                  AX
      ret
PRINT_EXT_MEMORY ENDP
```

PROC

PRINT_MCB

```
push AX
      push
            BX
      push
            CX
      push DX
      push
            ES
      push
            SI
            AH, 52h
      \text{mov}
      int
            21h
            AX, ES:[BX-2]
      mov
            ES, AX
      \text{mov}
            CX, CX
      xor
NEXT_MCB:
      inc
            \mathsf{CX}
      mov
            DX, offset STR_MCB_NUM
      push
            CX
      call
            WRITE
      mov
            AX, CX
      xor
            DX, DX
      call DEC_WORD_PRINT
OWNER_START:
      mov
            DX, offset STR_INFO
      call WRITE
            AX, AX
      xor
            AL, ES:[0h]
      mov
      push AX
      mov
            AX, ES:[1h]
      cmp
            AX, 0h
                   PRINT_FREE
      je
      cmp
            AX, 6h
                   PRINT_XMS
      jе
            AX, 7h
      cmp
                   PRINT_TM
      je
      cmp
            AX, 8h
      je
                   PRINT_DOS
            AX, 0FFFAh
      cmp
      je
                   PRINT_386CB
            AX, 0FFFDh
      cmp
      je
                   PRINT_386B
      cmp
            AX, 0FFFEh
      je
                   PRINT_386
            DX, DX
      xor
            HEX_WORD_PRINT
      call
      jmp
            AREA_SIZE_START
```

```
DX, offset STR_FREE
      mov
      jmp
            OWNER_END
PRINT_XMS:
            DX, offset STR_XMS
      mov
            OWNER_END
      jmp
PRINT_TM:
            DX, offset STR_TM
      mov
            OWNER_END
      jmp
PRINT_DOS:
            DX, offset STR_DOS
      mov
            OWNER_END
      jmp
PRINT 386CB:
            DX, offset STR_386CB
      mov
      jmp
            OWNER_END
PRINT_386B:
            DX, offset STR_386B
      mov
      jmp
            OWNER_END
PRINT_386:
            DX, offset STR_386
      mov
OWNER_END:
      call WRITE
AREA_SIZE_START:
            DX, offset STR_AREA_SIZE
      mov
      call WRITE
            AX, ES:[3h]
      mov
            BX, 10h
      mov
      mul
            BX
      call
           DEC_WORD_PRINT
            CX, 8
      mov
            SI, SI
      xor
            DX, offset END_LINE
      mov
      call WRITE
LAST_BYTES_START:
              DL, ES:[SI + 8h]
      mov
              AH, 02h
      mov
      int
              21h
      inc
              SI
              LAST_BYTES_START
      loop
              AX, ES:[3h]
      mov
              BX, ES
      mov
      add
              BX, AX
              BX
      inc
              ES, BX
      \text{mov}
```

```
\mathsf{AX}
             pop
                      \mathsf{CX}
             pop
             cmp
                      AL, 5Ah
                      ENDING
             je
                      DX, offset END_LINE
             mov
             call
                      WRITE
             jmp
                      NEXT_MCB
      ENDING:
                      SI
             pop
             pop
                      ES
             pop
                      DX
                      \mathsf{CX}
             pop
                      BX
             pop
             pop
                      \mathsf{AX}
      ret
PRINT_MCB
                    ENDP
BEGIN:
         call
                    FREE_MEM
         call
                  PRINT_AVL_MEMORY
             call PRINT_EXT_MEMORY
         call
                  PRINT_MCB
                  AL, AL
         xor
                    AH, 4Ch
           mov
                    21h
           int
      PROG_END:
LAB
         ends
end
         START
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR33.ASM

```
LAB segment
            ASSUME CS: LAB, DS: LAB, ES: NOTHING, SS: NOTHING
            org 100h
START: jmp BEGIN
STR_AVL_MEMORY
                       db "Avl mem: $"
                       db "Ext mem: $"
STR_EXT_MEMORY
STR MCB NUM
                       db "MCB num: $"
                       db " Area size: $"
STR AREA SIZE
END LINE
                       db 0Dh, 0Ah, "$"
                    db " kb", 0Dh, 0Ah, "$"
KBYTES
                            db " driver's top memory$"
STR_TM
                       db " MS DOS$"
STR_DOS
                       db 0Dh, 0Ah, "Block is $"
STR INFO
                       db "free$"
STR FREE
                       db " OS XMS UMB$"
STR_XMS
                       db " busy by 386MAX UMB$"
STR_386CB
                       db " blocked by 386MAX$"
STR_386B
                       db " 386MAX UMB$"
STR_386
WRITE
            PROC near
            push AX
            mov
                  AH, 09h
            int
                        21h
                 AX
            pop
      ret
WRITE
           ENDP
KBYTES_PRINT
                PROC
                        near
                DX
        push
                DX, offset KBYTES
        mov
                WRITE
        call
                DX
        pop
    ret
KBYTES_PRINT
                ENDP
DEC_WORD_PRINT
                  PROC; IN: AX
            push AX
            push
                 CX
            push
                 DX
            push
                 BX
                  BX, 10
            mov
                  CX, CX
            xor
      NUM:
            div
                  BX
            push DX
                  DX, DX
            xor
            inc
                  \mathsf{CX}
```

```
AX, 0h
              cmp
              jnz
                     NUM
       PRINT_NUM:
                     \mathsf{DX}
              pop
              or
                            DL, 30h
              mov
                     AH, 02h
              int
                     21h
              loop
                     PRINT_NUM
              pop
                     BX
                     DX
              pop
                     \mathsf{CX}
              pop
                     AX
              pop
       ret
DEC_WORD_PRINT
                     ENDP
HEX_BYTE_PRINT
                     PROC
                     AX
              push
                     \mathsf{BX}
              push
              push
                     DX
              mov
                     AH, 0
                     BL, 10h
              mov
              div
                     BL
                     DX, AX
              mov
                     AH, 02h
              mov
              cmp
                     DL, 0Ah
              jl
                            PRINT
              add
                     DL, 07h
       PRINT:
              add
                     DL, '0'
                     21h;
              int
                     DL, DH
              \text{mov}
                     DL, 0Ah
              cmp
              jl
                            PRINT_EXT
              add
                     DL, 07h
       PRINT_EXT:
                     DL, '0'
              add
              int
                     21h;
                     \mathsf{DX}
              pop
              pop
                     \mathsf{BX}
              pop
                     AX
       ret
HEX_BYTE_PRINT
                     ENDP
HEX_WORD_PRINT
                     PROC
              push AX
              push AX
              mov AL, AH
```

```
call HEX_BYTE_PRINT
             pop AX
             call HEX_BYTE_PRINT
             pop AX
      ret
HEX_WORD_PRINT
                   ENDP
FREE_MEM
             PROC
             push
                   \mathsf{AX}
             push
                   BX
             push
                   DX
                   BX, offset PROG_END
             mov
             add
                   BX, 100h
             shr
                   BX, 1
             shr
                   BX, 1
                   BX, 1
             shr
             shr
                   BX, 1; to paragraph
             mov
                   AH, 4Ah
             int
                   21h
             pop
                   DX
                   BX
             pop
             pop
                   AX
      ret
FREE_MEM
              ENDP
             PROC
ADD_MEM
             push AX
             push
                   BX
             push
                   DX
             mov
                   BX, 1000h
                   AH, 48h
             mov
             int
                   21h
                   \mathsf{DX}
             pop
             pop
                   BX
             pop
                   AX
      ret
ADD_MEM
              ENDP
 PRINT_AVL_MEMORY
                          PROC NEAR
                    \mathsf{AX}
           push
           push
                    BX
           push
                    DX
           push
                    SI
           xor
                   AX, AX
                   12h
             int
           mov
                    DX, offset STR_AVL_MEMORY
           call
                    WRITE
                          DX, DX
             xor
```

```
call
                  DEC_WORD_PRINT
        call
                KBYTES_PRINT
                  SI
          pop
                  DX
          pop
                  ВХ
          pop
          pop
                  AX
PRINT_AVL_MEMORY ENDP
PRINT_EXT_MEMORY PROC NEAR
           push AX
           push BX
           push DX
           push
                 SI
                 AL, 30h
           mov
           out
                 70h, AL
           in
                       AL, 71h
                 BL, AL
           mov
                 AL, 31h
           mov
           out
                 70h, AL
                     AL, 71h
           in
                 AH, AL
           mov
                 AL, BL
           mov
                 DX, offset STR_EXT_MEMORY
        mov
           call WRITE
                       DX, DX
           xor
           call DEC_WORD_PRINT
                KBYTES_PRINT
        call
                 SI
           pop
                 DX
           pop
                 BX
           pop
                 AX
           pop
PRINT_EXT_MEMORY ENDP
PRINT_MCB
                 PROC
           push AX
           push BX
           push CX
           push DX
           push
                 ES
                 SI
           push
           mov
                 AH, 52h
           int
                 21h
           mov
                 AX, ES:[BX-2]
                 ES, AX
           mov
                 CX, CX
           xor
```

```
NEXT_MCB:
            CX
      inc
            DX, offset STR_MCB_NUM
      mov
      push CX
      call WRITE
      mov
            AX, CX
            DX, DX
      xor
      call
            DEC_WORD_PRINT
OWNER_START:
            DX, offset STR_INFO
      mov
      call WRITE
      xor
            AX, AX
            AL, ES:[0h]
      mov
      push
            AX
            AX, ES:[1h]
      mov
      cmp
            AX, 0h
                  PRINT_FREE
      je
      cmp
            AX, 6h
      je
                  PRINT_XMS
      cmp
            AX, 7h
      je
                  PRINT_TM
            AX, 8h
      cmp
      je
                  PRINT_DOS
            AX, 0FFFAh
      cmp
      jе
                  PRINT_386CB
      cmp
            AX, 0FFFDh
                  PRINT_386B
      je
      \mathsf{cmp}
            AX, 0FFFEh
                  PRINT_386
      je
            DX, DX
      xor
      call
            HEX_WORD_PRINT
            AREA SIZE START
      jmp
PRINT FREE:
            DX, offset STR_FREE
      mov
            OWNER_END
      jmp
PRINT_XMS:
            DX, offset STR_XMS
      mov
            OWNER_END
      jmp
PRINT_TM:
      mov
            DX, offset STR_TM
            OWNER_END
      jmp
PRINT_DOS:
            DX, offset STR_DOS
      mov
            OWNER_END
      jmp
PRINT_386CB:
            DX, offset STR_386CB
      {\sf mov}
      jmp
            OWNER_END
PRINT_386B:
            DX, offset STR_386B
      mov
      jmp
            OWNER END
PRINT_386:
            DX, offset STR_386
      mov
```

```
OWNER_END:
            call WRITE
      AREA_SIZE_START:
                   DX, offset STR_AREA_SIZE
            mov
            call WRITE
                   AX, ES:[3h]
            mov
            mov
                   BX, 10h
            mul
                   BX
                  DEC_WORD_PRINT
            call
            mov
                   CX, 8
                   SI, SI
            xor
                   DX, offset END_LINE
            mov
            call WRITE
      LAST_BYTES_START:
                     DL, ES:[SI + 8h]
            mov
            mov
                     AH, 02h
            int
                     21h
            inc
                     SI
            loop
                     LAST_BYTES_START
                     AX, ES:[3h]
            mov
            mov
                     BX, ES
            add
                     BX, AX
            inc
                     BX
                     ES, BX
            mov
            pop
                     ΑX
                     \mathsf{CX}
            pop
            cmp
                     AL, 5Ah
                     ENDING
            jе
                     DX, offset END_LINE
            mov
            call
                     WRITE
                     NEXT_MCB
            jmp
      ENDING:
            pop
                     SI
                     ES
            pop
                     DX
            pop
                     \mathsf{CX}
            pop
            pop
                     BX
                     AX
            pop
      ret
PRINT_MCB
                   ENDP
BEGIN:
        call
                   FREE_MEM
        call
                 ADD_MEM
        call
                 PRINT AVL MEMORY
            call PRINT_EXT_MEMORY
        call
                 PRINT_MCB
```

xor AL, AL mov AH, 4Ch int 21h

PROG_END:

LAB ends end START

приложение д

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR34.ASM

```
LAB segment
```

```
ASSUME CS: LAB, DS: LAB, ES: NOTHING, SS: NOTHING org 100h
```

START: jmp BEGIN

```
STR_AVL_MEMORY
                      db "Avl mem: $"
                      db "Ext mem: $"
STR EXT MEMORY
                          "MCB num: $"
STR_MCB_NUM
                      db
                      db " Area size: $"
STR_AREA_SIZE
                      db 0Dh, 0Ah, "$"
END_LINE
                   db " kb", 0Dh, 0Ah, "$"
KBYTES
                            db " driver's top memory$"
STR_TM
STR DOS
                      db " MS DOS$"
STR_INFO
                      db 0Dh, 0Ah, "Block is $"
                      db "free$"
STR_FREE
                      db " OS XMS UMB$"
STR_XMS
STR_386CB
                      db "busy by 386MAX UMB$"
                      db " blocked by 386MAX$"
STR 386B
                      db " 386MAX UMB$"
STR_386
```

WRITE PROC near push AX

mov AH, 09h int 21h

pop AX

ret

WRITE ENDP

KBYTES_PRINT PROC near

push DX

mov DX, offset KBYTES

call WRITE pop DX

ret

KBYTES_PRINT ENDP

DEC_WORD_PRINT PROC ; IN: AX

push AX
push CX
push DX
push BX

mov BX, 10

```
CX, CX
             xor
      NUM:
             div
                    ВХ
             push
                    DX
             xor
                    DX, DX
             inc
                    \mathsf{CX}
                    AX, 0h
             cmp
             jnz
                    NUM
      PRINT_NUM:
             pop
                    DX
                           DL, 30h
             or
             \text{mov}
                    AH, 02h
             int
                    21h
             loop
                    PRINT_NUM
             pop
                    BX
                    DX
             pop
             pop
                    \mathsf{CX}
             pop
                    ΑX
      ret
DEC_WORD_PRINT
                    ENDP
                    PROC
HEX_BYTE_PRINT
             push AX
                    \mathsf{BX}
             push
             push
                    DX
                    AH, 0
             mov
             mov
                    BL, 10h
                    BL
             div
                    DX, AX
             mov
                    AH, 02h
             mov
                    DL, 0Ah
             cmp
             jl
                           PRINT
             add
                    DL, 07h
      PRINT:
                    DL, '0'
             add
             int
                    21h;
                    DL, DH
             {\sf mov}
                    DL, 0Ah
             cmp
                           PRINT_EXT
             jl
             add
                    DL, 07h
      PRINT_EXT:
                    DL, '0'
             add
```

```
int
                   21h;
                   DX
             pop
                   ВХ
             pop
                   ΑX
             pop
      ret
                   ENDP
HEX_BYTE_PRINT
HEX_WORD_PRINT
                   PROC
             push AX
             push AX
             mov AL, AH
             call HEX_BYTE_PRINT
             pop AX
             call HEX_BYTE_PRINT
             pop AX
      ret
HEX_WORD_PRINT
                   ENDP
\mathsf{FREE}\_\mathsf{MEM}
             PROC
             push
                   AX
             push
                   ВХ
             push
                   DX
                   BX, offset PROG_END
             mov
                   BX, 100h
             add
             shr
                   BX, 1
             shr
                   BX, 1
             shr
                   BX, 1
                   BX, 1 ; to paragraph
             shr
                   AH, 4Ah
             mov
                   21h
             int
                   DX
             pop
                   BX
             pop
             pop
                   ΑX
      ret
FREE_MEM
              ENDP
ADD_MEM
             PROC
             push
                      AX
             push
                      BX
                      \mathsf{DX}
             push
                      BX, 1000h
             mov
```

AH, 48h

mov

```
21h
             int
                      \mathsf{DX}
             pop
                      ВХ
             pop
                      \mathsf{AX}
             pop
      ret
              ENDP
ADD_MEM
PRINT_AVL_MEMORY PROC NEAR
           push
                    \mathsf{AX}
           push
                    BX
                    DX
           push
                    SI
           push
           xor
                    AX, AX
             int
                    12h
           mov
                    DX, offset STR_AVL_MEMORY
                    WRITE
           call
                          DX, DX
             xor
           call
                    DEC_WORD_PRINT
         call
                  KBYTES_PRINT
           pop
                    SI
           pop
                    DX
                    ВХ
           pop
                    \mathsf{AX}
           pop
      ret
PRINT_AVL_MEMORY ENDP
PRINT_EXT_MEMORY PROC NEAR
             push AX
             push
                   BX
             push
                   DX
             push
                   SI
                    AL, 30h
             mov
                    70h, AL
             out
             in
                          AL, 71h
                    BL, AL
             mov
                    AL, 31h
             mov
                    70h, AL
             out
             in
                          AL, 71h
                    AH, AL
             mov
                    AL, BL
             mov
```

```
DX, offset STR_EXT_MEMORY
        mov
            call WRITE
            xor
                        DX, DX
            call DEC_WORD_PRINT
                KBYTES_PRINT
        call
            pop
                  SI
                  DX
            pop
                  BX
            pop
            pop
                  \mathsf{AX}
      ret
PRINT_EXT_MEMORY ENDP
PRINT_MCB
                  PROC
            push AX
            push BX
            push CX
            push DX
            push ES
            push SI
                  AH, 52h
            mov
            int
                  21h
                  AX, ES:[BX-2]
            mov
                  ES, AX
            mov
                  CX, CX
            xor
      NEXT_MCB:
                  \mathsf{CX}
            inc
                  DX, offset STR_MCB_NUM
            mov
            push CX
            call WRITE
                  AX, CX
            mov
            xor
                  DX, DX
            call DEC_WORD_PRINT
      OWNER_START:
            mov
                  DX, offset STR_INFO
            call WRITE
            xor
                  AX, AX
                  AL, ES:[0h]
            mov
            push AX
            mov
                  AX, ES:[1h]
            cmp
                  AX, 0h
            je
                        PRINT_FREE
            cmp
                  AX, 6h
                        PRINT_XMS
            je
```

```
AX, 7h
      cmp
      je
                  PRINT_TM
            AX, 8h
      cmp
                  PRINT_DOS
      je
            AX, 0FFFAh
      cmp
      je
                  PRINT_386CB
            AX, 0FFFDh
      cmp
      jе
                  PRINT_386B
            AX, 0FFFEh
      cmp
      je
                  PRINT_386
            DX, DX
      xor
      call HEX_WORD_PRINT
      jmp
            AREA_SIZE_START
PRINT_FREE:
            DX, offset STR_FREE
      mov
      jmp
            OWNER_END
PRINT_XMS:
            DX, offset STR_XMS
      mov
      jmp
            OWNER END
PRINT_TM:
      mov
            DX, offset STR_TM
            OWNER_END
      jmp
PRINT_DOS:
            DX, offset STR_DOS
      \text{mov}
            OWNER_END
      jmp
PRINT_386CB:
      \text{mov}
            DX, offset STR_386CB
            OWNER_END
      jmp
PRINT_386B:
            DX, offset STR_386B
      mov
            OWNER_END
      jmp
PRINT_386:
      mov
            DX, offset STR_386
OWNER_END:
      call WRITE
AREA_SIZE_START:
            DX, offset STR_AREA_SIZE
      mov
      call WRITE
      mov
            AX, ES:[3h]
      mov
            BX, 10h
      mul
            BX
      call
            DEC_WORD_PRINT
            CX, 8
      mov
            SI, SI
      xor
```

```
DX, offset END_LINE
             mov
             call WRITE
      LAST_BYTES_START:
                      DL, ES:[SI + 8h]
             mov
             mov
                      AH, 02h
             int
                      21h
             inc
                      SI
                      LAST_BYTES_START
             loop
                      AX, ES:[3h]
             mov
             mov
                      BX, ES
             add
                      BX, AX
             inc
                      BX
             mov
                      ES, BX
                      \mathsf{AX}
             pop
             pop
                      \mathsf{CX}
             cmp
                      AL, 5Ah
                      ENDING
             je
             mov
                      DX, offset END_LINE
             call
                      WRITE
                      NEXT_MCB
             jmp
      ENDING:
                      SI
             pop
                      ES
             pop
                      DX
             pop
             pop
                      \mathsf{CX}
                      BX
             pop
                      \mathsf{AX}
             pop
      ret
PRINT_MCB
                   ENDP
BEGIN:
                  ADD_MEM
         call
         call
                   FREE_MEM
                  PRINT_AVL_MEMORY
             call PRINT_EXT_MEMORY
                  PRINT_MCB
         call
         xor
                  AL, AL
                    AH, 4Ch
           mov
```

int

21h

PROG_END:

LAB ends end START