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

Студент гр. 8381	 Облизанов А.Д
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. Структура МСВ представлена на рис. 1.

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

По сегментному адресу и размеру участка памяти, контролируемого этим МСВ можно определить местоположение следующего МСВ 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: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. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта.

Выполнение работы.

Написание работы производилось на базе операционной системы Windows 10 в редакторе Visual Code. Сборка, отладка производились на базе эмулятора DOSBox 0.74-3.

Был написан текст исходного .COM модуля с именем lr31.com, который обеспечивает вывод на экран

- Размера доступной памяти (в килобайтах)
- Размера расширенной памяти (в килобайтах)
- Данных с МСВ блоков:
 - о «Владелец» участка (или адрес PSP владельца участка)
 - о Размер участка памяти
 - о SD/SC/Другое (последние 8 байт MCB)

Вывод программы представлен на рис. 2.

```
D:\>lr31.com
Available memory: 640 kbytes
Extended memory: 15360 kbytes
MCB number: 1
Block is MS DOS Area size: 16
MCB number: 2
Block is free Area size: 64
MCB number: 3
               Area size: 256
Block is 0040
MCB number: 4
Block is 0192
               Area size: 144
MCB number: 5
Block is 0192
                Area size: 648912
LR31
```

Рисунок 2 – Выполнение lr31.com

Как видно из рисунка, программа занимает всю доступную память.

Далее был написан текст исходного модуля lr32.com, в котором программа освобождает память, которую не занимает. Результат выполнения программы представлен на рис. 3.

```
D:\>lr32.com
A∨ailable memory: 640 kbytes
Extended memory: 15360 kbytes
MCB number: 1
Block is MS DOS Area size: 16
MCB number: 2
Block is free
                Area size: 64
MCB number: 3
Block is 0040
              Area size: 256
MCB number: 4
Block is 0192
               Area size: 144
MCB number: 5
Block is 0192
                Area size: 1120
LR32
MCB number: 6
Block is free Area size: 647776
```

Рисунок 3 – Выполнение lr32.com

Как видно из рисунка, программа не занимает всю память. Освобожденная память относится к шестому блоку управления памятью, который является свободным.

Далее был написан текст исходного модуля lr33.com, в котором программа освобождает память, которую не занимает, а после этого запрашивает дополнительно 64 кбайт. Выполнение показано на рис. 4.

```
D:\>lr33.com
Available memory: 640 kbytes
Extended memory: 15360 kbytes
MCB number: 1
Block is MS DOS Area size: 16
MCB number: 2
Block is free Area size: 64
MCB number: 3
Block is 0040
                Area size: 256
MCB number: 4
Block is 0192
                Area size: 144
MCB number: 5
Block is 0192
                Area size: 1136
LR33
MCB number: 6
Block is 0192
                Area size: 65536
LR33
MCB number: 7
Block is free
                Area size: 582208
```

Рисунок 4 – Выполнение lr33.com

Как видно из рисунка, дополнительно выделенная память для программы относится к 6-му блоку, а к 5-му блоку относится память, выделенная программе после освобождения памяти. Сегментный адрес PSP владельца участка памяти 5-го и 6-го блока совпадают.

Далее был написан текст исходного модуля lr34.com, в котором программа сначала запрашивает дополнительно 64 кбайт, а затем освобождает память. Выполнение показано на рис. 5.

D:\>lr34.com Available memory: 640 kbytes Extended memory: 15360 kbytes MCB number: 1 Block is MS DOS Area size: 16 MCB number: 2 Block is free Area size: 64 MCB number: 3 Area size: 256 Block is 0040 MCB number: 4 Block is 0192 Area size: 144 MCB number: 5 Block is 0192 Area size: 1136 LR34 MCB number: 6 Block is free Area size: 647760

Рисунок 5 – Выполнение lr34.com

Как видно из рисунка, дополнительная память не была выделена для программы. Выделенная под программу память относится к 5-му блоку и имеет объем меньше 64 кбайт.

Контрольные вопросы

1. Что означает «доступный объем» памяти?

Максимальный объем памяти, который может быть доступен для запуска и выполнения программ.

Примечание: в модуле lr34.com программа изначально занимает весь доступный объем памяти, из-за чего невозможно выделить дополнительно 64 кбайт.

2. Где МСВ блок вашей программы в списке?

- Lr31.com, lr32.com, lr34.com MCB блок программы пятый
- Lr33.com MCB блок программы пятый, но также шестой для дополнительно выделенных во время работы программы 64 кбайт.

3. Какой размер памяти занимает программа в каждом случае?

- Lr31.com. Программа занимает 648 912 байт, то есть всю доступную ей память.
- Lr32.com. Программа занимает 1120 байт после освобождения памяти. Для освобождения в конец программы была добавлена метка, по смещению которой можно определить объем памяти, необходимый для программы, добавляя некоторой запас.
- Lr33.com. Программа занимает 1120 байт после освобождения памяти (пятый блок), а также занимает дополнительно 65536 байт после выделения (шестой блок).
- Lr34.com. Программа занимает 1120 байт после неудачной попытки выделения дополнительной памяти и (удачного) освобождения. Причина того, что память не была выделена, в 1-м контрольном вопросе.

Выводы.

В ходе выполнения лабораторной работы был изучен список блоков управления памятью, а также 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

;DATA

AVL_MEMORY_INFO db "Available memory: \$" EXT_MEMORY_INFO db "Extended memory: \$"

MCB_NUM_INFO db "MCB number: \$"
AREA_SIZE_INFO db " Area size: \$"

END_LINE db 0Dh, 0Ah, "\$"

KBYTES db "kbytes", 0Dh, 0Ah, "\$"

OWNER_INFO db 0Dh, 0Ah, "Block is \$"

OWNER FREE db "free\$"

OWNER_XMS db " OS XMS UMB\$"

OWNER_TM db " driver's top memory\$"

OWNER_DOS db " MS DOS\$"

OWNER_386CB db " busy by 386MAX UMB\$" OWNER_386B db " blocked by 386MAX\$"

OWNER 386 db " 386MAX UMB\$"

; There is some CUSTOM procedures...

PRINT_STRING PROC near

push AX

mov AH, 09h int 21h

pop AX

ret

PRINT_STRING ENDP

KBYTES_PRINT PROC near

push DX

mov DX, offset KBYTES

call PRINT_STRING

pop DX

ret

KBYTES_PRINT ENDP

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

```
int
                  21h;
                  DL, DH
            mov
                  DL, 0Ah
            cmp
            jl
                         PRINT_EXT
            add
                  DL, 07h
      PRINT_EXT:
                  DL, '0'
            add
            int
                  21h;
                  DX
            pop
                  ВХ
            pop
                  \mathsf{AX}
            pop
      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
    ; here is the action...
PRINT_AVL_MEMORY PROC NEAR
          push
                   \mathsf{AX}
          push
                   BX
                   DX
          push
                   SI
          push
          xor
                  AX, AX
            int
                  12h
                   DX, offset AVL_MEMORY_INFO
          mov
          call
                   PRINT_STRING
            xor
                         DX, DX
          call
                   DEC_WORD_PRINT
        call
                 KBYTES_PRINT
          pop
                   SI
                   DX
          pop
```

```
\mathsf{BX}
          pop
                  ΑX
          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
            mov
                  BL, AL
                  AL, 31h
            mov
            out
                  70h, AL
            in
                       AL, 71h
                 AH, AL
            mov
            mov
                 AL, BL
                  DX, offset EXT_MEMORY_INFO
            call PRINT_STRING
            xor
                       DX, DX
            call DEC_WORD_PRINT
        call KBYTES_PRINT
                  SI
            pop
                  DX
            pop
                  BX
            pop
            pop
                  \mathsf{AX}
      ret
PRINT_EXT_MEMORY ENDP
                  PROC
PRINT MCB
            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:
      inc
            \mathsf{CX}
            DX, offset MCB_NUM_INFO
      mov
      push
            \mathsf{CX}
      call
            PRINT_STRING
            AX, CX
      mov
            DX, DX
      xor
            DEC_WORD_PRINT
      call
OWNER_START:
            DX, offset OWNER_INFO
      mov
      call PRINT_STRING
      xor
            AX, AX
            AL, ES:[0h]
      mov
      push
            AX
      mov
            AX, ES:[1h]
      cmp
            AX, 0h
      je
                  PRINT_FREE
      cmp
            AX, 6h
      je
                  PRINT_XMS
            AX, 7h
      cmp
      jе
                  PRINT_TM
      cmp
            AX, 8h
                  PRINT_DOS
      je
            AX, 0FFFAh
      cmp
      je
                  PRINT_386CB
            AX, 0FFFDh
      cmp
      je
                  PRINT_386B
            AX, 0FFFEh
      cmp
                  PRINT_386
      je
            DX, DX
      xor
           HEX_WORD_PRINT
      call
      jmp
            AREA_SIZE_START
PRINT FREE:
            DX, offset OWNER_FREE
      mov
            OWNER_END
      jmp
PRINT_XMS:
      mov
            DX, offset OWNER_XMS
      jmp
            OWNER_END
PRINT_TM:
            DX, offset OWNER_TM
      mov
      jmp
            OWNER_END
PRINT_DOS:
            DX, offset OWNER_DOS
      mov
```

```
OWNER_END
      jmp
PRINT_386CB:
      mov
            DX, offset OWNER_386CB
            OWNER_END
      jmp
PRINT_386B:
            DX, offset OWNER_386B
      mov
            OWNER_END
      jmp
PRINT_386:
            DX, offset OWNER_386
      mov
OWNER_END:
      call PRINT_STRING
AREA_SIZE_START:
      mov
            DX, offset AREA_SIZE_INFO
      call PRINT_STRING
            AX, ES:[3h]
      mov
      mov
            BX, 10h
      mul
            BX
      call
           DEC_WORD_PRINT
      mov
            CX, 8
            SI, SI
      xor
            DX, offset END_LINE
      mov
      call PRINT_STRING
LAST_BYTES_START:
              DL, ES:[SI + 8h]
      mov
      mov
              AH, 02h
      int
              21h
      inc
              SI
              LAST_BYTES_START
      loop
              AX, ES:[3h]
      mov
              BX, ES
      mov
              BX, AX
      add
      inc
              BX
              ES, BX
      mov
              AX
      pop
              CX
      pop
      cmp
              AL, 5Ah
      je
              ENDING
      mov
              DX, offset END_LINE
              PRINT_STRING
      call
              NEXT_MCB
      jmp
ENDING:
              SI
      pop
```

pop ES
pop DX
pop CX
pop BX
pop AX
ret

PRINT_MCB ENDP

BEGIN:

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 Б

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR32.ASM

LAB segment

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

org 100h

START: jmp BEGIN

;DATA

db "Available memory: \$" AVL_MEMORY_INFO

db "Extended memory: \$" EXT_MEMORY_INFO

"MCB number: \$" MCB_NUM_INFO db

AREA_SIZE_INFO db Area size: \$"

db 0Dh, 0Ah, "\$" END_LINE

db "kbytes", 0Dh, 0Ah, "\$" **KBYTES**

db 0Dh, 0Ah, "Block is \$" OWNER_INFO

db "free\$" OWNER FREE

db " OS XMS UMB\$" OWNER_XMS

OWNER_TM db " driver's top memory\$"

db " MS DOS\$" OWNER DOS

db "busy by 386MAX UMB\$" OWNER_386CB db " blocked by 386MAX\$" OWNER_386B

db " 386MAX UMB\$" OWNER 386

; There is some CUSTOM procedures...

PRINT_STRING PROC near

push AX

AH, 09h mov 21h int

pop AX

ret

PRINT_STRING **ENDP**

KBYTES_PRINT PROC near

> push DX

DX, offset KBYTES mov

call PRINT_STRING

DX pop

ret

KBYTES_PRINT **ENDP**

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

```
int
                   21h;
                   DL, DH
            mov
                   DL, 0Ah
             cmp
             jl
                         PRINT_EXT
             add
                   DL, 07h
      PRINT_EXT:
                   DL, '0'
             add
             int
                   21h;
                   DX
             pop
                   ВХ
             pop
                   \mathsf{AX}
             pop
      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
            PROC
FREE_MEM
             push AX
             push
                   \mathsf{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
                   21h
             int
                   DX
             pop
             pop
                   BX
             pop
                   \mathsf{AX}
      ret
FREE_MEM
            ENDP
```

```
; here is the action...
PRINT_AVL_MEMORY PROC NEAR
                  AX
          push
                  BX
          push
                  DX
          push
          push
                  SI
                  AX, AX
          xor
            int
                  12h
                  DX, offset AVL_MEMORY_INFO
          mov
          call
                  PRINT_STRING
                        DX, DX
            xor
          call
                  DEC_WORD_PRINT
        call
                KBYTES_PRINT
          pop
                  SI
          pop
                  DX
          pop
                  BX
          pop
                  AX
      ret
PRINT_AVL_MEMORY ENDP
PRINT_EXT_MEMORY PROC NEAR
            push AX
            push
                 BX
            push DX
                 SI
            push
                  AL, 30h
            mov
                  70h, AL
            out
            in
                        AL, 71h
                  BL, AL
            mov
                  AL, 31h
            mov
            out
                  70h, AL
                        AL, 71h
            in
                  AH, AL
            mov
                  AL, BL
            mov
                  DX, offset EXT_MEMORY_INFO
        mov
            call PRINT_STRING
            xor
                        DX, DX
            call DEC_WORD_PRINT
                KBYTES_PRINT
        call
```

```
SI
             pop
             pop
                   \mathsf{DX}
                   ВХ
             pop
                   AX
             pop
      ret
PRINT_EXT_MEMORY ENDP
PRINT_MCB
                   PROC
             push AX
             push
                   BX
             push CX
             push DX
             push
                   ES
             push
                  SI
             mov
                   AH, 52h
             int
                   21h
                   AX, ES:[BX-2]
             mov
                   ES, AX
             mov
                   CX, CX
             xor
      NEXT_MCB:
             inc
                   \mathsf{CX}
             mov
                   DX, offset MCB_NUM_INFO
             push CX
             call PRINT_STRING
                   AX, CX
             mov
                   DX, DX
             xor
                   DEC_WORD_PRINT
             call
      OWNER_START:
                   DX, offset OWNER_INFO
             mov
             call PRINT_STRING
             xor
                   AX, AX
                   AL, ES:[0h]
             mov
                   AX
             push
             mov
                   AX, ES:[1h]
             \mathsf{cmp}
                   AX, 0h
                          PRINT_FREE
             je
             cmp
                   AX, 6h
             je
                          PRINT_XMS
                   AX, 7h
             cmp
             je
                          PRINT_TM
             \mathsf{cmp}
                   AX, 8h
             je
                          PRINT_DOS
                   AX, 0FFFAh
             cmp
```

```
je
                  PRINT_386CB
      cmp
           AX, 0FFFDh
                  PRINT_386B
      je
            AX, 0FFFEh
      cmp
      je
                  PRINT_386
      xor
            DX, DX
      call HEX_WORD_PRINT
            AREA_SIZE_START
      jmp
PRINT_FREE:
      mov
            DX, offset OWNER_FREE
            OWNER END
      jmp
PRINT_XMS:
      mov
            DX, offset OWNER_XMS
           OWNER_END
      jmp
PRINT_TM:
      mov
            DX, offset OWNER_TM
           OWNER_END
      jmp
PRINT_DOS:
      mov
            DX, offset OWNER_DOS
      jmp
            OWNER_END
PRINT_386CB:
      mov
            DX, offset OWNER_386CB
      jmp
           OWNER_END
PRINT_386B:
            DX, offset OWNER_386B
      mov
           OWNER_END
      jmp
PRINT_386:
            DX, offset OWNER_386
      mov
OWNER_END:
      call PRINT_STRING
AREA_SIZE_START:
      mov
            DX, offset AREA_SIZE_INFO
      call PRINT_STRING
      mov
           AX, ES:[3h]
            BX, 10h
      mov
      mul
            BX
      call DEC_WORD_PRINT
            CX, 8
      mov
            SI, SI
      xor
            DX, offset END_LINE
      mov
      call PRINT_STRING
LAST_BYTES_START:
              DL, ES:[SI + 8h]
     mov
```

```
AH, 02h
             mov
             int
                      21h
             inc
                      SI
             loop
                      LAST_BYTES_START
                      AX, ES:[3h]
             mov
                      BX, ES
             mov
                      BX, AX
             add
                      ВХ
             inc
                      ES, BX
             mov
                      \mathsf{AX}
             pop
                      \mathsf{CX}
             pop
                      AL, 5Ah
             cmp
                      ENDING
             je
             mov
                      DX, offset END_LINE
             call
                      PRINT_STRING
             jmp
                      NEXT_MCB
      ENDING:
                      SI
             pop
                      ES
             pop
                      \mathsf{DX}
             pop
                      \mathsf{CX}
             pop
                      BX
             pop
                      AX
             pop
      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
         START
end
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR33.ASM

```
LAB segment
```

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

org 100h

START: jmp BEGIN

;DATA

AVL_MEMORY_INFO db "Available memory: \$"

EXT_MEMORY_INFO db "Extended memory: \$"

MCB_NUM_INFO db "MCB number: \$"

AREA_SIZE_INFO db " Area size: \$"

END_LINE db 0Dh, 0Ah, "\$"

KBYTES db "kbytes", 0Dh, 0Ah, "\$"

OWNER_INFO db 0Dh, 0Ah, "Block is \$"

OWNER_FREE db "free\$"

OWNER_XMS db " OS XMS UMB\$"

OWNER_TM db " driver's top memory\$"

OWNER_DOS db " MS DOS\$"

OWNER_386CB db " busy by 386MAX UMB\$"
OWNER_386B db " blocked by 386MAX\$"

OWNER 386 db " 386MAX UMB\$"

; There is some CUSTOM procedures...

PRINT_STRING PROC near

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT_STRING ENDP

KBYTES_PRINT PROC near

push DX

mov DX, offset KBYTES

call PRINT_STRING

pop DX

ret

KBYTES_PRINT ENDP

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

```
int
                   21h;
                   DL, DH
            mov
                   DL, 0Ah
             cmp
             jl
                         PRINT_EXT
             add
                   DL, 07h
      PRINT_EXT:
                   DL, '0'
             add
             int
                   21h;
                   DX
             pop
                   ВХ
             pop
                   \mathsf{AX}
             pop
      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
            PROC
FREE_MEM
             push AX
             push
                   \mathsf{BX}
             push DX
                   BX, offset PROG_END
            mov
                   BX, 100h
             add
             shr
                   BX, 1
             shr
                   BX, 1
                   BX, 1
             shr
             shr
                   BX, 1 ; to paragraph
                   AH, 4Ah
            mov
                   21h
             int
                   DX
             pop
             pop
                   BX
             pop
                   \mathsf{AX}
      ret
FREE_MEM
            ENDP
```

```
ADD_MEM
             PROC
             push AX
                   \mathsf{BX}
             push
                   DX
             push
                   BX, 1000h
             mov
                   AH, 48h
             mov
             int
                   21h
                   \mathsf{DX}
             pop
                   ВХ
             pop
             pop
                   AX
      ret
ADD_MEM
              ENDP
    ; here is the action...
PRINT_AVL_MEMORY PROC NEAR
           push
                    \mathsf{AX}
           push
                    BX
                   DX
           push
                   SI
           push
           xor
                   AX, AX
             int
                   12h
                   DX, offset AVL_MEMORY_INFO
           mov
           call
                   PRINT_STRING
                          DX, DX
             xor
           call
                    DEC_WORD_PRINT
         call
                  KBYTES_PRINT
           pop
                    SI
                    DX
           pop
                    BX
           pop
           pop
                    \mathsf{AX}
      ret
PRINT_AVL_MEMORY ENDP
PRINT_EXT_MEMORY PROC NEAR
             push AX
             push BX
             push
                   DX
             push
                   SI
                   AL, 30h
             mov
```

```
70h, AL
            out
                       AL, 71h
            in
                  BL, AL
            mov
                  AL, 31h
            mov
                  70h, AL
            out
            in
                       AL, 71h
                  AH, AL
            mov
                 AL, BL
            mov
                  DX, offset EXT_MEMORY_INFO
        mov
            call PRINT_STRING
            xor
                        DX, DX
            call DEC_WORD_PRINT
        call
                KBYTES_PRINT
            pop
                  SI
            pop
                  DX
            pop
                  BX
            pop
                  AX
      ret
PRINT_EXT_MEMORY ENDP
PRINT_MCB
                  PROC
            push AX
            push BX
            push CX
            push DX
            push ES
            push SI
                  AH, 52h
            mov
                  21h
            int
                  AX, ES:[BX-2]
            mov
                  ES, AX
            mov
            xor
                  CX, CX
      NEXT_MCB:
            inc
                  \mathsf{CX}
            mov
                  DX, offset MCB_NUM_INFO
            push CX
            call PRINT_STRING
            mov
                  AX, CX
                  DX, DX
            xor
            call DEC_WORD_PRINT
      OWNER_START:
            mov
                 DX, offset OWNER_INFO
```

```
call PRINT_STRING
            AX, AX
      xor
      mov
            AL, ES:[0h]
      push AX
            AX, ES:[1h]
      mov
      cmp
            AX, 0h
      je
                  PRINT_FREE
            AX, 6h
      cmp
                  PRINT_XMS
      jе
            AX, 7h
      cmp
                  PRINT TM
      je
            AX, 8h
      cmp
      je
                  PRINT_DOS
            AX, 0FFFAh
      cmp
                  PRINT_386CB
      je
      cmp
            AX, 0FFFDh
      je
                  PRINT_386B
            AX, 0FFFEh
      cmp
      je
                  PRINT_386
      xor
            DX, DX
      call HEX_WORD_PRINT
      jmp
            AREA_SIZE_START
PRINT_FREE:
            DX, offset OWNER_FREE
      mov
            OWNER_END
      jmp
PRINT_XMS:
            DX, offset OWNER_XMS
      mov
            OWNER_END
      jmp
PRINT_TM:
            DX, offset OWNER_TM
      mov
            OWNER_END
      jmp
PRINT_DOS:
            DX, offset OWNER_DOS
      mov
            OWNER_END
      jmp
PRINT 386CB:
            DX, offset OWNER_386CB
      mov
            OWNER_END
      jmp
PRINT_386B:
      mov
            DX, offset OWNER_386B
      jmp
            OWNER_END
PRINT_386:
            DX, offset OWNER_386
      mov
OWNER_END:
      call PRINT_STRING
```

```
AREA_SIZE_START:
                    DX, offset AREA_SIZE_INFO
             mov
             call PRINT_STRING
             mov
                    AX, ES:[3h]
                    BX, 10h
             mov
             mul
                    BX
             call DEC_WORD_PRINT
                    CX, 8
             mov
                    SI, SI
             xor
                    DX, offset END_LINE
             mov
             call PRINT_STRING
      LAST_BYTES_START:
                      DL, ES:[SI + 8h]
             mov
                      AH, 02h
             mov
             int
                      21h
             inc
                      SI
             loop
                      LAST_BYTES_START
                      AX, ES:[3h]
             mov
                      BX, ES
             mov
                      BX, AX
             add
             inc
                      BX
                      ES, BX
             \text{mov}
                      \mathsf{AX}
             pop
                      \mathsf{CX}
             pop
             cmp
                      AL, 5Ah
                      ENDING
             je
                      DX, offset END_LINE
             mov
             call
                      PRINT_STRING
             jmp
                      NEXT_MCB
      ENDING:
                      SI
             pop
                      ES
             pop
                      DX
             pop
                      \mathsf{CX}
             pop
                      ВХ
             pop
                      \mathsf{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

;DATA

AVL_MEMORY_INFO db "Available memory: \$"

EXT_MEMORY_INFO db "Extended memory: \$"

MCB_NUM_INFO db "MCB number: \$"

AREA_SIZE_INFO db " Area size: \$"

END_LINE db 0Dh, 0Ah, "\$"

KBYTES db "kbytes", 0Dh, 0Ah, "\$"

OWNER_INFO db 0Dh, 0Ah, "Block is \$"

OWNER FREE db "free\$"

OWNER_XMS db " OS XMS UMB\$"

OWNER_TM db " driver's top memory\$"

OWNER DOS db " MS DOS\$"

OWNER_386CB db " busy by 386MAX UMB\$"
OWNER_386B db " blocked by 386MAX\$"

OWNER_386 db " 386MAX UMB\$"

; There is some CUSTOM procedures...

PRINT_STRING PROC near

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT_STRING ENDP

KBYTES_PRINT PROC near

push DX

mov DX, offset KBYTES

call PRINT_STRING

pop DX

ret

KBYTES_PRINT ENDP

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

```
int
                   21h;
                   DL, DH
            mov
                   DL, 0Ah
             cmp
             jl
                         PRINT_EXT
             add
                   DL, 07h
      PRINT_EXT:
                   DL, '0'
             add
             int
                   21h;
                   DX
             pop
                   ВХ
             pop
                   \mathsf{AX}
             pop
      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
            PROC
FREE_MEM
             push AX
             push
                   \mathsf{BX}
             push DX
            mov
                   BX, offset PROG_END
                   BX, 100h
             add
             shr
                   BX, 1
             shr
                   BX, 1
                   BX, 1
             shr
                   BX, 1 ; to paragraph
             shr
                   AH, 4Ah
            mov
                   21h
             int
                   DX
             pop
             pop
                   BX
             pop
                   \mathsf{AX}
      ret
FREE_MEM
            ENDP
```

```
ADD_MEM
             PROC
             push
                      \mathsf{AX}
                      ВХ
             push
                      DX
             push
                      BX, 1000h
             mov
                      AH, 48h
             \text{mov}
             int
                      21h
                      DX
             pop
                      BX
             pop
                      AX
             pop
      ret
ADD_MEM
              ENDP
    ; here is the action...
PRINT_AVL_MEMORY PROC NEAR
           push
                    \mathsf{AX}
           push
                    BX
                    DX
           push
           push
                    SI
           xor
                   AX, AX
             int
                   12h
                    DX, offset AVL_MEMORY_INFO
           mov
           call
                    PRINT_STRING
                          DX, DX
             xor
           call
                    DEC_WORD_PRINT
         call
                  KBYTES_PRINT
           pop
                    SI
                    DX
           pop
                    BX
           pop
           pop
                    \mathsf{AX}
      ret
PRINT_AVL_MEMORY ENDP
PRINT_EXT_MEMORY PROC NEAR
             push AX
             push
                   BX
             push
                   DX
             push
                   SI
                   AL, 30h
             mov
```

```
70h, AL
            out
                       AL, 71h
            in
                  BL, AL
            mov
                  AL, 31h
            mov
                  70h, AL
            out
            in
                       AL, 71h
                  AH, AL
            mov
                 AL, BL
            mov
                  DX, offset EXT_MEMORY_INFO
        mov
            call PRINT_STRING
            xor
                        DX, DX
            call DEC_WORD_PRINT
        call
                KBYTES_PRINT
            pop
                  SI
            pop
                  DX
            pop
                  BX
            pop
                  AX
      ret
PRINT_EXT_MEMORY ENDP
PRINT_MCB
                  PROC
            push AX
            push BX
            push CX
            push DX
            push ES
            push SI
                  AH, 52h
            mov
                  21h
            int
                  AX, ES:[BX-2]
            mov
                  ES, AX
            mov
            xor
                  CX, CX
      NEXT_MCB:
            inc
                  \mathsf{CX}
            mov
                  DX, offset MCB_NUM_INFO
            push CX
            call PRINT_STRING
            mov
                  AX, CX
                  DX, DX
            xor
            call DEC_WORD_PRINT
      OWNER_START:
            mov
                 DX, offset OWNER_INFO
```

```
call PRINT_STRING
            AX, AX
      xor
      mov
            AL, ES:[0h]
      push AX
            AX, ES:[1h]
      mov
            AX, 0h
      cmp
      je
                  PRINT_FREE
            AX, 6h
      cmp
      je
                  PRINT_XMS
            AX, 7h
      cmp
                  PRINT TM
      je
            AX, 8h
      cmp
      je
                  PRINT_DOS
            AX, 0FFFAh
      cmp
                  PRINT_386CB
      je
      cmp
            AX, 0FFFDh
      je
                  PRINT_386B
            AX, 0FFFEh
      cmp
      je
                  PRINT_386
      xor
            DX, DX
      call HEX_WORD_PRINT
      jmp
            AREA_SIZE_START
PRINT_FREE:
            DX, offset OWNER_FREE
      mov
            OWNER_END
      jmp
PRINT_XMS:
            DX, offset OWNER_XMS
      mov
            OWNER_END
      jmp
PRINT_TM:
            DX, offset OWNER_TM
      mov
            OWNER_END
      jmp
PRINT_DOS:
            DX, offset OWNER_DOS
      mov
            OWNER_END
      jmp
PRINT 386CB:
            DX, offset OWNER_386CB
      mov
            OWNER_END
      jmp
PRINT_386B:
      mov
            DX, offset OWNER_386B
      jmp
            OWNER_END
PRINT_386:
            DX, offset OWNER_386
      mov
OWNER_END:
      call PRINT_STRING
```

```
AREA_SIZE_START:
                    DX, offset AREA_SIZE_INFO
             mov
             call PRINT_STRING
             mov
                    AX, ES:[3h]
                    BX, 10h
             mov
             mul
                    BX
             call DEC_WORD_PRINT
                    CX, 8
             mov
                    SI, SI
             xor
                    DX, offset END_LINE
             mov
             call PRINT_STRING
      LAST_BYTES_START:
                      DL, ES:[SI + 8h]
             mov
                      AH, 02h
             mov
             int
                      21h
             inc
                      SI
             loop
                      LAST_BYTES_START
                      AX, ES:[3h]
             mov
                      BX, ES
             mov
                      BX, AX
             add
             inc
                      BX
                      ES, BX
             \text{mov}
                      \mathsf{AX}
             pop
                      \mathsf{CX}
             pop
             cmp
                      AL, 5Ah
                      ENDING
             je
                      DX, offset END_LINE
             mov
             call
                      PRINT_STRING
             jmp
                      NEXT_MCB
      ENDING:
                      SI
             pop
                      ES
             pop
                      DX
             pop
                      \mathsf{CX}
             pop
                      ВХ
             pop
                      \mathsf{AX}
             pop
      ret
PRINT_MCB
                    ENDP
BEGIN:
         call
                  ADD_MEM
         call
                    FREE_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