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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Операционные системы»

Тема: Исследование организации управления основной памятью

Студент гр. 8383 _____ Колмыков В.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

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

Процедуры, используемые в работе.

Название процедуры	Описание
AVAIBLE_MEMORY	Процедура вывода размера доступной памяти в байтах
WRITE	Процедура печати строки по смещению DX
WRITE_DEX_WORD	Вывод содержимого DX и AX в 10-ирично̆й с.с.
EXTENDED_MEMORY	Процедура вывода размера расширенной памяти
GET_MCB	Процедура вывода MCB
WRITE_HEX_WORD	Печать содержимого AX в 16-ричной с. с.
WRITE_HEX_BYTE	Печать содержимого AL в 16-ирично̆й с. с.
FREE_MEMORY	Функция освобождения памяти
ALLOCATE_MEMORY	Функция выделения памяти

Ход работы.

Был написан и отлажен COM модуль, который выбирает и распечатывает следующую информацию:

1) Кол-во доступной памяти
2) Размер расширенной памяти
3) Выводит цепочку MCB

Пример вывода программы продемонстрирован на рис. 1.

```
Avaible memory: 648912 bytes
Extended memory: 15360 kbytes

MCB number 1
Owner:  MS DOS
Area size: 16 bytes



MCB number 2
Owner:  free
Area size: 64 bytes
DPMILOAD


MCB number 3
Owner: 0040
Area size: 256 bytes



MCB number 4
Owner: 0192
Area size: 144 bytes



MCB number 5
Owner: 0192
Area size: 648912 bytes
LR3_1
```

Рисунок 1 – Пример работы первой версии программы

В программу было добавлено освобождение лишней памяти. Пример работы программы приведен на рис. 2. Видно, что освобожденная часть памяти теперь относится к дополнительному шестому блоку управления памятью.

```
Avaible memory: 648912 bytes
Memory was free successfuly
Extended memory: 15360 kbytes

MCB number 1
Owner:  MS DOS
Area size: 16 bytes


MCB number 2
Owner:  free
Area size: 64 bytes
DPMILOAD


MCB number 3
Owner: 0040
Area size: 256 bytes


MCB number 4
Owner: 0192
Area size: 144 bytes


MCB number 5
Owner: 0192
Area size: 1232 bytes
LR3_2


MCB number 6
Owner:  free
Area size: 647664 bytes
ry avail
```

Рисунок 2 – Пример работы второй версии программы

В программу было добавлено запрос выделения 64 кб памяти после освобождения. Пример работы программы приведен на рис. 3. Выделенная память относится к 6 блоку, освобожденная – к 7.

```
Avaible memory: 648912 bytes
Memory was free successfuly
Memory was allocate successfuly
Extended memory: 15360 kbytes

MCB number 1
Owner:  MS DOS
Area size: 16 bytes


MCB number 2
Owner:  free
Area size: 64 bytes
DPMILOAD


MCB number 3
Owner: 0040
Area size: 256 bytes


MCB number 4
Owner: 0192
Area size: 144 bytes


MCB number 5
Owner: 0192
Area size: 1328 bytes
LR3_3


MCB number 6
Owner: 0192
Area size: 65536 bytes
LR3_3


MCB number 7
Owner:  free
Area size: 582016 bytes
AЌ
```

Рисунок 3 – Пример работы третьей версии программы

Программа была изменена так, что запрос выделения памяти происходит до ее освобождения. Результат работы приведен на рис. 4. Память не была выделена.

```
Avaible memory: 648912 bytes
Error, memory wasn't allocate
Memory was free successfuly
Extended memory: 15360 kbytes

MCB number 1
Owner:  MS DOS
Area size: 16 bytes


MCB number 2
Owner:  free
Area size: 64 bytes
DPMILOAD


MCB number 3
Owner: 0040
Area size: 256 bytes


MCB number 4
Owner: 0192
Area size: 144 bytes


MCB number 5
Owner: 0192
Area size: 1328 bytes
LR3_4


MCB number 6
Owner:  free
Area size: 647568 bytes
ine not
```

Рисунок 4 – Пример работы некорректной версии программы

Ответы на контрольные вопросы.

1) Что означает «доступный объем» памяти?

   Размер памяти в системе, доступный для запуска и выполнения программ.

2) Где MCB блок вашей программы в списке?

   В первой, второй и четвертой версиях программы это пятый блок. В третьем случае еще и шестой (выделенный во время выполнения).

3) Какой размер памяти занимает программа в каждом случае?

   Версия 1: программа занимает всю доступную память (648 912 байт).

   Версия 2: 1232 байта (столько было запрошено оставить при освобождении).

Версия 3: 1328 байт (осталось при освобождении памяти) + 65536 байт (выделилось при выполнении)

Версия 4: 1328 байт (осталось при освобождении). Дополнительная память не выделилась.

Выводы.

В ходе выполнения лабораторной работы были исследованы структуры данных и работа функций управления памятью ядра ОС.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

```
LR3 SEGMENT
      ASSUME CS:LR3, DS:LR3, SS: NOTHING, ES:NOTHING
      org 100h
      START:
            jmp BEGIN
      ;данные
      STR_AVAIBLE_MEMORY db 13, 10, "Avaible memory: $"
      STR_BYTES db " bytes$"
      STR_EXTENDED_MEMORY db 13, 10, "Extended memory: $"
      STR_KBYTE db " kbytes$"
      STR_ENDL db 13, 10, "$"
      STR_MCB_NUM db 13, 10, "MCB number $"
      STR_OWNER db 13, 10, "Owner: $"
      STR_O1 db " free$";
      STR_O2 db " OS XMS UMB$";
      STR_O3 db " driver's top memory$";
      STR_O4 db " MS DOS$";
      STR_O5 db " control block 386MAX UMB$";
      STR_O6 db " blocked 386MAX$";
      STR_O7 db " 386MAX UMB$";
      STR_AREA_SIZE db 13, 10, "Area size: $"
      STR_ERROR_OF_FREEDOM db 13, 10, "Error, memory wasn't free$"
      STR_SUCCES_OF_FREEDOM db 13, 10, "Memory was free successfuly$"
      STR_ERROR_OF_ALLOCATE db 13, 10, "Error, memory wasn't allocate$"
      STR_SUCCES_OF_ALLOCATE db 13, 10, "Memory was allocate
successfuly$"
      ;_____________________________________________
      ;код
      WRITE PROC
            push AX
            mov AH, 9h
            int 21h
            pop AX
            ret
      WRITE ENDP
      ;_____________________________________________
      WRITE_DEC_WORD PROC
            push AX
            push CX
            push DX
            push BX

            mov BX, 10
            xor CX, CX
      GETTING_NUMS:
            div BX
            push DX
            xor DX, DX
            inc CX
            cmp AX, 0h
            jnz GETTING_NUMS


      WRITING:
            pop DX
            or DL, 30h
            mov AH, 02h
```

```
      int 21h
      loop WRITING

      pop BX
      pop DX
      pop CX
      pop AX
      ret
WRITE_DEC_WORD ENDP
;_____________________________________________
AVAIBLE_MEMORY PROC
      push AX
      push BX
      push DX

      mov DX, offset STR_AVAIBLE_MEMORY
      call WRITE
      mov AH, 4Ah
      mov BX, 0FFFFh
      int 21h
      mov AX, BX
      mov BX, 10h
      mul BX
      call WRITE_DEC_WORD
      mov DX, offset STR_BYTES
      call WRITE

      pop DX
      pop BX
      pop AX
      ret
AVAIBLE_MEMORY ENDP
;_____________________________________________
EXTENDED_MEMORY PROC
      push AX
      push BX
      push DX

      mov DX, offset STR_EXTENDED_MEMORY
      call WRITE
      mov AL, 30h
      out 70h, AL
      in AL, 71h
      mov BL, AL
      mov AL, 31h
      out 70h, AL
      in AL, 71h
      mov BH, AL
      mov AX, BX
      xor DX, DX
      call WRITE_DEC_WORD
      mov DX, offset STR_KBYTE
      call WRITE

      pop DX
      pop BX
      pop AX
      ret
EXTENDED_MEMORY ENDP
;_____________________________________________
GET_MCB PROC
      push AX
      push BX
```

```
      push CX
      push DX
      push ES
      push SI

      mov AH, 52h
      int 21h
      mov AX, ES:[BX-2]
      mov ES, AX
      xor CX, CX
ANOTHER_MCB:
      inc CX
      mov DX, offset STR_ENDL
      call WRITE
      mov DX, offset STR_MCB_NUM
      push CX
      call WRITE
      mov AX, CX
      xor DX, DX
      call WRITE_DEC_WORD
      mov DX, offset STR_OWNER
      call WRITE
      xor AX, AX
      mov AL, ES:[0h]
      push AX
      mov AX, ES:[1h]

      cmp AX, 0h
      je AREA_FREE
      cmp AX, 6h
      je AREA_DRIVER
      cmp AX, 7h
      je AREA_TOP
      cmp AX, 8h
      je AREA_DOS
      cmp AX, 0FFFAh
      je AREA_BLOCK
      cmp AX, 0FFFDh
      je AREA_BLOCKED
      cmp AX, 0FFFEh
      je AREA_LAST
      xor DX, DX
      call WRITE_HEX_WORD
      jmp AFTER_SWITCH


AREA_FREE:
      mov DX, offset STR_O1
      jmp END_OF_SWITCH
AREA_DRIVER:
      mov DX, offset STR_O2
      jmp END_OF_SWITCH
AREA_TOP:
      mov DX, offset STR_O3
      jmp END_OF_SWITCH
AREA_DOS:
      mov DX, offset STR_O4
      jmp END_OF_SWITCH
AREA_BLOCK:
      mov DX, offset STR_O5
      jmp END_OF_SWITCH
AREA_BLOCKED:
      mov DX, offset STR_O6
```

```
      jmp END_OF_SWITCH
AREA_LAST:
      mov DX, offset STR_07
END_OF_SWITCH:
      call WRITE

AFTER_SWITCH:
      mov DX, offset STR_AREA_SIZE
      call WRITE
      mov AX, ES:[3h]
      mov BX, 10h
      mul BX
      call WRITE_DEC_WORD
      mov DX, offset STR_BYTES
      call WRITE
      mov CX, 8
      xor SI, SI
      mov DX, offset STR_ENDL
      call WRITE
PRINT_LAST_BYTES:
      mov DL, ES:[SI + 8h]
      mov AH, 02h
      int 21h
      inc SI
      loop PRINT_LAST_BYTES

      mov AX, ES:[3h]
      mov BX, ES
      add BX, AX
      inc BX
      mov ES, BX
      pop AX
      pop CX
      cmp AL, 5Ah
      je END_PROC
      mov DX, offset STR_ENDL
      call WRITE
      jmp ANOTHER_MCB

END_PROC:
      pop SI
      pop ES
      pop DX
      pop CX
      pop BX
      pop AX
      ret
GET_MCB ENDP
;_____________________________________________
WRITE_HEX_WORD PROC
      push AX

      push AX
      mov AL, AH
      call WRITE_HEX_BYTE
      pop AX
      call WRITE_HEX_BYTE

      pop AX
      ret
WRITE_HEX_WORD ENDP
;_____________________________________________
WRITE_HEX_BYTE PROC
```

```
            push AX
            push BX
            push DX

            mov AH, 0
            mov BL, 16
            div BL
            mov DX, AX
            mov AH, 02h
            cmp DL, 0Ah
            jl PRINT
            add DL, 7
      PRINT:
            add DL, '0'
            int 21h;

            mov DL, DH
            cmp DL, 0Ah
            jl PRINT2
            add DL, 7
      PRINT2:
            add DL, '0'
            int 21h;

            pop DX
            pop BX
            pop AX
            ret
      WRITE_HEX_BYTE ENDP
;_____________________________________________________
      FREE_MEMORY PROC
            push AX
            push BX
            push DX

            mov BX, offset STACK_END
            add BX, 10Fh
            shr BX, 4
            mov AH, 4Ah
            int 21h
            jnc SUCCES
            mov DX, offset STR_ERROR_OF_FREEDOM
            call WRITE
            jmp RETURN

      SUCCES:
            mov DX, offset STR_SUCCES_OF_FREEDOM
            call WRITE

      RETURN:
            pop DX
            pop BX
            pop AX
            ret
      FREE_MEMORY ENDP
;_____________________________________________
      ALLOCATE_MEMORY PROC
            push AX
            push BX
            push DX

            mov BX, 1000h
            mov AH, 48h
```

```
            int 21h
            jnc ALLOCATE_SUCCES
            mov DX, offset STR_ERROR_OF_ALLOCATE
            call WRITE
            jmp ALLOCATE_RETURN

      ALLOCATE_SUCCES:
            mov DX, offset STR_SUCCES_OF_ALLOCATE
            call WRITE

      ALLOCATE_RETURN:
            pop DX
            pop BX
            pop AX
            ret
      ALLOCATE_MEMORY ENDP
;_____________________________________________
      BEGIN:
            call AVAIBLE_MEMORY
            call FREE_MEMORY
            call ALLOCATE_MEMORY
            call EXTENDED_MEMORY
            call GET_MCB

            ;TO DOS
            xor AL, AL
            mov AH, 4Ch
            int 21h

      STACK_FOR_FREEDOM:
            DW 128 dup(0)
      STACK_END:
LR3 ENDS
END START
```