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(ка) гр. 9382 Павлов Р.В.

Преподаватель Ефремов М.А.

Санкт-Петербург

2021

Постановка задачи.

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

В лабораторной работе исследуются структуры данных и работа функций управления памятью ядра операционной системы.

Функции и структуры данных:

main	Вызов реализованных процедур, вывод информации на экран
print_string	Вывод указанной строки
clr_mem	Очистка памяти
byte_to_dec	Перевод байтового значения в десятичное число
byte_to_hex	Перевод байтового значения в шестнадцатеричное число
word_to_hex	Перевод слова в шестнадцатеричное число
mem_to_dec	Запись объёма занимаемой памяти в строку
print_mem_size	Вывод объёма занимаемой памяти
print_mcb_info	Вывод информации о блоке управления памятью

print_mcbs	Вывод последовательно всех блоков памяти
av_mem	Доступная память
ext_mem	Расширенная память
MCB_num	Номер блока MCB
MCB_addr	Адрес блока MCB
MCB_size	Размер блока MCB
PSP_addr	Адрес PSP
SC_SD	Информация об SC/SD

Последовательность действий:

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

1) Количество доступной памяти.

2) Размер расширенной памяти.

3) Выводит цепочку блоков управления памятью.

Адреса при выводе представляются шестнадцатеричными числами. Объем памяти функциями управления памятью выводится в параграфах. Необходимо преобразовать его в байты и выводить в виде десятичных чисел. Последние восемь байт MCB выводятся как символы, не следует преобразовывать их в шестнадцатеричные числа.

Запустите программу и внимательно оцените результаты. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 2. Измените программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah

прерывания 21h (пример в разделе «Использование функции 4AH»). Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущем шаге. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 3. Измените программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48H прерывания 21H. Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущих шагах. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 4. Измените первоначальный вариант программы, запросив 64Кб памяти функцией 48H прерывания 21H до освобождения памяти. Обязательно обрабатывайте завершение функций ядра, проверяя флаг CF. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 5. Оцените результаты, полученные на предыдущих шагах. Ответьте на контрольные вопросы и оформите отчет.

Необходимые сведения для составления программы

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

MCB имеет следующую структуру:

Смещение	Длина поля (байт)	Содержимое поля
00h	1	тип MCB: 5Ah, если последний в списке, 4Dh, если не последний
01h	2	Сегментный адрес PSP владельца участка памяти, либо 0000h - свободный участок, 0006h - участок принадлежит драйверу OS XMS UMB 0007h - участок является исключенной верхней памятью драйверов 0008h - участок принадлежит MS DOS FFFAh - участок занят управляющим блоком 386MAX UMB FFFDh - участок заблокирован 386MAX FFFEh - участок принадлежит 386MAX UMB
03h	2	Размер участка в параграфах
05h	3	Зарезервирован
08h	8	"SC" - если участок принадлежит MS DOS, то в нем системный код "SD" - если участок принадлежит MS DOS, то в нем системные данные

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

Адрес первого MCB хранится во внутренней структуре MS DOS, называемой "List of Lists" (список списков). Доступ к указателю на эгу структуру можно получить используя функцию 52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX—2] и есть адрес самого первого MCB.

Размер расширенной памяти находится в ячейках 3011, 3111 CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта. Размер расширенной памяти в Кбайтах можно определить обращаясь к ячейкам CMOS следующим образом:

mov AL,30h ; запись адреса ячейки CMOS out 70h,AL

in AL,7lh ; чтение младшего байта

mov BL,AL ; размера расширенной памяти mov AL,3lh ; запись адреса ячейки CMOS out 70h,A

in AL,7lh ; чтение старшего байта

; размера расширенной памяти

Выполнение работы.

1. Создан загрузочный модуль типа .COM, выводящий на экран информацию об объёме доступной памяти, расширенной памяти, а

также данные MCB (Memory Control Blocks).

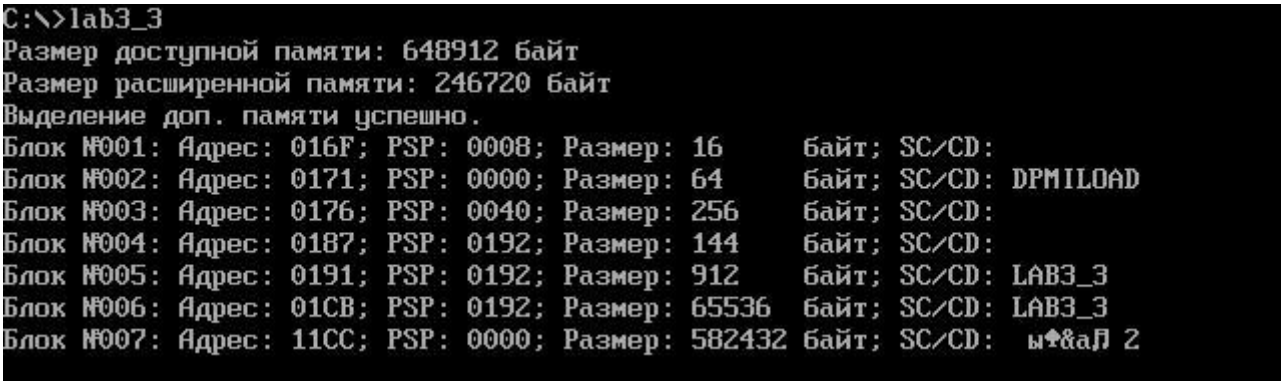
```
C:\>lab3_1
Размер доступной памяти: 648912 байт
Размер расширенной памяти: 246720 байт
Блок №001: Адрес: 016F; PSP: 0008; Размер: 16     байт; SC/CD:
Блок №002: Адрес: 0171; PSP: 0000; Размер: 64     байт; SC/CD: DPMILOAD
Блок №003: Адрес: 0176; PSP: 0040; Размер: 256    байт; SC/CD:
Блок №004: Адрес: 0187; PSP: 0192; Размер: 144    байт; SC/CD:
Блок №005: Адрес: 0191; PSP: 0192; Размер: 648912 байт; SC/CD: LAB3_1
```

2. Добавлено освобождение занимаемой программой памяти, получен следующий результат:

```
C:\>lab3_2
Размер доступной памяти: 648912 байт
Размер расширенной памяти: 246720 байт
Блок №001: Адрес: 016F; PSP: 0008; Размер: 16     байт; SC/CD:
Блок №002: Адрес: 0171; PSP: 0000; Размер: 64     байт; SC/CD: DPMILOAD
Блок №003: Адрес: 0176; PSP: 0040; Размер: 256    байт; SC/CD:
Блок №004: Адрес: 0187; PSP: 0192; Размер: 144    байт; SC/CD:
Блок №005: Адрес: 0191; PSP: 0192; Размер: 832    байт; SC/CD: LAB3_2
Блок №006: Адрес: 01C6; PSP: 0000; Размер: 648064 байт; SC/CD:
```

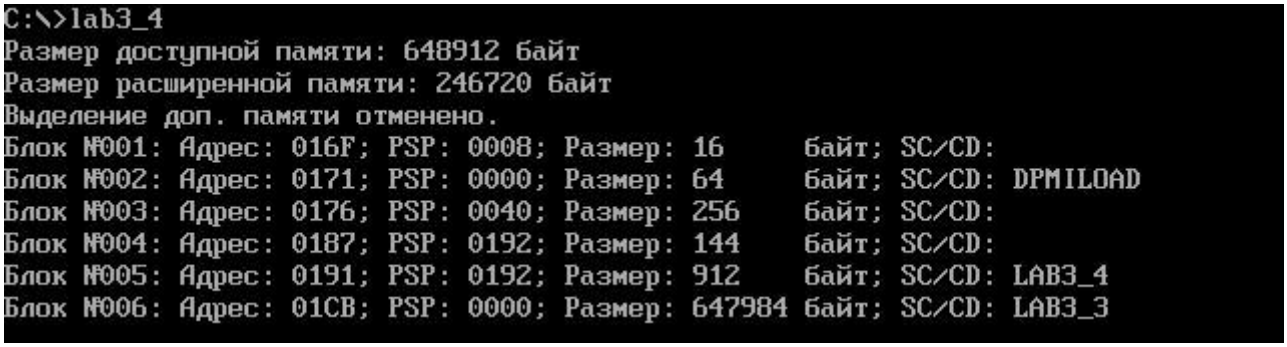
Как видно, программе была оставлена только необходимая память, а остальная была освобождена.

3. Добавлен запрос на выделение 64Кб памяти после освобождения памяти, получен следующий результат:



Теперь программе выделено также 65536 байт памяти из имеющихся 648064 свободных.

4. Запрос на выделение памяти помещён перед вызовом процедуры освобождения памяти. Теперь в выделении памяти отказано, поскольку свободной памяти не имеется:



Ответы на контрольные вопросы

1) Область основной памяти, которая выделяется программе и доступна ей во время выполнения.

2) Основной блок MCB во всех случаях – это блок №5, но на третьем скриншоте видно, что программе выделено два блока – это 5 и 6 блоки.

3) В первом случае – 648912 байт. Во втором – 832 байта. В третьем – 912 байт и 65536 байт дополнительно. В четвёртом – 912 байт.

Выводы.

Было исследовано и реализовано управление основной памятью ОС, изучена структура МСВ и организация распределения памяти.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Имя файла: lab3_1.asm

```
.model tiny

.code
      org 100h

      main proc far
            xor ax, ax
            push ds
            push ax

            call print_mem_size
            call print_mcbs

            mov ah, 4Ch
            int 21h

            retf
      main endp

      clr_mem proc near
      reset:
            mov byte ptr [si], ' '
            inc si
            loop reset
      retn
      clr_mem endp

      print_string proc near
            push ax
            mov ah, 9
            int 21h
            pop ax
            retn
      print_string endp

      byte_to_dec proc near
            push ax
            push bx
            push cx

            xor ah, ah
            mov bx, 10
            add si, 2

            mov cx, 3
            c1:
                  div bl
                  add ah, '0'
                  mov [si], ah
                  dec si
                  xor ah,ah
                  loop c1

            pop cx
            pop bx
```

```
      pop ax
      retn
byte_to_dec endp

byte_to_hex proc near
      push ax
      push bx
      push cx

      xor ah, ah
      mov bx, 16
      add si, 1

      mov cx, 2
      c2:
            div bl
            cmp ah, 10
            jl digit1
            add ah, 7
            digit1:
                  add ah, '0'
            mov [si], ah
            dec si
            xor ah, ah
            loop c2

      pop cx
      pop bx
      pop ax
      retn
byte_to_hex endp

word_to_hex proc near
      push ax
      push bx
      push cx
      push dx

      xor dx, dx
      mov bx, 16
      add si, 3

      mov cx, 4
      c3:
            div bx
            cmp dl, 10
            jl digit2
            add dl, 7
            digit2:
                  add dl, '0'
            mov [si], dl
            dec si
            xor dx, dx
            loop c3

      pop dx
      pop cx
      pop bx
      pop ax
      retn
word_to_hex endp

mem_to_dec proc near
      push ax
```

```
push bx
push cx
push dx
push si

  mov bx, 10h
  mul bx
  mov bx, 0ah
  xor cx, cx

  remnants:
  div bx
  push dx
  inc cx
  xor dx, dx
  cmp ax, 0h
  jnz remnants

  xtoc:
  pop dx
  or dl, 30h
  mov [si], dl
  inc si
  loop xtoc

pop si
pop dx
pop cx
pop bx
pop ax
  retn
  mem_to_dec endp

  print_mem_size proc near
mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx

mov si, offset av_mem + 25
call mem_to_dec
mov dx, offset av_mem
call print_string

mov al, 30h
out 70h, al
in al, 71h
mov al, 31h
out 70h, al
in al, 71h
mov ah, al

mov si, offset ext_mem + 27
call mem_to_dec
mov dx, offset ext_mem
call print_string

retn
  print_mem_size endp

  print_mcb_info proc near
push ax
push dx
push si
```

```
push di
push cx

mov ax, es
mov si, offset MCB_addr + 7
call word_to_hex
mov dx, offset MCB_addr
call print_string

mov ax, es:[1]
mov si, offset PSP_addr + 5
call word_to_hex
mov dx, offset PSP_addr
call print_string

mov ax, es:[3]
mov si, offset MCB_size + 8
call mem_to_dec
mov dx, offset MCB_size
call print_string

mov dx, offset SC_SD
call print_string

  add dx, 8
  push dx
  mov si, 8
  mov cx, 8
  scsd_out:
        mov dl, es:[si]
        mov ah, 02h
        int 21h
        inc si
        loop scsd_out

  pop dx
  call print_string

pop cx
pop di
pop si
pop dx
pop ax

retn
  print_mcb_info endp

  print_mcbs proc near

push es

mov ah, 52h
int 21h
mov es, es:[bx-2]
mov cl, 1

  iter_mcb:
        mov si, offset MCB_num + 6
        mov al, cl
        call byte_to_dec
        mov dx, offset MCB_num
        call print_string

        call print_mcb_info
```

```
                mov al, es:[0]
                cmp al, 5ah
                je exit

                mov bx, es:[3]
                mov ax, es
                add ax, bx
                inc ax
                mov es, ax

                inc cl

                push cx
                mov si, offset MCB_num + 6
                mov cx, 3
                call clr_mem

                mov si, offset MCB_size + 8
                mov cx, 6
                call clr_mem
                pop cx

                jmp iter_mcb

          exit:
                pop es

          retn
          print_mcbs endp


          av_mem db 'Размер доступной памяти: ', 6 dup(?), ' байт', 13, 10, '$'
          ext_mem db 'Размер расширенной памяти: ', 6 dup(?), ' байт', 13,
          10,
'$'
                                    '
          MCB_num db 'Блок №   : ,  '$'
          MCB_add  db 'Адрес:      dup(?),
          r        ',             4 ';         $'
          PSP_add  db 'PSP:       dup(?), ';
          r        ',           4 $'
          MCB_siz  db           ',   dup(?),   байт', ';
          e        'Размер:     6    '         $'
          SC_SD    'SC/CD:      13
          db       $',          ,   10, '$'


      end main
```

Имя файла: lab3_2.asm

```
.model tiny

.code
      org 100h

      main proc far
            xor ax, ax
            push ds
            push ax
```

```
call print_mem_size
call free
call print_mcbs

mov ah, 4Ch
```

```
        int 21h

        retf
  main endp

  free proc near
mov ax, offset edge
mov bx, 10h
xor dx, dx
div bx
inc ax
mov bx, ax
mov ah, 4ah
int 21h
retn
  free endp

  clr_mem proc near
  reset:
        mov byte ptr [si], ' '
        inc si
        loop reset
  retn
  clr_mem endp

  print_string proc near
        push ax
        mov ah, 9
        int 21h
        pop ax
        retn
  print_string endp

  byte_to_dec proc near
        push ax
        push bx
        push cx

        xor ah, ah
        mov bx, 10
        add si, 2

        mov cx, 3
        c1:
              div bl
              add ah, '0'
              mov [si], ah
              dec si
              xor ah,ah
              loop c1

        pop cx
        pop bx
        pop ax
        retn
  byte_to_dec endp

  byte_to_hex proc near
        push ax
        push bx
        push cx

        xor ah, ah
        mov bx, 16
```

```
        add si, 1

        mov cx, 2
        c2:
              div bl
              cmp ah, 10
              jl digit1
              add ah, 7
              digit1:
                    add ah, '0'
              mov [si], ah
              dec si
              xor ah, ah
              loop c2

        pop cx
        pop bx
        pop ax
        retn
  byte_to_hex endp

  word_to_hex proc near
        push ax
        push bx
        push cx
        push dx

        xor dx, dx
        mov bx, 16
        add si, 3

        mov cx, 4
        c3:
              div bx
              cmp dl, 10
              jl digit2
              add dl, 7
              digit2:
                    add dl, '0'
              mov [si], dl
              dec si
              xor dx, dx
              loop c3

        pop dx
        pop cx
        pop bx
        pop ax
        retn
  word_to_hex endp

  mem_to_dec proc near
        push ax
push bx
push cx
push dx
push si

  mov bx, 10h
  mul bx
  mov bx, 0ah
  xor cx, cx

  remnants:
```

```
  div bx
  push dx
  inc cx
  xor dx, dx
  cmp ax, 0h
  jnz remnants

  xtoc:
  pop dx
  or dl, 30h
  mov [si], dl
  inc si
  loop xtoc

pop si
pop dx
pop cx
pop bx
pop ax
  retn
  mem_to_dec endp

  print_mem_size proc near
mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx

mov si, offset av_mem + 25
call mem_to_dec
mov dx, offset av_mem
call print_string

mov al, 30h
out 70h, al
in al, 71h
mov al, 31h
out 70h, al
in al, 71h
mov ah, al

mov si, offset ext_mem + 27
call mem_to_dec
mov dx, offset ext_mem
call print_string

retn
  print_mem_size endp

  print_mcb_info proc near
push ax
push dx
push si
push di
push cx

mov ax, es
mov si, offset MCB_addr + 7
call word_to_hex
mov dx, offset MCB_addr
call print_string

mov ax, es:[1]
mov si, offset PSP_addr + 5
```

```
call word_to_hex
mov dx, offset PSP_addr
call print_string

mov ax, es:[3]
mov si, offset MCB_size + 8
call mem_to_dec
mov dx, offset MCB_size
call print_string

mov dx, offset SC_SD
call print_string

  add dx, 8
  push dx
  mov si, 8
  mov cx, 8
  scsd_out:
        mov dl, es:[si]
        mov ah, 02h
        int 21h
        inc si
        loop scsd_out

  pop dx
  call print_string

pop cx
pop di
pop si
pop dx
pop ax

retn
  print_mcb_info endp

  print_mcbs proc near

push es

mov ah, 52h
int 21h
mov es, es:[bx-2]
mov cl, 1

  iter_mcb:
        mov si, offset MCB_num + 6
        mov al, cl
        call byte_to_dec
        mov dx, offset MCB_num
        call print_string

        call print_mcb_info

        mov al, es:[0]
        cmp al, 5ah
        je exit

        mov bx, es:[3]
        mov ax, es
        add ax, bx
        inc ax
        mov es, ax
```

```
            inc cl

            push cx
            mov si, offset MCB_num + 6
            mov cx, 3
            call clr_mem

            mov si, offset MCB_size + 8
            mov cx, 6
            call clr_mem
            pop cx

            jmp iter_mcb

      exit:
            pop es

      retn
      print_mcbs endp


      av_mem db 'Размер доступной памяти: ', 6 dup(?), ' байт', 13, 10, '$'
      ext_mem db 'Размер расширенной памяти: ', 6 dup(?), ' байт', 13,
      10,
'$'

                                    '
      MCB_num db 'Блок №   : ,  '$'
      MCB_add  db 'Адрес:      dup(?),
      r        ',             4 ';         $'
      PSP_add  db 'PSP:       dup(?), ';
      r        ',           4 $'
      MCB_siz  db           ',   dup(?),   байт', ';
      e        'Размер:     6    '         $'
      SC_SD    'SC/CD:      13
      db       $',          ,   10, '$'

      edge:
end main


Имя файла: lab3_3.asm

.model tiny

.code
      org 100h

      main proc far
            xor ax, ax
            push ds
            push ax

            call print_mem_size
            call free
            call extend
            call print_mcbs
            call free

            mov ah, 4Ch
            int 21h

            retf
      main endp
```

```
extend proc near
mov bx, 1000h
mov ah, 48h
int 21h
```

17

```
  mov dx, offset ext_report
  call print_string

  jc deny

  add dx, 23
  call print_string
  jmp exit1

  deny:
     add dx, 34
     call print_string
     jmp exit1

  exit1:
     retn
  extend endp

  free proc near
mov ax, offset edge
mov bx, 10h
xor dx, dx
div bx
inc ax
mov bx, ax
mov ah, 4ah
int 21h
retn
  free endp

  clr_mem proc near
  reset:
        mov byte ptr [si], ' '
        inc si
        loop reset
  retn
  clr_mem endp

  print_string proc near
        push ax
        mov ah, 9
        int 21h
        pop ax
        retn
  print_string endp

  byte_to_dec proc near
        push ax
        push bx
        push cx

        xor ah, ah
        mov bx, 10
        add si, 2

        mov cx, 3
        c1:
              div bl
              add ah, '0'
              mov [si], ah
              dec si
              xor ah,ah
              loop c1
```

```
      pop cx
      pop bx
      pop ax
      retn
byte_to_dec endp

byte_to_hex proc near
      push ax
      push bx
      push cx

      xor ah, ah
      mov bx, 16
      add si, 1

      mov cx, 2
      c2:
            div bl
            cmp ah, 10
            jl digit1
            add ah, 7
            digit1:
                  add ah, '0'
            mov [si], ah
            dec si
            xor ah, ah
            loop c2

      pop cx
      pop bx
      pop ax
      retn
byte_to_hex endp

word_to_hex proc near
      push ax
      push bx
      push cx
      push dx

      xor dx, dx
      mov bx, 16
      add si, 3

      mov cx, 4
      c3:
            div bx
            cmp dl, 10
            jl digit2
            add dl, 7
            digit2:
                  add dl, '0'
            mov [si], dl
            dec si
            xor dx, dx
            loop c3

      pop dx
      pop cx
      pop bx
      pop ax
      retn
word_to_hex endp
```

```
  mem_to_dec proc near
        push ax
push bx
push cx
push dx
push si

  mov bx, 10h
  mul bx
  mov bx, 0ah
  xor cx, cx

  remnants:
  div bx
  push dx
  inc cx
  xor dx, dx
  cmp ax, 0h
  jnz remnants

  xtoc:
  pop dx
  or dl, 30h
  mov [si], dl
  inc si
  loop xtoc

pop si
pop dx
pop cx
pop bx
pop ax
  retn
  mem_to_dec endp

  print_mem_size proc near
mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx

mov si, offset av_mem + 25
call mem_to_dec
mov dx, offset av_mem
call print_string

mov al, 30h
out 70h, al
in al, 71h
mov al, 31h
out 70h, al
in al, 71h
mov ah, al

mov si, offset ext_mem + 27
call mem_to_dec
mov dx, offset ext_mem
call print_string

retn
  print_mem_size endp

  print_mcb_info proc near
```

```
push ax
push dx
push si
push di
push cx

mov ax, es
mov si, offset MCB_addr + 7
call word_to_hex
mov dx, offset MCB_addr
call print_string

mov ax, es:[1]
mov si, offset PSP_addr + 5
call word_to_hex
mov dx, offset PSP_addr
call print_string

mov ax, es:[3]
mov si, offset MCB_size + 8
call mem_to_dec
mov dx, offset MCB_size
call print_string

mov dx, offset SC_SD
call print_string

  add dx, 8
  push dx
  mov si, 8
  mov cx, 8
  scsd_out:
        mov dl, es:[si]
        mov ah, 02h
        int 21h
        inc si
        loop scsd_out

  pop dx
  call print_string

pop cx
pop di
pop si
pop dx
pop ax

retn
  print_mcb_info endp

  print_mcbs proc near

push es

mov ah, 52h
int 21h
mov es, es:[bx-2]
mov cl, 1

  iter_mcb:
        mov si, offset MCB_num + 6
        mov al, cl
        call byte_to_dec
        mov dx, offset MCB_num
```

```
                call print_string

                call print_mcb_info

                mov al, es:[0]
                cmp al, 5ah
                je exit

                mov bx, es:[3]
                mov ax, es
                add ax, bx
                inc ax
                mov es, ax

                inc cl

                push cx
                mov si, offset MCB_num + 6
                mov cx, 3
                call clr_mem

                mov si, offset MCB_size + 8
                mov cx, 6
                call clr_mem
                pop cx

                jmp iter_mcb

          exit:
                pop es

         retn
          print_mcbs endp


         av_mem db 'Размер доступной памяти: ', 6 dup(?), ' байт', 13, 10, '$'
          ext_mem db 'Размер расширенной памяти: ', 6 dup(?), ' байт', 13,
          10,
'$'

                                  '
          MCB_num db 'Блок №   : ,  '$'
          MCB_add  db 'Адрес:      dup(?),
          r        ',             4 ';         $'
          PSP_add  db 'PSP:       dup(?), ';
          r        ',           4 $'
          MCB_siz  db           ',   dup(?),   байт', ';
          e        'Размер:     6    '         $'
          SC_SD    'SC/CD:      13
          db       $',          ,   10, '$'
        ext_report db 'Выделение доп. памяти $успешно.', 13, 10, '$отменено.',
13, 10, '$'

          edge:
      end main
```

Имя файла: lab3_4.asm

```
.model tiny

.code
      org 100h
```

```
main proc far
      xor ax, ax
      push ds
      push ax
```

```
        call print_mem_size
        call extend
        call free
        call print_mcbs

        mov ah, 4Ch
        int 21h

        retf
  main endp

  extend proc near
  mov bx, 1000h
  mov ah, 48h
  int 21h

  mov dx, offset ext_report
  call print_string

  jc deny

  add dx, 23
  call print_string
  jmp exit1

  deny:
     add dx, 34
     call print_string
     jmp exit1

  exit1:
     retn
  extend endp

  free proc near
mov ax, offset edge
mov bx, 10h
xor dx, dx
div bx
inc ax
mov bx, ax
mov ah, 4ah
int 21h
retn
  free endp

  clr_mem proc near
  reset:
        mov byte ptr [si], ' '
        inc si
        loop reset
  retn
  clr_mem endp

  print_string proc near
        push ax
        mov ah, 9
        int 21h
        pop ax
        retn
  print_string endp

  byte_to_dec proc near
        push ax
```

```
      push bx
      push cx

      xor ah, ah
      mov bx, 10
      add si, 2

      mov cx, 3
      c1:
            div bl
            add ah, '0'
            mov [si], ah
            dec si
            xor ah,ah
            loop c1

      pop cx
      pop bx
      pop ax
      retn
byte_to_dec endp

byte_to_hex proc near
      push ax
      push bx
      push cx

      xor ah, ah
      mov bx, 16
      add si, 1

      mov cx, 2
      c2:
            div bl
            cmp ah, 10
            jl digit1
            add ah, 7
            digit1:
                  add ah, '0'
            mov [si], ah
            dec si
            xor ah, ah
            loop c2

      pop cx
      pop bx
      pop ax
      retn
byte_to_hex endp

word_to_hex proc near
      push ax
      push bx
      push cx
      push dx

      xor dx, dx
      mov bx, 16
      add si, 3

      mov cx, 4
      c3:
            div bx
            cmp dl, 10
```

```
                jl digit2
                add dl, 7
                digit2:
                      add dl, '0'
                mov [si], dl
                dec si
                xor dx, dx
                loop c3

          pop dx
          pop cx
          pop bx
          pop ax
          retn
    word_to_hex endp

    mem_to_dec proc near
          push ax
push bx
push cx
push dx
push si

    mov bx, 10h
    mul bx
    mov bx, 0ah
    xor cx, cx

    remnants:
    div bx
    push dx
    inc cx
    xor dx, dx
    cmp ax, 0h
    jnz remnants

    xtoc:
    pop dx
    or dl, 30h
    mov [si], dl
    inc si
    loop xtoc

pop si
pop dx
pop cx
pop bx
pop ax
    retn
    mem_to_dec endp

    print_mem_size proc near
mov ah, 4ah
mov bx, 0ffffh
int 21h
mov ax, bx

mov si, offset av_mem + 25
call mem_to_dec
mov dx, offset av_mem
call print_string

mov al, 30h
out 70h, al
```

```
in al, 71h
mov al, 31h
out 70h, al
in al, 71h
mov ah, al

mov si, offset ext_mem + 27
call mem_to_dec
mov dx, offset ext_mem
call print_string

retn
  print_mem_size endp

  print_mcb_info proc near
push ax
push dx
push si
push di
push cx

mov ax, es
mov si, offset MCB_addr + 7
call word_to_hex
mov dx, offset MCB_addr
call print_string

mov ax, es:[1]
mov si, offset PSP_addr + 5
call word_to_hex
mov dx, offset PSP_addr
call print_string

mov ax, es:[3]
mov si, offset MCB_size + 8
call mem_to_dec
mov dx, offset MCB_size
call print_string

mov dx, offset SC_SD
call print_string

  add dx, 8
  push dx
  mov si, 8
  mov cx, 8
  scsd_out:
        mov dl, es:[si]
        mov ah, 02h
        int 21h
        inc si
        loop scsd_out

  pop dx
  call print_string

pop cx
pop di
pop si
pop dx
pop ax

retn
  print_mcb_info endp
```

```
  print_mcbs proc near

push es

mov ah, 52h
int 21h
mov es, es:[bx-2]
mov cl, 1

  iter_mcb:
        mov si, offset MCB_num + 6
        mov al, cl
        call byte_to_dec
        mov dx, offset MCB_num
        call print_string

        call print_mcb_info

        mov al, es:[0]
        cmp al, 5ah
        je exit

        mov bx, es:[3]
        mov ax, es
        add ax, bx
        inc ax
        mov es, ax

        inc cl

        push cx
        mov si, offset MCB_num + 6
        mov cx, 3
        call clr_mem

        mov si, offset MCB_size + 8
        mov cx, 6
        call clr_mem
        pop cx

        jmp iter_mcb

  exit:
        pop es

retn
  print_mcbs endp


 av_mem db 'Размер доступной памяти: ', 6 dup(?), ' байт', 13, 10, '$'
  ext_mem db 'Размер расширенной памяти: ', 6 dup(?), ' байт', 13,
  10,
'$'

                          '
  MCB_num db 'Блок №   : ,  '$'
  MCB_add  db 'Адрес:      dup(?),
  r        ',             4 ';         $'
  PSP_add  db 'PSP:       dup(?), ';
  r        ',           4 $'
  MCB_siz  db           ',   dup(?),   байт', ';
  e        'Размер:     6    '         $'
  SC_SD    'SC/CD:      13
  db       $',          ,   10, '$'
```

```
          ext_report db 'Выделение доп. памяти $успешно.', 13, 10, '$отменено.',
13, 10, '$'

             edge:
      end main
```