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. 9381 _____ Любимов В.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

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.

Описание функций и структуры данных

1. TETR_TO_HEX – переводит число, представляемое четырьмя младшими битами в регистре AL, в 16-ричную цифру-символ.
2. BYTE_TO_HEX - переводит число, содержащиеся в регистре AL, в 16-ричные цифры, записывающиеся в регистры AL и AH.
3. WRD_TO_HEX – переводит слово в регистре AX в четыре 16-ричные цифры, записывающие по адресу, находящемуся в DI.
4. BYTE_TO_DEC - переводит число, содержащиеся в регистре AL, в 10-ричные цифры, записывающиеся по адресу, находящемуся в SI.
5. PRINT_MES – при помощи функции 9h из прерывания 21h выводит строку на экран.
6. HEX_TO_DEC – переводит шестнадцатеричное число в десятичное и записывает результат по адресу в di.

7. GET_FREE_MEM – получает количество доступной памяти и при помощи функции HEX_TO_DEC записывает результат в десятичном формате по адресу в di.

8. GET_EXTENDED_MEM – получает количество расширенной памяти и при помощи функции HEX_TO_DEC записывает результат в десятичном формате по адресу в di.

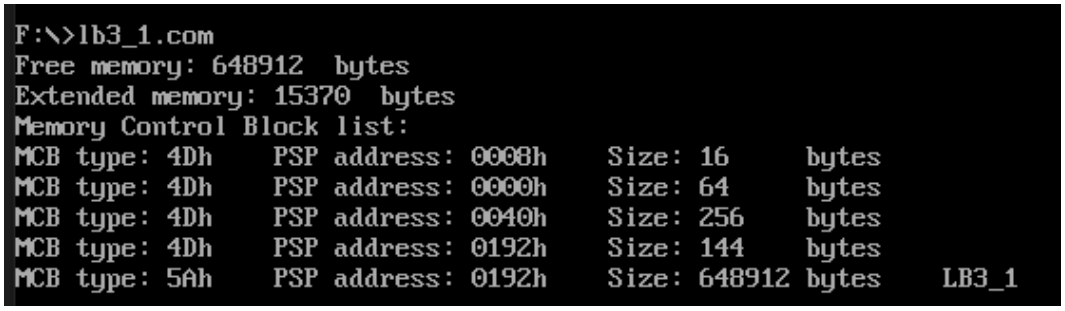
9. GET_MCB – выводит содержимое цепочки блоков управления памятью (MCB).

10. MY_FREE – освобождает память, не используемую программой.

11. TAKE_MEM – запрашивает 64 килобайта памяти. Выводит сообщение об ошибки или успехи в зависимости от успеха запроса памяти.

Ход выполнения работы

1. Разработан и написан модуль типа .COM выводящий на экран количество доступной и расширенной памяти и список блоков управления памятью.



2. Исходная программа доработана так, что она освобождает не занимаемую ею память.

```
F:\>lb3_2.com
Free memory: 648912  bytes
Extended memory: 15370  bytes
Memory Control Block list:
MCB type: 4Dh    PSP address: 0008h    Size: 16     bytes
MCB type: 4Dh    PSP address: 0000h    Size: 64     bytes
MCB type: 4Dh    PSP address: 0040h    Size: 256    bytes
MCB type: 4Dh    PSP address: 0192h    Size: 144    bytes
MCB type: 4Dh    PSP address: 0192h    Size: 4096   bytes    LB3_2
MCB type: 5Ah    PSP address: 0000h    Size: 644800 bytes
```

3. Программа доработана так, что после освобождения памяти она запрашивает 64 килобайта функцией 48h прерывания 21h. На экран выводится информация об успешности запроса памяти.

```
F:\>lb3_3.com
Free memory: 648912  bytes
Extended memory: 15370  bytes
64 kb has been suceessfuly requested!
Memory Control Block list:
MCB type: 4Dh    PSP address: 0008h    Size: 16     bytes
MCB type: 4Dh    PSP address: 0000h    Size: 64     bytes
MCB type: 4Dh    PSP address: 0040h    Size: 256    bytes
MCB type: 4Dh    PSP address: 0192h    Size: 144    bytes
MCB type: 4Dh    PSP address: 0192h    Size: 4096   bytes    LB3_3
MCB type: 4Dh    PSP address: 0192h    Size: 65536  bytes    LB3_3
MCB type: 5Ah    PSP address: 0000h    Size: 579248 bytes
```

4. Программа доработана так, что перед освобождением памяти она запрашивает 64 килобайта функцией 48h прерывания 21h. На экран выводится информация об успешности запроса памяти.

```
F:\>lb3_4.com
Free memory: 648912  bytes
Extended memory: 15370  bytes
Error has been happened during memory requesting!
Memory Control Block list:
MCB type: 4Dh    PSP address: 0008h    Size: 16     bytes
MCB type: 4Dh    PSP address: 0000h    Size: 64     bytes
MCB type: 4Dh    PSP address: 0040h    Size: 256    bytes
MCB type: 4Dh    PSP address: 0192h    Size: 144    bytes
MCB type: 4Dh    PSP address: 0192h    Size: 4096   bytes    LB3_4
MCB type: 5Ah    PSP address: 0000h    Size: 644800 bytes      ¤┬♥.â
```

Ответы на контрольные вопросы

1. Что означает “доступный объем памяти”?

Максимальный объём памяти, доступный программам для использования.

2. Где MCB блок Вашей программы в списке?

У моей программы значение адреса её PSP равно 0192h, то есть поле “PSP address” имеет значение 0192h во всех блоках управления памятью, относящихся к моей программе.

3. Какой размер памяти занимает программа в каждом случае?

1. 648912 байт – вся доступная память
2. 4096 байт – только необходимая для работы память
3. 4096 + 65536 = 69632 байт – необходимая для работы память и выделенные 64 килобайта
4. 4096 байт – так как запрос на выделение памяти производился до её освобождения, то вся память была занята нашей программой и получить 64 килобайта памяти не удалось.

Вывод

В ходе работы изучены структуры данных и работа функций управления памятью ядра операционной системы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

lab3_3.asm

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

Start:
    jmp     Begin

free_mem_mes db 'Free memory: ', '$'
free_mem db '        bytes', 0DH, 0AH, '$'
extended_mem_mes db 'Extended memory: ', '$'
extended_mem db '       bytes', 0DH, 0AH, '$'
mcb_table db 'Memory Control Block list: ', 0DH, 0AH, '$'
mcb_type db 'MCB type:   h    ', '$'
mcb_adress db 'PSP address:     h    ', '$'
mcb_size db 'Size:        bytes    ', '$'
mem_get_success db '64 kb has been suceessfuly requested!', 0DH, 0AH, '$'
mem_get_fail db 'Error has been happened during memory requesting!', 0DH, 0AH, '$'
some_mes db 0DH, 0AH, '$'

TETR_TO_HEX PROC NEAR
    and     al, 0Fh
    cmp     al, 09
    jbe     Next
    add     al, 07
Next:
    add     al, 30h
    ret
TETR_TO_HEX ENDP

BYTE_TO_HEX PROC NEAR
    push    cx
    mov     ah, al
    call    TETR_TO_HEX
    xchg    al, ah
    mov     cl, 4
    shr     al, cl
    call    TETR_TO_HEX
    pop     cx
    ret
BYTE_TO_HEX ENDP

WRD_TO_HEX PROC near
    push    bx
    mov     bh, ah
    call    BYTE_TO_HEX
    mov     [di], ah
    dec     di
    mov     [di], al
    dec     di
    mov     al, bh
    call    BYTE_TO_HEX
    mov     [di], ah
    dec     di
    mov     [di], al
```

```
    pop     bx
    ret
WRD_TO_HEX ENDP

BYTE_TO_DEC PROC near
    push    cx
    push    dx
    xor     ah, ah
    xor     dx, dx
    mov     cx, 10
Loop_bd:
    div     cx
    or      dl, 30h
    mov     [si], dl
    dec     si
    xor     dx, dx
    cmp     ax, 10
    jae     Loop_bd
    cmp     al, 00h
    je      End_l
    or      al, 30h
    mov     [si], al
End_l:
    pop     dx
    pop     cx
    ret
BYTE_TO_DEC ENDP

PRINT_MES  PROC near
    push    ax
    mov     ah, 9
    int     21h
    pop     ax
    ret
PRINT_MES ENDP

HEX_TO_DEC PROC near
    mov     bx, 0Ah
    xor     cx, cx

byte_step:
    div     bx
    push    dx
    inc     cx
    xor     dx, dx
    cmp     ax, 0
    jne     byte_step

add_symbol:
    pop     dx
    or      dl, 30h

    mov     [di], dl
    inc     di
    loop    add_symbol
    ret
HEX_TO_DEC ENDP

GET_FREE_MEM PROC near
    mov     ah, 4ah
    mov     bx, 0ffffh
    int     21h
```

```
    mov     ax, bx
    mov     bx, 16
    mul     bx
    call    HEX_TO_DEC
    ret
GET_FREE_MEM ENDP

GET_EXTENDED_MEM PROC near
    mov     al, 30h
    out     70h, al
    in      al, 71h
    mov     al, 31h
    out     70h, al
    in      al, 71h
    mov     bh, al
    mov     ax, bx

    mov     bx, 1h
    mul     bx
    call    HEX_TO_DEC
    ret
GET_EXTENDED_MEM ENDP

GET_MCB PROC near
    mov     ah, 52h
    int     21h
    mov     ax, es:[bx-2]
    mov     es, ax

mcb:
    mov       al, es:[0]
    call      BYTE_TO_HEX
    mov             di, offset mcb_type
    add     di, 10
    mov       [di], ax
    mov             dx, offset mcb_type
    call      PRINT_MES

    mov     ax, es:[1]
    mov     di, offset mcb_adress + 16
    call    WRD_TO_HEX
    mov     dx, offset mcb_adress
    call    PRINT_MES

    mov     ax, es:[3]
    mov     di, offset mcb_size + 6
    mov     bx, 16
    mul     bx
    call    HEX_TO_DEC
    mov     dx, offset mcb_size
    call    PRINT_MES

    mov     bx, 8
    mov cx, 7

scsd:
    mov     dl, es:[bx]
    mov     ah, 02h
    int     21h
    inc     bx
    loop    scsd
```

```
    mov     dx, offset some_mes
    call    PRINT_MES

    mov     bx, es:[3h]
    mov     al, es:[0h]
    cmp     al, 5Ah
    je      end_point

    mov     ax, es
    inc     ax
    add     ax, bx
    mov     es, ax
    jmp     mcb

end_point:
    ret
GET_MCB ENDP

MY_FREE PROC near
    push ax
    push bx

    mov bx, 100h
    mov ah, 4ah
    int 21h

    pop bx
    pop ax

    ret
MY_FREE ENDP

TAKE_MEM PROC near
    mov ah, 48h
    mov bx, 1000h
    int 21h

    jb fail
    jmp success

fail:
    mov dx, offset mem_get_fail
    call PRINT_MES
    jmp take_mem_end

success:
    mov dx, offset mem_get_success
    call PRINT_MES

take_mem_end:
    ret
TAKE_MEM ENDP

Begin:
    mov     dx, offset free_mem_mes
    call    PRINT_MES
    mov     di, offset free_mem
    call    GET_FREE_MEM

    mov     dx, offset free_mem
    call    PRINT_MES
    mov     dx, offset extended_mem_mes
```

```
    call    PRINT_MES

    mov     di, offset extended_mem
    call    GET_EXTENDED_MEM
    mov     dx, offset extended_mem
    call    PRINT_MES

    call MY_FREE
    call TAKE_MEM

    mov     dx, offset mcb_table
    call    PRINT_MES
    call    GET_MCB

    xor     al, al
    mov     ah, 4ch
    int     21h
MAIN ENDS
            END     Start
```