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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Операционные системы»

Тема: Сопряжение стандартного и пользовательского обработчиков прерываний

Студентка гр. 7381 _____ Машина Ю. Д.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2019

Постановка задачи.

Исследовать возможность встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Пользовательский обработчик прерывания получает управление по прерыванию при нажатии клавиши на клавиатуре. Он обрабатывает скан-код и осуществляет определенные действия, если скан-код совпадает с определенными кодами, которые он должен обрабатывать. Если скан-код не совпадает с этими кодами, то управление передается стандартному прерыванию.

Процедуры:

- MAIN – главная процедура.
- ROUT – функция обработки прерывания, при нажатии клавиши левый Ctl, выводит символ «@»;
- Print – выводит сообщение на экран;
- Set_Interruption – устанавливает пользовательское прерывание;
- Rem_Int – удаляет пользовательское прерывание в поле векторов прерываний;

Структуры данных:

- alreadyloaded – строка, оповещающая о том, что прерывание уже было загружено;
- loaded – строка, оповещающая о том, что прерывание загружено;
- unloaded – строка, оповещающая о том, что прерывание выгружено;
- IDFN – идентификатор пользовательского прерывания;

- KEEP_IP – переменная, хранящая значение регистра IP;
- KEEP_CS – переменная, хранящая значение регистра CS;
- KEEP_PSP – переменная, хранящая значение PSP;
- KEEP_SS – переменная, хранящая значение регистра SS;

- KEEP_AX – переменная, хранящая значение регистра AX;
- KEEP_SP – переменная, хранящая значение регистра SP;
- REQ_KEY – переменная, хранящая скан-код для обработки нажатия левой клавиши Ctrl.

Ход работы.

Был написан текст исходного .EXE модуля, программа которого проверяет, установлено ли пользовательское прерывание с вектором 09h. Если прерывание не установлено, то устанавливает резидентную функцию для обработки пребывания и настраивает вектор прерываний. Если же прерывание установлено, то выводится соответствующее сообщение. В обоих случаях осуществляется выход по функции 4Ch прерывания int 21h. Выгрузка прерывания по соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем осуществляет выход по функции 4Ch прерывания int 21h. Результаты работы программы приведены на рис. 1, рис. 2, рис. 3, рис. 4.

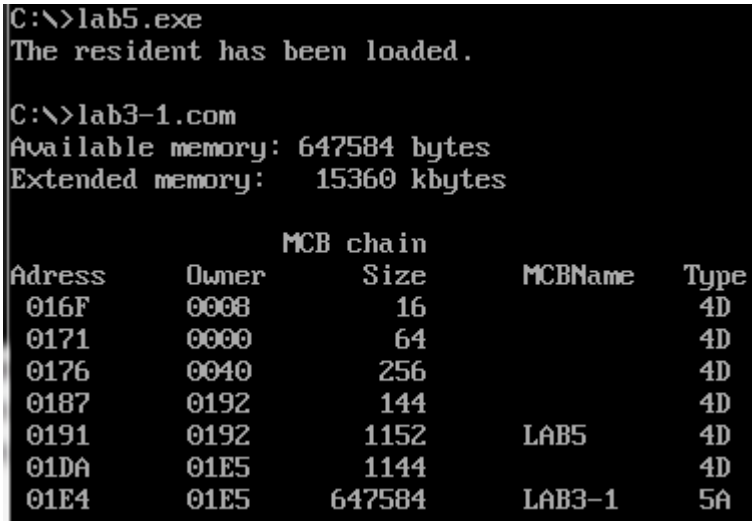

Рисунок 1 — Загрузка обработчика прерываний

Рисунок 2 — Повторная загрузка обработчика прерываний

Рисунок 3 — Проверка изменения поведения нажатия клавиши ctrl

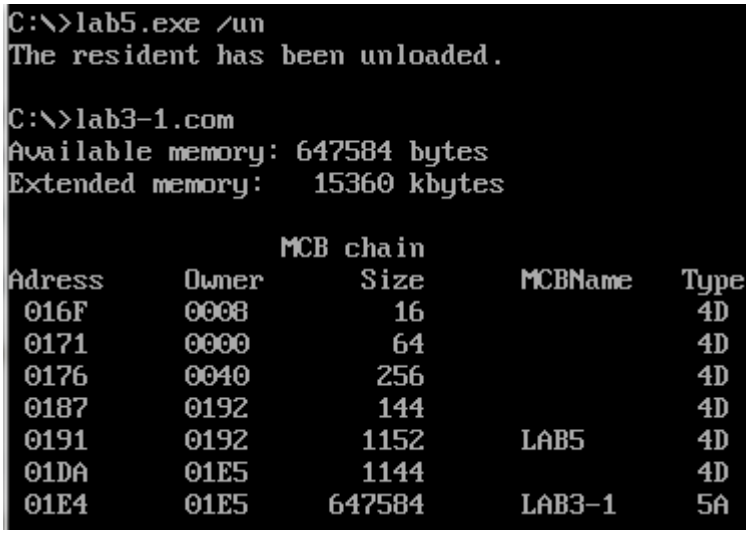

Рисунок 4 — Выгрузка обработчика прерываний и состояние памяти после выгрузки

Ответы на контрольные вопросы.

1) Какого типа прерывания использовались в работе?

Ответ: программные прерывания 21h, 16h, 35h, 25h, а также пользовательское прерывание 09h.

2) Чем отличается скан-код от кода ASCII?

Ответ: грубо говоря, скан-код характеризует клавишу, каждой клавише соответствует свой скан-код. Таким образом, если стоит английский язык, то скан-коду 16 будет соответствовать символ «Q», если же поставлен французский язык, то тому же скан-коду 16 будет

соответствовать символ «A». Код ASCII (не связан напрямую с клавиатурой) определяет закрепленный на ней символ.

Выводы.

В данной лабораторной работе была исследована возможность встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Был построен пользовательский обработчик прерываний, встроенный в стандартный обработчик от клавиатуры.

ПРИЛОЖЕНИЕ А

КОД ИСХОДНОЙ ПРОГРАММЫ

```
AStack SEGMENT STACK
     DW 100h DUP(?)
AStack ENDS

DATA SEGMENT
    alreadyloaded db 'The resident has already been
loaded.',0DH,0AH,'$'
    unloaded db 'The resident has been unloaded.',0DH,0AH,'$'
    loaded db 'The resident has been loaded.',0DH,0AH,'$'
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, ES:DATA, SS:AStack

Print PROC NEAR
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
Print ENDP

ROUT PROC FAR
jmp ROUT_begin

IDFN db  '0000'
KEEP_AX dw  0
KEEP_SS dw  0
KEEP_SP dw  0
KEEP_IP dw  0
KEEP_CS dw  0
KEEP_PSP dw  0

REQ_KEY db  1Dh
INTER_STACK dw  64 dup (?)
END_STACK dw  0

ROUT_begin:
    mov KEEP_AX,ax
    mov KEEP_SS,ss
    mov KEEP_SP,sp
    mov ax,cs
    mov ss,ax
    mov sp,offset END_STACK
    mov ax,KEEP_AX
    push ax
    push dx
    push ds
    push es
    in al,60H ; читать ключ
```

```
    cmp al,REQ_KEY ; это требуемый код?
    je do_req ; да, активизировать обработку REQ_KEY, нет-уйти на
исходный обработчик
    pushf ; Сохранить в стеке регистр FLAGS
    call dword ptr cs:KEEP_IP
    jmp end_ROUT
do_req:
    push ax
    ;для обработки аппаратого прерывания
    in al,61h ; взять значение порта управления клавиатурой
    mov ah, al ; сохранить его
    or al,80h ; установить бит разрешения для клавиатуры
    out 61h,al ; вывести его в управляющий порт
    xchg ah,al  ; извлечь исходное значение порта
    out 61h,al ; и записать его обратно
    mov al,20h ; послать сигнал "конец прерывания"
    out 20h,al ; контроллеру прерываний 8259
    pop ax
add_to_buff:
    mov cl,'@' ;
    mov ah,05h ; запись символа в буфер клавиатуры
    mov ch,00h
    int 16h
    or al, al ; проверка переполнения буфера
    jz end_ROUT
    mov ax,es:[1Ah]
    mov es:[1Ch],ax
    jmp add_to_buff
end_ROUT:
    pop es
    pop ds
    pop dx
    pop ax
    mov ss,KEEP_SS
    mov sp,KEEP_SP

    mov al,20h
    out 20h,al
;Если аппаратное прерывание не заканчивается этими строками,
то
;микросхема 8259 не очистит  информацию  регистра  обслуживания,  с
тем
;чтобы была разрешена обработка прерываний с более низкими уровнями
    mov ax,KEEP_AX
    iret
LAST_BYTE:
ROUT ENDP

Set_Interruption PROC
    push ax
    push dx
    push ds
    mov ah,35h ; дать вектор прерывания
    mov al,09h ; номер прерывания
```

```
    int 21h
    mov KEEP_IP,bx
    mov KEEP_CS,es
    mov dx,offset ROUT
    mov ax,seg ROUT
    mov ds,ax ; вектор прерывания: адрес программы обработки
прерывания
    mov ah,25h ; установить вектор прерывания
    mov al,09h ; номер прерывания
    int 21h
    pop ds
    mov dx,offset loaded
    call Print
    pop dx
    pop ax
    ret
Set_Interruption ENDP

Rem_Int PROC
    push ax
    push ds
    CLI
    mov ah,35h ; функция получения вектора
    mov al,09h
    int 21h
    mov si,offset KEEP_IP
    sub si,offset ROUT
    mov dx,es:[bx+si]
    mov ax,es:[bx+si+2]
    mov ds,ax
    mov ah,25h ; функция установки вектора
    mov al,09h
    int 21h
    pop ds
    mov ax,es:[bx+si-2]
    mov es,ax
    mov ax,es:[2Ch]
    push es
    mov es,ax
    mov ah,49h ; Освободить распределенный блок памяти
    int 21h
    pop es
    mov ah,49h
    int 21h
    STI
    pop ax
    ret
Rem_Int ENDP

MAIN PROC Far
    mov ax,DATA
    mov ds,ax
    mov KEEP_PSP,es
    mov ah,35h
```

```
    mov al,09h
    int 21h
    mov si,offset IDFN ; сигнатура, идентифицирующая резидент
    sub si,offset ROUT
    mov ax,'00'
    cmp ax,es:[bx+si]
    jne not_loaded
    cmp ax,es:[bx+si+2]
    je loadd
not_loaded:
    call Set_Interruption
    mov dx,offset LAST_BYTE
    mov cl,4
    shr dx,cl
    inc dx
    add dx,CODE
    sub dx,KEEP_PSP
    xor al,al
    mov ah,31h ;завершиться и остаться резидентным
    int 21h
loadd:
    push es
    push ax
    mov ax,KEEP_PSP
    mov es,ax
    mov al,es:[82h]
    cmp al,'/'
    jne not_unloaded
    mov al,es:[83h]
    cmp al,'u'
    jne not_unloaded
    mov al,es:[84h]
    cmp al,'n'
    je unload
not_unloaded:
    pop ax
    pop es
    mov dx,offset alreadyloaded
    call Print
    jmp ending
unload:
    pop ax
    pop es
    call Rem_Int
    mov dx,offset unloaded
    call Print
ending:
    xor al,al
    mov ah,4Ch ; выход
    int 21H
MAIN ENDP
CODE ENDS

END MAIN
```