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

ОТЧЕТ

по практической работе №5
по дисциплине «Операционные системы»
Тема: Сопряжение стандартного и пользовательского обработчиков прерываний

Студент гр. 8381 Киреев К.А.

Преподаватель Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Пользовательский обработчик прерываний получает управление по прерыванию (int 09h) 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 стандартном прерыванию.

Основные теоретические положения.

Клавиатура содержит микропроцессор, который воспринимает каждое нажатие на клавишу и посылает скан-код в порт микросхемы интерфейса с периферией. Когда скан-код поступает в порт, то вызывается аппаратное прерывание клавиатуры (INT 09H). Процедура обработки этого прерывания считывает номер клавиши из порта 60H, преобразует номер клавиши в соответствующий код, выполняет установку флагов в байтах состояния, загружает номер клавиши и полученный код в буфер клавиатуры.

В прерывании клавиатуры можно выделить три основных шага:

1. Прочитать скан-код и послать клавиатуре подтверждающий сигнал.

2. Преобразовать скан-код в номер кода или в установку регистра статуса клавиш-переключателей.

3. Поместить код клавиши в буфер клавиатуры.

Текущее содержимое буфера клавиатуры определяется указателями на начало и конец записи. Расположение в памяти необходимых данных представлено в таблице 1.

Таблица 1 – Буфер клавиатуры

Адрес в памяти	Размер в байтах	Содержимое
0040:001A	2	Адрес начала буфера клавиатуры
0040:001C	2	Адрес конца буфера клавиатуры
0040:001E	32	Буфер клавиатуры
0040:0017	2	Байты состояния

Флаги в байтах состояния устанавливаются в 1, если нажата соответствующая клавиша или установлен режим. Соответствие флагов и клавиш показано ниже на рис. 1.

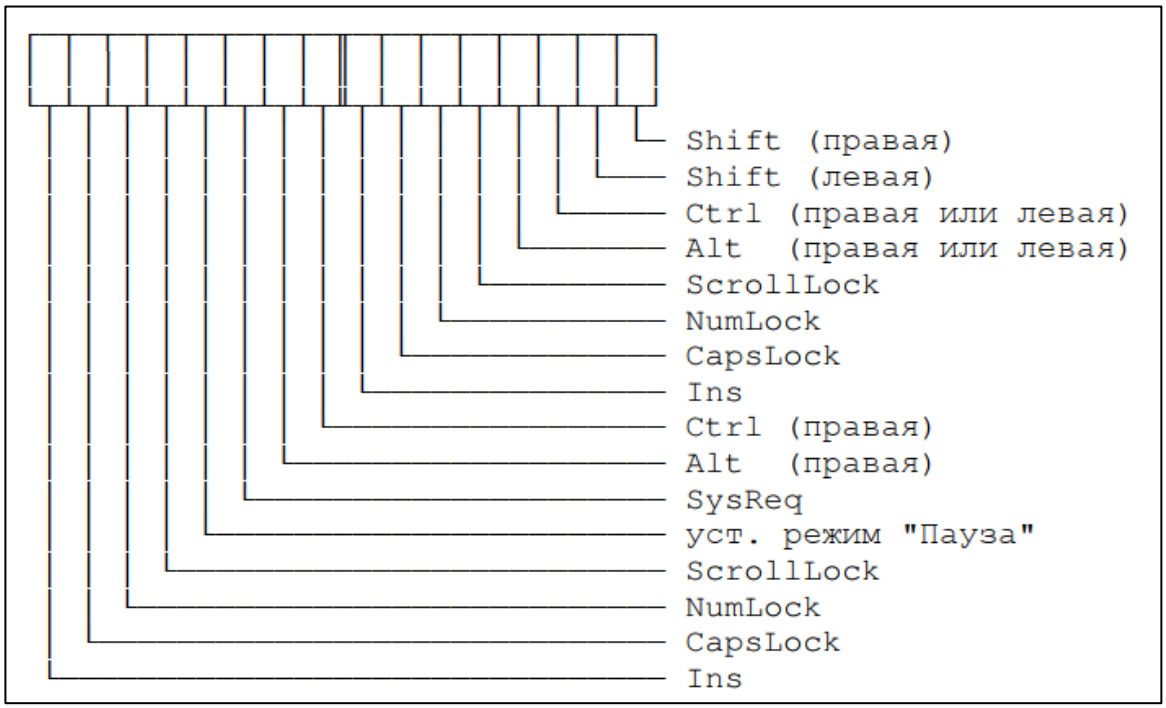

Рисунок 1 – Соответствие флагов и клавиш

Выполнение работы.

Написан текст исходного EXE модуля, который выполняет некоторые функции. Проверяет, установлено ли пользовательское прерывание с вектором 09H. Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляется выход о функции 4Ch прерывания int 21h. Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h. Выгрузка прерывания о соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем осуществляется выход по функции 4Ch прерывания int 21h.

Полученный исходный модуль был отлажен. Результаты выполнения программы представлены на рис. 2.

Рисунок 2 – Результат выполнения OS5.EXE

Работа прерывания была проверена введением различных символов, обрабатываемых установленным обработчиком и стандартным обработчиком.

Обработка прерывания:

- Клавиши нечетных цифр выводят символ ‘!’
- Клавиши четных цифр выводят символ ‘?’
- Клавиши qwerty заменяются на 012345 соответственно
- По клавише ‘c’ включается CapsLock и печатается пробел
- По клавише ‘b’ выключается CapsLock и печатается пробел

Необходимо было проверить размещение прерывания в памяти. Для этого была запущена программа OS3A.COM, которая отображает карту памяти в виде списка блоков MCB. Результат выполнения программы представлен на рис. 3.

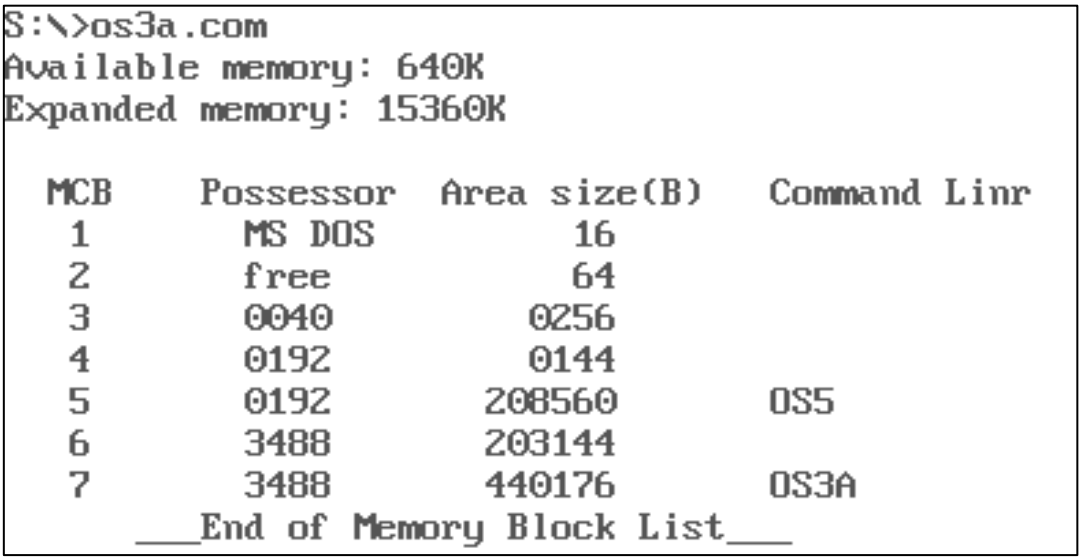

Рисунок 3 – Состояние памяти после загрузки прерывания

После повторного запуска программа определила установленный обработчик прерываний. Результат выполнения программы представлен на рис. 4.

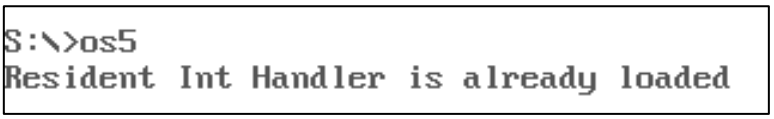

Рисунок 4 – Повторный запуск программы

Далее программа была запущена с ключом выгрузки, чтобы убедиться, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также была запущена программа OS3A.COM. Результаты выполнения программы представлен на рис. 5.

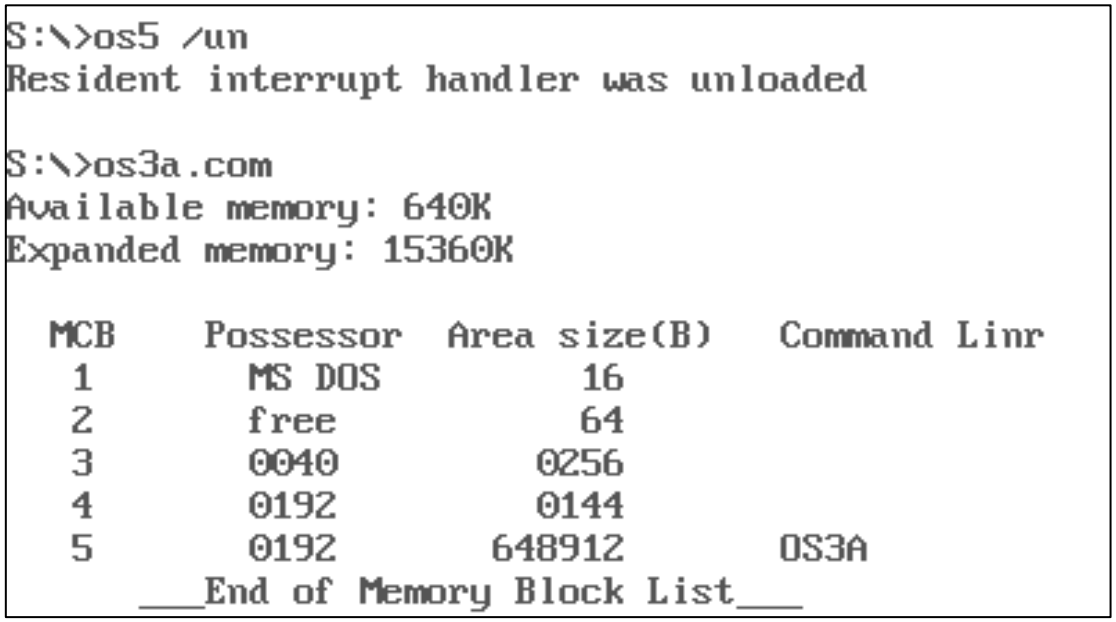

Рисунок 5 – Состояние памяти после выгрузки прерывания

Контрольные вопросы

- **Какого типа прерывания использовались в работе?**
  - Аппаратные прерывания (INT 09H)
  - Прерывания функций BIOS для обслуживания аппаратуры компьютера (INT 16H)
  - Прерывания функций DOS (INT 21H)

- **Чем отличается скан-код от кода ASCII?**

Если скан-код характеризует клавишу, которая была нажата, то код ASCII определяет закрепленный за ней символ.

Вывод.

В результате выполнения данной лабораторной работы были исследованы возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. OS4.ASM

```
Astack segment stack
    dw 256 dup(?)
Astack ends

data segment
    load_msg db 'Resident Interrupt Handler was loaded', 13,
10, '$'
     alrd_load_msg  db  'Resident  Int  Handler  is  already
loaded', 13, 10, '$'
    unload_msg db 'Resident interrupt handler was unloaded',
13, 10, '$'
data ends

code segment
    assume cs:code, ds:data, ss:Astack
rout proc far
    jmp rout_start
     signature  dw  4321h  ;сигнатура,  которая  идентифицирует
резидент
     keep_psp dw ?
    keep_ip dw ?
    keep_cs dw ?
     sign db ?
     rout_start:
    push ax
    push bx
    push cx
    push dx
    push si
    push di
    push es
    push ds
     mov ax, seg sign
     mov ds, ax
     in al, 60h
```

```
    irpc case, 2468A ;клавиши нечетных цифр выводят !
    cmp al, 0&case&h
    je type_M
    endm
    irpc case, 3579 ;клавиши четных цифр выводят ?
    cmp al, 0&case&h
    je type_Z
    endm
    irpc case, 012345 ;qwerty заменяется на 012345
    cmp al, 1&case&h
    je type_qy&case&
    endm

    cmp al, 2Eh ;по клавише с включается CapsLk и печатается
пробел
    je on_CL
    cmp al, 30h ;по клавише b выключается CapsLk и печатается
пробел
    je off_CL

    pushf
    call dword ptr cs:keep_ip
    jmp rout_ending

    irpc met, 012345
    type_qy&met&:
        mov sign, 3&met&h
        jmp signal
    endm
    type_M:
        mov  sign, '!'
        jmp  signal
    type_Z:
        mov  sign, '?'
        jmp  signal

    on_CL:
```

```
    xor ax, ax
    mov es, ax
    mov al, 01000000b ;готовим бит 6 (CapsLk) к установке
    or es:[417h], al ;меняем байт статуса
    mov  sign, ' '
    jmp  signal
off_CL:
    xor ax, ax
    mov es, ax
    mov al, 10111111b ;сбрасываем бит 6
    and es:[417h], al ;меняем байт статуса
    mov  sign, ' '
    jmp  signal
;сигнал подтверждения микропроцессору клавиатуры
signal:
    in al, 61h ;читаем состояние порта 61h
    mov ah, al
    or al, 80h ;устанавливаем бит 7
    out 61h, al ;посылаем измененный байт в порт
    xchg ah, al
    out 61h, al ;возвращаем состочние порта 61h
    mov al, 20h
    out 20h, al
record_sign:
    mov ah, 05h
    mov cl, sign ;пишем символ в буфер клавиатуры
    mov ch, 00h
    int 16h
    or al, al ;проверка переполнения буфера
    jz rout_ending
    cli ;запрещаем прерывания
    xor ax, ax
    mov es, ax
    mov al, es:[41Ah] ;указатель на голову буфера
    mov es:[41Ch], al ;посылаем его в указатель хвоста
    sti ;разрешаем прерывания
    jmp record_sign
rout_ending:
```

```
     pop ds
     pop es
    pop di
     pop  si
     pop dx
     pop cx
     pop bx
     pop ax
    mov al, 20h
    out 20h, al
    iret
rout endp
last_byte:

print proc near
     push ax
     mov ah, 09h
     int 21h
     pop ax
     ret
print endp

rout_load proc near
    push ax
     push bx
     push cx
     push dx
    push es
     push ds
     mov ah, 62h
     int 21h
     push bx
     pop es ;в es PSP
     Interrupt_handler_load: ;загрузка обработчика прерывания
     mov ah, 35h ;функция получения вектора
     mov al, 09h ;номер вектора
     int 21h ;es:bx - адрес обработчика прерывания
     mov keep_cs, es ;запоминание сегмента
```

```
    mov keep_ip, bx ;запоминание смещения
     lea  dx, load_msg
     call print
    ;для функции 25h прерывания 21h
    ;al - номер прерывания
    ;ds:dx - адрес программы обработки прерывания
     lea dx, rout ;смещение процедуры
     mov ax, seg rout ;сегмент процедуры
     mov ds, ax
     mov ah, 25h ;функция установки вектора
     mov al, 09h ;номер вектора
     int 21h ;замена прерывания
     pop ds
    ;для функции 31h прерывания 21h
    ;al - код выхода
    ;dx - объем памяти, оставляемой резидентной, в параграфах
    ;выходит в родительский процесс, сохраняя код выхода в al
    ;DOS устанавливает начальное распределение памяти
    ;далее возвращает управление родительскому процессу, оставляя указанную память резидентной
    lea di, last_byte
     mov dx, (di+10Fh)/16
     ;К длине резидентной части программы прибавляется размер PSP (1OOh) и еще число 15 (Fh),
     ;чтобы после получения размера программы в параграфах результат был округлен в большую сторону
     xor al, al ;0 - нормальное завершение
     mov ah, 31h
     int 21h
     mov ah, 4Ch
     int 21h
    pop es
    pop dx
     pop cx
     pop bx
     pop ax
     ret
rout_load endp
```

```
rout_unload proc near
    cli
     push ax
     push bx
     push cx
     push dx
     push ds
     push es
     push si
     push di
     mov ah, 62h
     int 21h
     push bx
     pop es ;в es PSP
     ;командная строка при запуске программы находится по
адресу es:[80h]
     cmp byte ptr es:[82h], '/'
     jne alrd_load_rout
     cmp byte ptr es:[83h], 'u'
     jne alrd_load_rout
     cmp byte ptr es:[84h], 'n'
     jne alrd_load_rout
     lea  dx, unload_msg
     call print
     mov ah, 35h ;функция получения вектора
     mov al, 09h ;номер вектора
     int 21h ;es:bx - адрес обработчика прерывания
     push ds
     mov si, offset keep_ip
     sub si, offset rout ;si - смещение ip
     mov dx, es:[bx+si] ;адрес ip
     mov ax, es:[bx+si+2] ;адрес cs
     mov ds, ax
     mov ah, 25h
     mov al, 09h
     int 21h
     pop ds
```

```
    mov ax, es:[bx+si-2] ;адрес psp
     mov es, ax
     push es
     mov ax, es:[2ch] ;сегментный адрес среды
     mov es, ax
     ;DOS Function 49H: Освободить распределенный блок памяти
     mov ah, 49h
     int 21h
     pop es ;адрес psp
     mov ah, 49h
     int 21h
     jmp unload_ending
     alrd_load_rout:
     mov dx, offset alrd_load_msg
     call print
     unload_ending:
    sti
     xor ax, ax
     mov es, ax
     mov al, 10111111b
     and es:[417h], al
     pop di
    pop si
     pop es
     pop ds
     pop dx
     pop cx
     pop bx
     pop ax
     ret
rout_unload endp

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

```
     mov  keep_psp, es
    mov ah, 35h ;функция получения вектора
     mov al, 09h ;номер вектора
     int 21h ;es:bx - адрес обработчика прерывания
     lea di, signature ;адрес, записанный в векторе прерывания
     sub di, offset rout ;di - смещение сигнатуры
     cmp ES:[bx+di], 4321h ;сравнение значения сигнатуры с
реальным кодом
     je rl ;если совпадают, то резидент установлен
     call rout_load ;иначе не установлен
     rl:  call rout_unload
     mov ax, 4C00h
     int 21h
     ret
main endp
code ends
end main
```