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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Организация ЭВМ и систем»

Тема: Разработка собственного прерывания.

Студент гр. 9383 _____ Рыбников Р.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить основные принципы работы с прерываниями. Применить на практике полученные знания о прерываниях, разработав своё собственное прерывание.

Краткие сведения.

Прерывание - это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств (обработка сигнала таймера, нажатия клавиши и т.д.). Когда возникает прерывание, процессор прекращает выполнение текущей программы (если ее приоритет ниже) и запоминает в стеке вместе с регистром флагов адрес возврата(CS:IP) - места, с которого будет продолжена прерванная программа. Затем в CS:IP загружается адрес программы обработки прерывания и ей передается управление. Адреса 256 программ обработки прерываний, так называемые векторы прерывания, имеют длину по 4 байта (в первых двух хранится значение IP , во вторых - CS) и хранятся в младших 1024 байтах памяти. Программа обработки прерывания должна заканчиваться инструкцией IRET (возврат из прерывания), по которой из стека восстанавливается адрес возврата и регистр флагов.

Программа обработки прерывания - это отдельная процедура, имеющая структуру:

```
SUBR_INT PROC FAR
PUSH AX ; сохранение изменяемых регистров
...
<действия по обработке прерывания>
POP AX ; восстановление регистров
...
MOV AL, 20H
OUT 20H,AL
IRET
SUBR_INT ENDP
```

Две последние строки обработчика прерывания, указанные перед командой IRET выхода из прерывания, необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное.

Замечание: в лабораторной работе действиями по обработке прерывания может быть вывод на экран некоторого текста, вставка цикла задержки в вывод сообщения или включение звукового сигнала.

Программа, использующая новые программы обработки прерываний при своем завершении должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H возвращает текущее значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. В соответствии с этим, программа должна содержать следующие инструкции:

```
; -- в сегменте данных
KEEP_CS DW 0 ; для хранения сегмента
KEEP_IP DW 0 ; и смещения вектора прерывания
; -- в начале программы
MOV AH, 35H ; функция получения вектора
MOV AL, 1CH ; номер вектора
INT 21H
MOV KEEP_IP, BX ; запоминание смещения
MOV KEEP_CS, ES ; и сегмента вектора прерывания
```

Для установки адреса нового обработчика прерывания в поле векторов прерываний используется функция 25H прерывания 21H, которая помещает заданные адреса сегмента и смещения обработчика в вектор прерывания с заданным номером.

```
PUSH DS
MOV DX, OFFSET ROUT ; смещение для процедуры в DX
MOV AX, SEG ROUT ; сегмент процедуры
MOV DS, AX ; помещаем в DS
MOV AH, 25H ; функция установки вектора
MOV AL, 60H ; номер вектора
```

```
INT 21H ; меняем прерывание
POP DS
```

Далее может выполняться вызов нового обработчика прерывания.

В конце программы восстанавливается старый вектор прерывания

```
CLI
PUSH DS
MOV DX, KEEP_IP
MOV AX, KEEP_CS
MOV DS, AX
MOV AH, 25H
MOV AL, 1CH
INT 21H ; восстанавливаем старый вектор прерывания
POP DS
STI
```

Для работы с портами существуют специальные команды IN и OUT. IN вводит значение из порта ввода-вывода, OUT выводит.

43h - запись управляющего слова в регистр режима канала

42h - загрузка счетчика канала 2, чтение счетчика канала 2

Канал 2 - генератор звука

Инструкция loop уменьшает значение в регистре CX в реальном режиме или ECX в защищённом. Если после этого значение CX не ноль, то команда loop прыгает на метку.

Команды CLI и STI служат для установки или сброса флага прерываний, что позволяет включать или отключать реакцию на внешние прерывания. Команда CLI(Clear Interrupt flag) сбрасывает флаг IF в значение 0, что запрещает прерывания. Команда STI (Set -//-) устанавливает флаг IF в значение 1, что разрешает прерывания.

Текст задания.

Вариант 2B.

Разработать собственное прерывание.

1 - 60h - прерывание пользователя - должно генерироваться в программе;

B - Выдача звукового сигнала;

Ход работы.

Была реализована программа, которая с помощью прерывания 60h активирует звуковой сигнал.

Чтобы это сделать, был сохранён сегмент вектора прерывания и смещение изначального прерывания 60h.

После прерывание 60h было переопределено под вызов функции SUBR_INT, в которой выполняется воспроизведение звукового сигнала через динамик.

После воспроизведения звукового сигнала, старый вектор прерывания восстанавливается обратно.

Функция MAIN завершается операндом ret.

Исходный код программы представлен в приложении А.

Тестирование программы.

При запуске программы пользователь услышит из динамика короткий звуковой сигнал установленной частоты.

Выводы.

Были изучены основные принципы работы с прерываниями. Также были применены на практике полученные знания о прерываниях, было разработано собственное прерывание, воспроизводящее звуковой сигнал.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

main.asm

```
;
_____________________________________________________________________
_______
; Задание: 60h - прерывание пользователя - должно генерироваться в
программе |
;              выдача звукового сигнала (действие, реализуемое          |
;              программой обработки прерывания)                         |
;
_____________________________________________________________________
_______|
AStack SEGMENT STACK
  db 256 DUP(?)

AStack ENDS

DATA SEGMENT
  KEEP_CS DW 0 ;хранение сегмента
  KEEP_IP DW 0 ;хранение смещения прерывания

DATA ENDS

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

```
;
_____________________________________________________________________
_______
SUBR_INT PROC FAR
    push ax
    push dx

    ; подача звукового сигнала
    Sound:
    mov al, 10110110b ; управляющее слово таймера: канал 2, режим 3,
двоич.слово
    out 43h, al       ; вывод в регистр режима
    mov ax, 777       ; частота звука

    out 42h, al       ; младший байт счетчика
    mov al, ah
    out 42h, al       ; старший байт счетчика

    in al, 61h        ; порт PB
    mov ah, al
    or al, 3          ; устанавливаем биты 0-1
    out 61h, al
    sub cx, cx

    ; выключаем звук
    Sound_OFF:
    loop Sound_OFF
    mov al, 61h
    out 61h, al
```

```
  pop dx            ; восстанавливаем регистры
  pop ax

  mov al, 20h ; разрешение обработки прерываний
  out 20h, al ; более низкого уровня


  iret        ; конец прерывания
SUBR_INT ENDP

;
```

```
; ---------------------------------------
MAIN PROC FAR
  mov ax, DATA
  mov ds, ax

  mov ah, 35h          ; ф-ия получения вектора
  mov al, 60h          ; номер прерывания
  int 21h

  mov KEEP_IP, bx       ; запомнили смещение
  mov KEEP_CS, es       ; запомнили сегмент вектора прерывания

  push ds               ; сохранили ds
```

```
mov ax, seg SUBR_INT     ; сегмент процедуры в ax
mov dx, offset SUBR_INT  ; смещение процедуры
mov ds, ax

mov ah, 25h              ; функция установки вектора
mov al, 60h              ; номер вектора
int 21h                  ; изменение прерывания
pop ds                   ; восстанавливаем ds


int 60h                  ; наше прерывания
; ---------------------------------------


CLI
push ds

mov dx, KEEP_IP     ; восстановили смещение для прерывания
mov ax, KEEP_CS     ; восстановили сегмент прерывания
mov ds, ax

mov ah, 25h         ; функция установки вектора
mov al, 60h         ; номер нашего прерывания

int 21h             ; изменили прерывания
pop ds

STI
```

```
  mov ah, 4ch
  int 21h

MAIN ENDP
CODE ENDS
END MAIN
```