Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ЭВМ

Отчет по лабораторной работе №5

«Программирование клавиатуры»

Проверил: Выполнил:

к.т.н., доцент студент гр. 150503

Одинец Д.Н. Федорович И.И.

Минск 2023

1. Постановка задачи

Программируя клавиатуру помигать ее индикаторами. Алгоритм мигания произвольный. Условия реализации программы, необходимые для выполнения лабораторной работы:

1. Запись байтов команды должна выполняться только после проверки незанятости входного регистра контроллера клавиатуры. Проверка осуществляется считывание и анализом регистра состояния контроллера клавиатуры.
2. Для каждого байта команды необходимо считывать и анализировать код возврата. В случае считывания кода возврата, требующего повторить передачу байта, необходимо повторно, при необходимости – несколько раз, выполнить передачу байта. При этом повторная передача данных не исключает выполнения всех оставшихся условий.
3. Для определения момента получения кода возврата необходимо использовать аппаратное прерывания от клавиатуры.

Все коды возврата должны быть выведены на экран в шестнадцатеричной форме.

1. Алгоритм

#### Для вывода на экран скан-кодов или кодов возврата необходимо заменить обработчик прерывания 09h. При вызове данного обработчика выводится значение из порта 60h на экран. При управлении индикаторами значение из порта 60h (код возврата) необходимо анализировать на случай необходимости повторной передачи байтов команды.

#### Для управления индикаторами клавиатуры используется команда *EDh*. Второй байт этой команды содержит битовую маску для настройки индикаторов (бит 0 – состояние Scroll Lock, бит 1 – состояние Num Lock, бит 2 – состояние Caps Lock). В данной программе управление индикаторами реализовано в функции void indicator (int mask), где mask – битовая маска, определяющая состояние индикаторов.

1. Листинг программы

Далее приведен листинг программы, реализующей все поставленные задачи.

#include <stdio.h>

#include <stdlib.h>

#include <dos.h>

#define DELAY\_TIME 1000

#define SUCCESS 0xFA

#define KEYBOARD\_INTERRUPT 0x09

#define KEYBOARD\_LIGHTS\_CODE 0xED

#define EMPTY\_BUFFER 0x02 // 0000 0010

#define KEYBOARD\_STATE\_REGISTER 0x64

#define KEYBOARD\_DATA\_REGISTER 0x60

#define NONE 0x00

#define SCROLL\_LOCK 0x01

#define NUM\_LOCK 0x02

#define CAPS\_LOCK 0x04

typedef unsigned char byte;

int command\_succeeded = 0;

void interrupt (\*old\_handler)(void);

void interrupt new\_handler(void) {

byte scan\_code = inp(KEYBOARD\_DATA\_REGISTER);

printf("Scan code: %X\n", scan\_code);

// FA - succeed

// FE - error

command\_succeeded = (scan\_code == SUCCESS);

old\_handler();

}

void set\_mask(byte mask) {

int i = 0;

while (!command\_succeeded) {

// Wait until buffer is empty

while ((inp(KEYBOARD\_STATE\_REGISTER) & EMPTY\_BUFFER)); // 0000 0010

outp(KEYBOARD\_DATA\_REGISTER, KEYBOARD\_LIGHTS\_CODE);

// Wait until buffer is empty

while ((inp(KEYBOARD\_STATE\_REGISTER) & EMPTY\_BUFFER));

outp(KEYBOARD\_DATA\_REGISTER, mask);

// 0000 caps num scroll

if (++i == 3) {

fputs("Failed to set mask 3 times in a row\n", stderr);

setvect(KEYBOARD\_INTERRUPT, old\_handler);

exit(EXIT\_FAILURE);

}

}

command\_succeeded = 0;

}

int main(void) {

old\_handler = getvect(KEYBOARD\_INTERRUPT);

setvect(KEYBOARD\_INTERRUPT, new\_handler);

set\_mask(NUM\_LOCK);

delay(DELAY\_TIME);

set\_mask(CAPS\_LOCK);

delay(DELAY\_TIME);

set\_mask(SCROLL\_LOCK);

delay(DELAY\_TIME);

set\_mask(NONE);

delay(DELAY\_TIME);

set\_mask(SCROLL\_LOCK | NUM\_LOCK | CAPS\_LOCK);

delay(DELAY\_TIME);

set\_mask(NONE);

setvect(KEYBOARD\_INTERRUPT, old\_handler);

return EXIT\_SUCCESS;

}

1. Тестирование программ

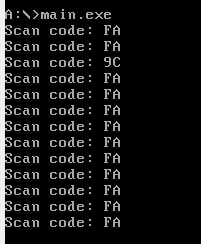


Рисунок 4.1 – Результат работы программы при запуске.

1. Заключение

В данной лабораторной работе были выполнены все поставленные задачи: написана программа, которая мигает индикаторами.

Программа компилировалась в Borland C и запускалась в DOS, который эмулировался с помощью MS-DOS в VirtualBox.