**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Организация ЭВМ и систем»**

Тема: Написание собственного прерывания.

|  |  |  |
| --- | --- | --- |
| Студент гр. 1383 |  | Депрейс А.С |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2022

## **Цель работы.**

## Изучение прерываний на языке ассемблера, разработка собственного обработчика прерываний.

**Задание.**

Вариант 07.

Заменить прерывание от системного таймера, которое генерируется автоматически операционной системой 18 раз в сек, на выполнение чтения и вывода на экран отсчета часов реального времени из памяти CMOS (в формате BCD).

**Выполнение работы**

В сегменте данных хранятся:

keep\_cs – место для хранения сегмента прерывания.

keep\_ip – место для храниния смещения прерывания.

Inter\_msg – строка хранящая текущее время.

Написана функция WriteMsg, которая выводит сообщение в консоль.

Написана функция Set\_time\_string, которая совершает деление значения в регистре al, остаток записывает в регистр ah. Затем к ah и al прибавляет код ‘0’.

Написан обработчик прерывания, который считывает время из "постоянных" (CMOS) часов реального времени, затем конвертирует данные в строку и выводит ее.

В основном теле программы при помощи функции 35 прерывания 21H сохраняется значение вектора прерывания в keep\_cs и keep\_ip, затем при помощи функции 25 прерывания 21H устанавливается вектор прерывания на наш реализованный обработчик прерываний. После этого программа ждет 10 секундс при помощи функции 86 прерывания 15h.

В конце программы восстанавливается старый вектор прерывания.

**Приложение А.**

**Исходный код программы.**

ASSUME CS:Code, SS:Stack, DS:Data

Stack SEGMENT STACK

DW 1024 DUP(0)

Stack ENDS

Data SEGMENT

inter\_msg DB ?,?,':',?,?,':',?,?,0DH,'$'

keep\_cs DW 0

keep\_ip DW 0

Data ENDS

Code SEGMENT

WriteMsg PROC NEAR

mov ah,9

int 21h

ret

WriteMsg ENDP

Set\_time\_string PROC NEAR

mov bl, 10h

mov ah, 0

div bl

add al, '0'

add ah, '0'

ret

Set\_time\_string ENDP

my\_int PROC FAR

push ax

push dx

push cx

read\_time:

mov ah, 02H

int 1AH

mov al, ch

call Set\_time\_string

mov inter\_msg, al

mov inter\_msg + 1, ah

mov al, cl

call Set\_time\_string

mov inter\_msg + 3, al

mov inter\_msg + 4, ah

mov al,dh

call Set\_time\_string

mov inter\_msg + 6, al

mov inter\_msg + 7, ah

mov dx, OFFSET inter\_msg

call WriteMsg

end\_inter:

pop cx

pop dx

pop ax

mov al, 20H

out 20H, al

iret

my\_int ENDP

main PROC FAR

push ds

xor ax, ax

push ax

mov ax, Data

mov ds, ax

mov ah, 35H ; функция получения вектора

mov al, 08H ; номер вектора

int 21H

mov keep\_ip, BX ; запоминание смещения

mov keep\_cs, ES ; и сегмента

push ds

mov dx, OFFSET my\_int ; смещение для процедуры в DX

mov ax, SEG my\_int ; сегмент процедуры

mov ds, ax ; помещаем в DS

mov ah, 25H ; функция установки вектора

mov al, 08H ; номер вектора

int 21H ; меняем прерывание

pop ds

xor ax, ax

mov ah, 86h

mov cx, 98h

mov dx, 9680h

int 15h

cli

push ds

mov dx, keep\_ip

mov ax, keep\_cs

mov ds, ax

mov ah, 25h

mov al, 08h

int 21h

pop ds

sti

ret

main ENDP

Code ENDS

END main