**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: Написание собственного прерывания**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент гр.1303 | |  | Иевлев Е.А. |
| Преподаватель |  |  | Ефремов М.А. |

Санкт-Петербург

2022

**Цель работы.**

Изучить, как работают прерывания. Написать собственное прерывание.

**Задание.**

* соответствии с 8 вариантом шифр задания – 2A, где 2 – 60h прерывание пользователя – должно генерироваться в программе; A – выполнить вывод сообщения на экран заданное число раз, после чего вставить фиксированную задержку и вывести сообщение о завершении обработчика.

**Выполнение работы.**

В сегменте стека Astack, выделяется 1Кбайт памяти.

* сегменте данных DATA содержится две переменных для хранения старого прерывания, содержавшегося по смещению 60h, – KEEP\_CS, KEEP\_IP. Также в этом сегменте содержится сообщение message, выводящееся во время работы прерывания и end\_message, выводящееся после завершения работы прерывания.
* сегменте кода сперва определяется процедура пользовательского прерывания SUBR\_INT. До входа в процедуру прерывания на стеке сохраняются значения регистров ax-dx. С помощью метки output строка из ds:dx выводится заданное в cx количество раз. Далее реализована задержка после вывода строк с помощью прерывания 1Ah. В регистре bx содержится требуемая задержка в тактах процессора, далее к ней прибавляется текущее время работы программы,

которое прерыванием 1Ah записывается в cx, dx. Далее в цикле происходит сравнение bx с текущем временем работы программы, и если время больше, то происходит выход из цикла. Далее при помощи прерывания 21h происходит вывод строки, сообщающей о завершении работы прерывания, хранящееся по адресу ds:offset end\_message. Перед выходом из прерывания восстанавливаются регистры из стека. Вызов прерывания происходит в головной процедуре Main. Для этого

2

сначала с помощью прерывания 21h получается прерывание, хранящееся по смещению 60h. В переменных, указанных в сегменте данных, сохраняется старое прерывание. Новое прерывание SUBR\_INT записывается по смещению 60h также с помощью прерывания 21h. Далее задаются значения регистров: в ds:dx должна лежать выводящаяся несколько раз строка, в cx – количество раз сколько нужно вывести строку, в bx – время задержки, в ds:offset – сообщение о завершении.

После вызова нового прерывания происходит восстановление старого прерывания и выход из программы.

**Тестирование.**

Работа программы с заданными условиями представлена на рисунке 1.

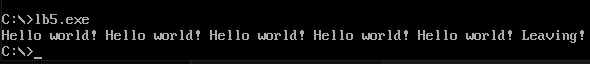
При вызове прерывания:

ds:dx message (message –«Hello world! $»)

cx = 05h (количество повторов выводимых сообщений = 5)

bx = 30h (время задержки в тактах процессора)

ds:offset end\_message (end\_message –«Leaving!$»)



Риc. 1

**Выводы.**

* + ходе выполнения лабораторной работы были изучены виды прерываний
* работа с ними. Была написана программа, реализующая собственное прерывание: вывод сообщений разное количество раз, с разным временным интервалом.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММ**

Название файла: *main.asm*

AStack SEGMENT STACK

dw 1024 dup(?)

AStack ENDS

DATA SEGMENT

KEEP\_CS dw 0

KEEP\_IP dw 0

message db 'Hello world! $'

end\_message db 'Leaving!$'

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:Astack

SUBR\_INT PROC FAR

push ax

push bx

push cx

push dx

mov ah, 09h

output:

int 21h

loop output

mov ah, 0

int 1Ah

add bx, dx

delay:

mov ah, 0

int 1Ah

cmp bx, dx

jg delay

mov dx, offset end\_message

mov ah, 09h

int 21h

pop ax

pop bx

pop cx

pop dx

mov al, 20h

out 20h, al

iret

SUBR\_INT ENDP

Main PROC FAR

push ds

sub ax, ax

push ax

mov ax, data

mov ds, ax

mov ah, 35h

mov al, 60h

int 21h

mov KEEP\_CS, es

mov KEEP\_IP, bx

push ds

mov dx, offset SUBR\_INT

mov ax, seg SUBR\_INT

mov ds, ax

mov ah, 25h

mov al, 60h

int 21h

pop ds

mov dx, offset message

mov cx, 05h

mov bx, 30h

int 60h

CLI

push ds

mov dx, KEEP\_IP

mov ax, KEEP\_CS

mov ds, ax

mov ah, 25h

mov al, 1ch

int 21h

pop ds

STI

mov ah, 4ch

int 21h

Main ENDP

CODE ENDS

END Main