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 _____ Валиев Р.Р.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Изучить как работает прерывание. Написать собственное прерывание.

Задание.

Вариант 8.

60h - прерывание пользователя - должно генерироваться в программе;

Выполнить вывод сообщения на экран заданное число раз,

после чего вставить фиксированную задержку и вывести сообщение о завершении обработчика.

Выполнение работы.

DATA – сегмент данных в программе. Он содержит: old_seg, old_ip – переменные для хранения старого прерывания, содержавшегося по смещению 60h, out_msg – сообщение которое будет выводиться прерыванием, end_msg – сообщение о завершении работы прерывания.

AStack – сегмент стека в программе.CODE – сегмент кода в программе.

Процедура пользовательского прерывания называется CUSTOM_INT. В ней сначала на стеке сохраняются значения регистров при входе в прерывание. Далее при помощи метки print_loop выводится строка, содержащаяся по адресу DS:DX количество раз заданное в CX.

Пауза после вывода строк реализуется при помощи прерывания 1Ah. При вызове прерывания в регистре bx должна содержаться требуемая задержка (в тиках процессора). К требуемой задержке прибавляется текущее время в программе, которое прерыванием 1Ah записывается в CX, DX (в CX – старшая часть значения). Далее в цикле происходит сравнение значения bx с текущим временем программы, если оно больше времени в bx, то производится выход из цикла.

Далее при помощи прерывания 21h производится вывод завершающего сообщения, хранящегося по адресу DS:offset end_msg.

После вывода завершающего сообщения производится восстановление регистров из стека и выход их прерывания.

Вызов прерывания производится в процедуре MAIN. Для этого сначала при помощи прерывания 21h происходит получение прерывания, хранящегося по смещению 60h. Старое прерывание сохраняется в переменных old_seg, old_ip.

Далее также при помощи прерывания 21h происходит запись по смещению 60h нового прерывания CUSTOM_INT.

Когда прерывание установлено, происходит заполнение регистров в соответствии с инструкцией по использованию прерывания: в ds:dx должна лежать выводящаяся несколько раз строка, в cx – количество раз сколько нужно вывести строку, в bx – время паузы (в тиках процессора), в ds:offset end_msg – сообщение о завершении.

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

Тестирование.

Результат работы программы представлен на Рисунке 1.

Рисунок 1 – результат работы программы.

Выводы.

В ходе выполнения лабораторной работы изучены виды прерываний и работа с ними. В соответствие с заданием создано собственное прерывание. Написана программа, выводящая строку заданное количество раз, после выставляющая задержку на заданное время и выводящая завершающее сообщение.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название программы: lab5.asm

```
DATA SEGMENT
   OLD_SEG DW 0
   OLD_IP DW 0
   OUT_MSG DB 'HELLO!$'
   END_MSG DB 'END!$'
DATA ENDS

ASTACK SEGMENT STACK
   DW 512 DUP(?)
ASTACK ENDS

CODE SEGMENT
   ASSUME CS:CODE, DS:DATA, SS:ASTACK

CUSTOM_INT PROC FAR
   ;STORING REGISTERS
   PUSH AX
   PUSH BX
   PUSH CX
   PUSH DX

   ; PRINT CX TIMES
   MOV AH, 9H

PRINT_LOOP:
   INT 21H
   LOOP PRINT_LOOP

   ; PAUSE
   MOV AH, 0
   INT 1AH
   ADD BX, DX

PAUSE:
   MOV AH, 0
   INT 1AH
```

```
    CMP BX, DX
    JG PAUSE

    ; PRINTING END MESSAGE
    MOV DX, OFFSET END_MSG
    MOV AH, 9H
    INT 21H

    ; RESTORING REGISTERS
    POP DX
    POP CX
    POP BX
    POP AX

    ; RETURN
    MOV AL, 20H
    OUT 20H, AL
    IRET
CUSTOM_INT ENDP

MAIN PROC FAR
    PUSH DS
    SUB AX, AX
    PUSH AX
    MOV AX, DATA
    MOV DS, AX

    ; STORING OLD INT
    MOV AX, 3560H
    INT 21H
    MOV OLD_SEG, ES
    MOV OLD_IP, BX

    ; SETTING CUSTOM INT
    PUSH DS
    MOV DX, OFFSET CUSTOM_INT
    MOV AX, SEG CUSTOM_INT
    MOV DS, AX
    MOV AX, 2560H
    INT 21H
    POP DS
```

```
    ; SETTING REGISTERS ACCORDING TO CUSTOM INT MANUAL
    MOV DX, OFFSET OUT_MSG
    MOV CX, 10H
    MOV BX, 36H
    INT 60H

    ; RESTORING OLD INT
    CLI
    PUSH DS
    MOV DX, OLD_IP
    MOV AX, OLD_SEG
    MOV DS, AX
    MOV AX, 251CH
    INT 21H
    POP DS
    STI

    RET

MAIN ENDP
CODE ENDS
    END MAIN
```