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. 0383 Самара Р. Д.

Преподаватель Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Написание собственного прерывания.

Задание.

Вариант 13 – 2f.

2 – 60h- прерывание пользователя, должно генерироваться в программе.

f - Реализовать вывод на экран заданного количества (3-5) сообщений, задержка между которыми возрастает в 2 раза, начиная от 1 сек.

Выполнение работы.

В начале программы определяется сегмент данных. В нём выделенные блоки памяти для хранения адреса старого смещения и для хранения сообщения, выводимого на экран. Для реализации прерывания была написана функция SUBR_INT. После сохранения регистров в стеке производится первая печать на экран. После печати выполняется проверка значения на верхушке стека. Если его значение равно нулю - программа завершается. В регистре al хранится значение текущей задержки, который удваивается, после чего происходит переход в блок start_proc. Для отсчёта секунд используется регистр bl, для хранения значения комера секунды регистр bh. С помощью прерывания int 21h с кодом 2ch происходит получение текущего номера секунды. Если он совпадает с сохраненным – значит секунда прошла, если нет – регистр bl уменьшается на единицу. После того, как значение регистра станет равным 0 – снова осуществляется вывод на экран. В блоке ending происходит восстановление регистров.

В конце программы вектор прерывания 60h восстанавливается с помощью переменных KEEP_CS и KEEP_IP.

Тестирование.

После запуска программы в консоль было выведено 5 сообщений с задержкой между ними в 1, 2, 4 и 8 секунд. Далее программа успешно

совершила выход из программы.

Выводы.

Был изучен механизм разработки собственных прерываний на языке Ассемблер.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл lab5.asm

```
DATA SEGMENT
        KEEP_CS DW 0 ; для хранения сегмента
        KEEP_IP DW 0 ; и смещения вектора прерывания
        HELLO   DB 'Hello World!',10,13,'$'
DATA ENDS

AStack    SEGMENT  STACK
          DW 512 DUP(?)    ;
AStack    ENDS


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

SUBR_INT PROC FAR
        jmp start_proc

        ss_int dw 0
         sp_int dw 0
        int_stack  DW 20 DUP(?)

start_proc:
    mov ss_int, ss
     mov sp_int, sp

    push dx
     push cx
     push bx
     push ax
     push ax

     mov al, 0
print_msg:
    MOV   AH,9
    MOV   DX, OFFSET HELLO
     int   21h  ; Вызов функции DOS по прерыванию
```

```
delay_count:
    pop cx
     dec cl
     jz ending
     push cx

     cmp al, 0   ;al - текущая задержка
     je one

     shl al, 1
     jmp start

one:
    inc al

start:
    mov bl, al  ;bl - отсчет секунд, bh - текущий номер секунды
     mov  ah, 2ch
     int  21h
     mov bh, dh

Delay:
     nop
     mov  ah, 2ch
     int  21h
     cmp dh, bh
     je Delay

    mov  bh, dh
     dec  bl
     jnz Delay
     jmp print_msg

ending:
     pop ax
     pop bx
     pop cx
     pop dx

     mov al,20h
     out 20h,al

     iret
```

```
SUBR_INT ENDP


Main      PROC  FAR

        push  DS        ;\  Сохранение адреса начала PSP в стеке
        sub   AX,AX     ; > для последующего восстановления по
        push  AX        ;/  команде ret, завершающей процедуру.
        mov   AX,DATA               ; Загрузка сегментного
        mov   DS,AX                 ; регистра данных.

        MOV AH, 35H ; функция получения вектора
        MOV AL, 60H ; номер вектора
        INT 21H ; возвращает текущее значение вектора прерывания
        MOV KEEP_IP, BX ; запоминание смещения
        MOV KEEP_CS, ES ; и сегмента вектора прерывания

        PUSH DS
        MOV DX, OFFSET SUBR_INT ; смещение для процедуры в DX
        MOV AX, SEG SUBR_INT ; 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

          mov al, 5
        int 60H; вызов измененного прерывания

        CLI
        PUSH DS
        MOV DX, KEEP_IP
        MOV AX, KEEP_CS
        MOV DS, AX
        MOV AH, 25H
        MOV AL, 60H
        INT 21H ; восстанавливаем старый вектор прерывания
        POP DS
        STI

        RET
Main      ENDP
```

```
CODE        ENDS
            END Main
```