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: «Разработка собственного прерывания»

Студент гр. 1381 _____ Возмитель В.Е.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022 г.

Цель работы

Написать программу обработки прерывания.

Задание/краткие сведения

Разработка собственного прерывания.

Прерывание — это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств (обработка сигнала таймера, нажатия клавиши и т. д.).
Когда возникает прерывание, процессор прекращает выполнение текущей программы (если ее приоритет ниже) и запоминает в стеке вместе с регистром флагов адрес возврата (CS:IP) - места, с которого будет продолжена прерванная программа. Затем в CS:IP загружается адрес программы обработки прерывания и ей передается управление.
Адреса 256 программ обработки прерываний, так называемые векторы прерывания, имеют длину по 4 байта (в первых двух хранится значение IP , во-вторых - CS) и хранятся в младших 1024 байтах памяти.
Программа обработки прерывания должна заканчиваться инструкцией IRET (возврат из прерывания), по которой из стека восстанавливается адрес возврата и регистр флагов.

Для хранения сегмента и смещения прерывания были созданы переменные KEEP_CS и KEEP_IP (обе по 2 байта). Функция 35H прерывания 21H возвращает текущее значение вектора прерывания (в данном случае 08H), и его смещение и сегмент заносятся в вышеупомянутые переменные для их последующего восстановления.

После этого с помощью функции 25H прерывания 21H устанавливается свое прерывание (процедура SUBR_INT) посредством помещения смещения в DX, а сегмента в DS. Далее с помощью функции 25H прерывания 21H восстанавливается старое прерывание.

Реализация собственного прерывания: для вывода десятичных чисел на экран была написана функция OutPut, которая выводит десятичные числа, находящиеся в AX (см. комментарии в коде).

Для получения отчета системных часов в тиках используется функция 00H прерывания 1AH, которая помещает в регистры CX и DX время в тиках (CX – старшее значение). Далее эти значения помещаются в AX, и для каждого из них вызывается функция вывода на экран.

В конце программы восстанавливается старый вектор прерывания:

```
CLI
PUSH DS
MOV  DX, KEEP_IP
MOV  AX, KEEP_CS
MOV  DS, AX
MOV  AH, 25H
MOV  AL, 1CH
INT  21H ;восстанавливаем вектор
POP  DS
STI
```

Команды CLI и STI служат для установки или сброса флага прерываний, что позволяет включать или отключать реакцию на внешние прерывания. Команда CLI сбрасывает флаг IF в значение 0, что запрещает прерывания. Команда STI устанавливает флаг IF в значение 1, что разрешает прерывания.

Вариант работа №1d

1 - 08h - прерывание от системного таймера - генерируется автоматически операционной системой 18 раз в сек;

D - Выполнить чтение и вывод на экран отсчета системных часов (в тиках, где 1 тик = 55 мсек).

Вывод

В ходе выполнения данной лабораторной работы были изучены способы создания, обработки и вызова прерываний в языке Assembler.

Приложение А. Код программы LB5.ASM

```
ASSUME CS:CODE, DS:DATA, SS:STACK

STACK    SEGMENT  STACK
          DW 1024 DUP(?)
STACK    ENDS

DATA SEGMENT
        KEEP_CS DW 0 ;буфер для хранения сегмента
        KEEP_IP DW 0 ;смещение вектора прерывания
DATA ENDS

CODE SEGMENT

OutPut PROC
     push dx
    push cx

    xor cx, cx ;cx - кол-во цифр
    mov bx, 10 ;основание сс

S1:
    xor dx, dx
    div bx ;делим число на основание сс и сохраняем остаток в стеке
    push dx
    inc cx ;увеличиваем счетчик

    test ax, ax ;сравнение с 0
    jnz S1

    mov ah, 2 ;вывод на дисплей

S2:
    pop dx
    add dl, '0'
    int 21h
    loop S2 ;Команда цикла(пока cx не 0)

    pop cx
    pop dx
    ret
OutPut endp


SUBR_INT PROC FAR

     JMP start

     save_sp DW 0000h
     save_ss DW 0000h
     INT_STACK DB 40 DUP(0)

start:
    mov save_sp, sp
     mov save_ss, ss
     mov sp, SEG INT_STACK
     mov ss, sp
```

```
      mov sp, offset start
      push ax
      push cx
      push dx ;сохранение изменяемых регистров

      mov ah, 00h ;читать часы(счетчик тиков)
      int 1AH

      mov ax, cx
      call OutPut ;вызов процедуры OutPut
      mov ax, dx
      call OutPut ;вызов процедуры OutPut

      pop dx
      pop cx
      pop ax ;возвращаем регистры в исходное состояние
      mov ss, save_ss
      mov sp, save_sp

      mov al, 20h
      OUT 20h, al

      iret ;возврат из прерывания

SUBR_INT ENDP


Main  PROC  FAR
      push DS ;запоминаем адрес psp
      sub AX,AX
      push AX
      mov AX,DATA ;получаем адрес DATA SEGMENT
      mov DS,AX ;записываем его в ds


;Запоминание текущего вектора прерывания
      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

;Установка 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, 08H ;номер вектора
      INT 21H ;меняем прерывание
      POP DS

      int 08H ;вызов прерывания

;Восстановление изначального вектора прерывания
      CLI
      PUSH DS
      MOV DX, KEEP_IP ;смещение для процедуры в DX
```

```
        MOV AX, KEEP_CS ;сегмент процедуры
        MOV DS, AX ;помещаем в DS
        MOV AH, 25H ;функция установки вектора
        MOV AL, 08H ;номер вектора
        INT 21H ;восстанавливаем вектор
        POP DS
        STI

        MOV AH, 4Ch
        INT 21h
Main    ENDP
CODE ENDS
        END Main
```

Приложение Б. Тестирование.

Табл. №1: Тестирование программы lab5.asm

Вход	Выход	Комментарий
-	1310048	Корректно.
-	1310062	Почти мгновенный запуск после первого теста. ~1 сек. Корректно.