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.0382 _____ Диденко Д.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Изучить способ написания собственного прерывания.

Задание.

Вариант 4.

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

Выполнить чтение и вывод на экран отсчета системных часов.

Основные теоретические положения.

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.

Выполнение работы.

Программа обработки прерывания - это отдельная процедура, имеющая структуру:

SUBR_INT PROC FAR

```
        PUSH AX   ; сохранение изменяемых регистров

        <действия по обработке прерывания>

        POP  AX   ; восстановление регистров
        MOV  AL, 20H
        OUT  20H,AL
        IRET
SUBR_INT  ENDP
```

В действиях по обработке прерывания вызывается процедура OutInt, которая посимвольно выводит в терминал число – количество тиков на данный момент.

Две последние строки перед IRET необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное.

Замечание: в лабораторной работе действиями по обработке прерывания может быть вывод на экран некоторого текста, вставка задержки в вывод сообщений, включение звукового сигнала и т.п.

Программа, использующая новые программы обработки прерываний, при своем завершении должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H возвращает текущее значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. В этом случае программа должна содержать следующие инструкции:

```
; -- в сегменте данных
    KEEP_CS DW 0 ; для хранения сегмента
    KEEP_IP DW 0 ; и смещения прерывания

; -- в начале программы
    MOV  AH, 35H   ; функция получения вектора
```

```
MOV  AL, 1CH   ; номер вектора
INT  21H
MOV  KEEP_IP, BX  ; запоминание смещения
MOV  KEEP_CS, ES  ; и сегмента
```

Для задания адреса собственного прерывания с заданным номером в таблицу векторов прерываний используется функция 25H прерывания 21H, которая устанавливает вектор прерывания на указанный адрес нового обработчика.

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

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

Результаты тестирования программы lab4.cpp представлены в табл. 1.

Таблица 1 – Тестирование программы lab4.cpp.

№ п/п	Вывод	Результат
1.	1942604	Программа работает верно
2.	1943043	Программа работает верно

Выводы.

Изучен способ написания собственного прерывания.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab5.asm

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

STACK    SEGMENT  STACK
          DW 1024 DUP(?)
STACK    ENDS

DATA SEGMENT
        KEEP_CS DW 0 ; для хранения сегмента
        KEEP_IP DW 0 ; и смещения вектора прерывания
           NUM DW 0
           MESSAGE DB 2 DUP(?)
DATA ENDS

CODE SEGMENT

OutInt PROC
      push DX
      push CX

    xor     cx, cx ; cx - количество цифр
    mov     bx, 10 ; основание сс. 10 для десятеричной и т.п.
oi2:
    xor     dx,dx
    div     bx ; делим число на основание сс и сохраняем остаток в стеке
    push    dx
    inc     cx; увеличиваем количество цифр в cx

    test    ax, ax ; проверка на 0
    jnz     oi2
; Вывод
    mov     ah, 02h
oi3:
```

```
    pop     dx
    add     dl, '0' ; перевод цифры в символ
    int     21h
; Повторим ровно столько раз, сколько цифр насчитали.
    loop    oi3 ; пока cx не 0 выполняется переход

      POP CX
      POP DX
    ret

OutInt endp


SUBR_INT PROC FAR
        JMP start_proc
           save_SP DW 0000h
           save_SS DW 0000h
           INT_STACK DB 40 DUP(0)
start_proc:
    MOV save_SP, SP
      MOV save_SS, SS
      MOV SP, SEG INT_STACK
      MOV SS, SP
      MOV SP, offset start_proc
      PUSH AX     ; сохранение изменяемых регистров
      PUSH CX
      PUSH DX

      mov AH, 00H
      int 1AH

      mov AX, CX
      call OutInt
      mov AX, DX
      call OutInt

      POP  DX
      POP  CX
```

```
      POP  AX    ; восстановление регистров
      MOV  SS, save_SS
      MOV  SP, save_SP
      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


      ; 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; на всякий вывод в консоль отдельно от отладчика

        ; Восстановление изначального вектора прерывания (можно закомментить)
        CLI
        PUSH DS
        MOV  DX, KEEP_IP
        MOV  AX, KEEP_CS
        MOV  DS, AX
        MOV  AH, 25H
        MOV  AL, 08H
        INT  21H           ; восстанавливаем вектор
        POP  DS
        STI

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