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
Вариант 14

Студент гр.0382 _____ Литягин С.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Написать собственное прерывание, согласно заданию.

Задание.

Прерывание - это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств (обработка сигнала таймера, нажатия клавиши и т.д.). Написать собственное прерывание согласно варианту

Вариант №14:

2g

2 – 60h – прерывание пользователя – должно генерироваться в программе;

g – Выполнить ввод и печать заданного количества символов, после чего вывести сообщение о завершении обработчика

Выполнение работы:

В сегменте данных DATA содержаться переменные: KEEP_CS, KEEP_IP для хранения сегмента и смещения старого прерывания соответственно, message для записи строки, что вводят (далее строка ввода-вывода), message2 для вывода строки о завершении.

Процедура пользовательского прерывания InOut. Сначала в ней мы сохраняем все изменяемые регистры в стеке. Помещаем в регистр AX константное значение siz, помещаем значение регистра AX в CX (далее следует цикл по вводу определенного кол-ва символов, а именно siz раз). Получаем смещение на начало строки для ввода-вывода, откуда начинать ввод, в регистр DI. Начинаем цикл lp. В нем мы используем функцию 01h прерывания 21h для получения очередного введенного символа (помещается в AL). Заносим в [DI] этот символ. Увеличиваем DI, чтобы на следующем шаге записывать символ в свободное место. По завершении цикла, в AH помещается значение функции вывода строки (09h). В DX помещается смещение на начало строки ввода-вывода. Вызываем 21h прерывание. В DX помещается смещение на начало

строки о завершении. Вызываем 21h прерывание. Строки выведены в консоль. Восстанавливаем регистры из цикла. Выходим из прерывания.

В процедуре Main запоминаем смещение и сегмент текущего 60h прерывания в KEEP_IP, KEEP_CS с помощью функции 35h прерывания 21h. Используя же функцию 25h прерывания 21, устанавливаем вектор прерывания 60h на наше прерывание InOut. Затем происходит его вызов. Когда его работа будет завершена – восстанавливаем старый вектор прерывания.

Исходный код программы см. в приложении А.

Тестирование:

Для проверки работоспособности были проведены тесты, результаты представлены в таблице 1.

Таблица 1 – Тестирование и результаты

Номер теста	Входные данные	Выходные данные	Верность результата
1	(ввод 10 символов) ffsaddgsfd	C:\>lb5.exe ffsaddgsfd ffsaddgsfd End!	верно
2	(ввод 15 символов) Fake id, please	C:\>lb5.exe Fake id, please Fake id, please End!	верно
3	(ввод 1 символа) 1	C:\>lb5.exe 1 1 End!	верно

Выводы.

В ходе работы были изучены прерывания. Также было написано собственное прерывание по вводу-выводу строки и строки о завершении.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb5.asm

```
siz equ 1 ; кол-во символов в строке для ввода

AStack SEGMENT STACK
 DW 12 DUP(?)
AStack ENDS

DATA SEGMENT
 KEEP_CS DW 0 ; для хранения сегмента
 KEEP_IP DW 0 ; и смещения прерывания
 message DB 0Dh, 0Ah,siz dup("$"), '$' ; строка для ввода-вывода
 message2 DB 0Dh, 0Ah,'End!', '$' ; строка о завершении
DATA ENDS

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

InOut PROC FAR
      push AX ; сохранение изменяемых регистров
      push CX
      push DX

      mov ax, siz
      mov cx, ax ; помещаем число вводов символов для цикла
      mov di, offset message ; получаем смещение на начало нашего
сообщения ввода-вывода
      add di, 2 ; смещаемся на 2 из-за двух первых символов, который
делают красиво
lp:
      mov ah, 01h ; функция ввода с клавиатуры (символ => al)
      int 21h ; вызываем прерывание
      mov [di], al ; помещаем символ в строку
      inc di ; увеличиваем смещение на следующий символ, доступный к
записи
      loop lp ; повторяем процесс, пока не введут нужное число символов

      mov ah, 09h ; функция вывода строки
      mov dx, offset message ; указываем смещение строки
      int 21h ; вывели
      mov dx, offset message2
      int 21h ; вывели

      pop DX ; счастливые восстановлеваем регистры
      pop CX
        pop AX
        mov AL, 20h ; для разрешения обрабоки прерываний
        out 20h,AL ; с более низкими уровнями, чем только что
обработанное
        iret
InOut ENDP
```

```
Main PROC FAR
      push DS
      sub AX,AX
      push AX
      mov AX, DATA
      mov DS, AX

      mov AH,35h ; возвращение текущего значения вектора прерывания
      mov AL,60h ; номер вектора
      int 21h
      mov KEEP_IP, BX ; запоминание смещения
      mov KEEP_CS, ES ; запоминание сегмента

      push DS
      mov DX, offset InOut ; смещение для процедуры
      mov AX, seg InOut ; сегмент процедуры
      mov DS, AX
      mov AH, 25h ; функция установки вектора
      mov AL, 60h ; номер вектора
      int 21h ; устанавливаем вектор прерывания на указанный адрес нового
обработчика
      pop DS

      int 60h ; вызываем прерывание пользователя

      CLI
      push DS
      mov DX, KEEP_IP
      mov AX, KEEP_CS
      mov DS, AX
      mov AH, 25h
      mov AL, 60h
      int 21h
      pop DS
      STI

      ret
Main ENDP
CODE ENDS
      END Main
```

Название файла: lb5.lst

```
Microsoft (R) Macro Assembler Version 5.10                  12/2/21
06:14:00
                                                             Page     1-1


      = 0001                    siz equ 1 ; PєPsP»-PIPs CЃPёPjPIPsP»PsPI
PI CЃC
                          ,CЂPsPєPµ PґP»CЏ PIPIPsPґP°

      0000                AStack SEGMENT STACK
      0000  000C[               DW 12 DUP(?)
             ????
                        ]

      0018                AStack ENDS

      0000                DATA SEGMENT
      0000  0000                KEEP_CS  DW  0  ;  PґP»CЏ  C…CЂP°PSPµPSPёCЏ
CЃPµPiP
                          jPµPSC,P°
      0002  0000                KEEP_IP   DW   0   ;   Pё   CЃPjPµC‰PµPSPёCЏ
PїCЂPµCЂC‹ P
                          IP°PSPёCЏ
      0004  0D 0A               message DB 0Dh, 0Ah,siz dup("$"), '$' ;
CЃC,CЂ
                          PsPєP° PґP»CЏ PIPIPsPґP°-PIC‹PIPsPґP°
            0001[
             24
                        ]
            24
      0008  0D 0A 45 6E 64 21    message2  DB  0Dh,  0Ah,'End!',  '$'  ;
CЃC,CЂPsPєP
                          ° Ps P·P°PIPµCЂC€PµPSPёPё
            24
      000F                DATA ENDS

      0000                CODE SEGMENT
                           ASSUME CS:CODE, DS:DATA, SS:AStack

      0000                InOut PROC FAR
      0000  50                  push AX ; CЃPsC…CЂP°PSPµPSPёPµ PёP·PjPµ
                          PSCЏPµPjC‹C… CЂPµPiPёCЃC,CЂPsPI
      0001  51                  push CX
      0002  52                  push DX

      0003  B8 0001                  mov ax, siz
      0006  8B C8                    mov   cx,   ax   ;   PїPsPjPµC‰P°PµPj
C‡PёCЃP»P
                          s    PIPIPsPґPsPI    CЃPёPjPIPsP»PsPI    PґP»CЏ
C†PёPєP»
                          P°
      0008  BF 0004 R           mov di, offset message ; PїPsP»CґC‡P°Pµ
                          Pj    CЃPjPµC‰PµPSPёPµ    PSP°    PSP°C‡P°P»Ps
PSP°C€PµP
```

```
                        iPs CЃPsPsP±C‰PµPSPёCЏ PIPIPsPґP°-PIC‹ PIPsPґP°
      000B  83 C7 02                add di, 2 ; CЃPjPµC‰P°PµPjCЃCЏ PSP°
2 P
                        ёP·-P·P°         PґPICЃC…         PїPµCЂPIC‹C…
CЃPёPjPIPsP»PsPI
                        , PєPsC,PsCЂC‹P№ PґPµP»P°CЂC, PєCЂP°CЃPёPIPs
      000E              lp:
      000E  B4 01                   mov   ah,   01h   ;   C„CЃPSPєC†PёCЏ
PIPIPsPґP°
                         CЃ PєP»P°PIPёP°C,CЃCЂC‹ (CЃPёPjPIPsP» => al)
      0010  CD 21                   int    21h    ;    PIC‹P·C‹PIP°PµPj
PїCЂPµCЂC‹PI
                        P°PSPёPµ
      0012  88 05                   mov  [di],  al   ;  PїPsPjPµC‰P°PµPj
CЃPёPjP
                        IPsP» PI CЃC,CЂPsPєCЃ
      0014  47                 inc di ; CЃPIPµP»PёC‡PёPIP°PµPj CЃPjPµC
                       ‰PµPSPёPµ PSP° CЃP»PµPґCЃCЂC‰PёP№ CЃPёPjPIPsP»,
Microsoft (R) Macro Assembler Version 5.10                    12/2/21
06:14:00
                                                              Page     1-2


                         PґPsCЃC,CЃPїPSC‹P№ Pє P·P°PїPёCЃPё
      0015  E2 F7                   loop   lp   ;   PїPsPIC,PsCЂCЏPµPj
PїCЂPsC†Pµ
                        CЃCЃ, PїPsPєP° PSPµ PIPIPµPґCЃC, PSCЃP¶PSPsPµ
C
                        ‡PёCЃP»Ps CЃPёPjPIPsP»PsPI

      0017  B4 09                   mov   ah,   09h   ;   C„CЃPSPєC†PёCЏ
PIC‹PIPsPґ
                        P° CЃC,CЂPsPєPё
      0019  BA 0004 R            mov dx, offset message ; CЃPєP°P·C‹PIP°
                        PµPj CЃPjPµC‰PµPSPёPµ CЃC,CЂPsPєPё
      001C  CD 21                   int 21h ; PIC‹PIPµP»Pё
      001E  BA 0008 R            mov dx, offset message2
      0021  CD 21                   int 21h ; PIC‹PIPµP»Pё

      0023  5A                   pop DX ; CЃC‡P°CЃC,P»PёPIC‹Pµ PIPsCЃCЃC
                        ,P°PSPsPIP»PµPIP°PµPj CЂPµPiPёCЃC,CЂC‹
      0024  59                   pop CX
      0025  58                      pop AX
      0026  B0 20                      mov    AL,    20h    ;    PґP»CЏ
CЂP°P·CЂPµC€PµPSPё
                        CЏ PsP±CЂP°P±PsPєPё PїCЂPµCЂC‹PIP°PSPёP№
      0028  E6 20                   out   20h,AL   ;   CЃ   P±PsP»PµPµ
PSPёP·PєPёPj
                        Pё   CЃCЂPsPIPSCЏPjPё,   C‡PµPj   C,PsP»CЊPєPs
C‡C,Ps
                         PsP±CЂP°P±PsC,P°PSPSPsPµ
      002A  CF                      iret
      002B              InOut ENDP


      002B              Main PROC FAR
```

```
      002B  1E                  push DS
      002C  2B C0                     sub AX,AX
      002E  50                  push AX
      002F  B8 ---- R           mov AX, DATA
      0032  8E D8                     mov DS, AX

      0034  B4 35                     mov AH,35h ; PIPsP·PICЂP°C‰PµPSPёPµ
C,P
                        µPєCґC‰PµPiPs P·PSP°C‡PµPSPёCЦ PIPµPєC,PsCЂP°
P
                        ïCЂPµCЂC‹PIP°PSPёCЦ
      0036  B0 60                     mov     AL,60h     ;     PSPsPjPµCЂ
PIPµPєC,PsCЂP°
      0038  CD 21                     int 21h
      003A  89 1E 0002 R              mov       KEEP_IP,       BX       ;
P·P°PïPsPjPёPSP°PSPёP
                        µ CЃPjPµC‰PµPSPёCЦ
      003E  8C 06 0000 R              mov       KEEP_CS,       ES       ;
P·P°PïPsPjPёPSP°PSPёP
                        µ CЃPµPiPjPµPSC,P°

      0042  1E                  push DS
      0043  BA 0000 R                 mov DX, offset InOut ; CЃPjPµC‰PµPSPёPµ
                         PґP»CЦ PïCЂPsC†PµPґCґCЂC‹
      0046  B8 ---- R                 mov AX, seg InOut ; CЃPµPiPjPµPSC, PïCЂ
                        PsC†PµPґCґCЂC‹
      0049  8E D8                     mov DS, AX
      004B  B4 25                     mov   AH,   25h   ;   C„CґPSPєC†PёCЦ
CґCЃC,P°PS
                        PsPIPєPё PIPµPєC,PsCЂP°
      004D  B0 60                     mov    AL,    60h    ;    PSPsPjPµCЂ
PIPµPєC,PsCЂP°
      004F  CD 21                     int               21h              ;
CґCЃC,P°PSP°PIP»PёPIP°PµPj PI
                        PµPєC,PsCЂ PïCЂPµCЂC‹PIP°PSPёCЦ PSP° CґPєP°P·P°
□Microsoft (R) Macro Assembler Version 5.10                       12/2/21
06:14:00
                                                                Page     1-3


                        PSPSC‹P№        P°PґCЂPµCЃ        PSPsPIPsPiPs
PsP±CЂP°P±PsC,
                        C‡PёPєP°
      0051  1F                 pop DS

      0052  CD 60                     int    60h    ;    PIC‹P·C‹PIP°PµPj
PïCЂPµCЂC‹PI
                        P°PSPёPµ PïPsP»CЂP·PsPIP°C,PµP»CЦ

      0054  FA                 CLI
      0055  1E                 push DS
      0056  8B 16 0002 R              mov DX, KEEP_IP
      005A  A1 0000 R          mov AX, KEEP_CS
      005D  8E D8                     mov DS, AX
      005F  B4 25                     mov AH, 25h
      0061  B0 60                     mov AL, 60h
```

```
0063  CD 21                     int 21h
0065  1F                   pop DS
0066  FB                   STI

0067  CB                   ret
0068                 Main ENDP
0068                 CODE ENDS
                          END Main
```


Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	0068	PARA	NONE	
DATA	000F	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr
INOUT	F PROC	0000	CODE Length = 002B
KEEP_CS	L WORD	0000	DATA
KEEP_IP	L WORD	0002	DATA
LP	L NEAR	000E	CODE
MAIN	F PROC	002B	CODE Length = 003D
MESSAGE	L BYTE	0004	DATA
MESSAGE2	L BYTE	0008	DATA
SIZ	NUMBER	0001	
@CPU	TEXT	0101h	
@FILENAME	TEXT	lb5	
@VERSION	TEXT	510	

```
     86 Source  Lines
     86 Total   Lines
     16 Symbols

  48030 + 459230 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```