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: Написание собственного прерывания
Вариант 8

Студентка гр.1381 _____ Рымарь М.И.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Изучить, как работают прерывания. Написать собственное прерывание.

Задание.

В соответствии с 8 вариантом шифр задания – 2a, где

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

a – выполнить вывод сообщения на экран заданное число раз, после чего вставить фиксированную задержку и вывести сообщение о завершении обработчика.

Выполнение работы.

В сегменте данных DATA содержится две переменных для хранения старого прерывания, содержавшегося по смещению 60h, – seg_prev, ip_prev. Также в этом сегменте содержится message_output – сообщение, которое будет выводиться во время работы прерывания, message_final – сообщение, которое будет выведено после завершения работы прерывания.

В сегменте стека Astack, как и требуется по заданию, выделяется 1Кбайт памяти, то есть dw 512.

В сегменте кода сначала определяем процедуру пользовательского прерывания INT_CUSTOM. Сначала на стеке сохраняются значения регистров до входа в прерывание. С помощью метки loop_output строка из ds:dx выводится заданное в cx количество раз. Далее реализована задержка после вывода строк с помощью прерывания 1Ah. В регистре bx содержится требуемая задержка в тактах процессора, далее к ней прибавляется текущее время работы программы, которое прерыванием 1Ah записывается в cx, dx. Далее в цикле происходит сравнение bx с текущем временем работы программы, если оно больше, то происходит выход из цикла. И при помощи прерывания 21h происходит вывод строки, сообщающей о завершении работы прерывания. Оно хранится по адресу ds:offset message_final. Далее перед выходом из прерывания восстанавливаются регистры из стека. Вызов прерывания происходит в процедуре Main. Для этого

сначала с помощью прерывания 21h получается прерывание, хранящееся по смещению 60h. В переменных, указанных в сегменте данных, сохраняется старое прерывание. Новое прерывание INT_CUSTOM записывается по смещению 60h также с помощью прерывания 21h. Далее задаются значения регистров: в ds:dx должна лежать выводящаяся несколько раз строка, в cx – количество раз сколько нужно вывести строку, в bx – время задержки, в ds:offset – сообщение о завершении.

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

Тестирование.

Работа программы с заданными условиями представлена на рисунке 1.

При вызове прерывания заданы следующие регистры:

ds:dx message_output (где message_output – это «Message output. »)

cx = 05h (количество повторов выводимых сообщений – 5, переведённое в 16-ричную систему счисления)

bx = 72h (время задержки в тактах процессора, в секундах приблизительно равное 6)

ds:offset message_final (где message_final – это «FINAL MESSAGE.»)

```
C:\>LAB5.EXE
Message output. Message output. Message output. Message output. Message output.
FINAL MESSAGE.
C:\>
```

Рисунок 1 – Работа программы

Выводы.

В ходе выполнения лабораторной работы были изучены виды прерываний и работа с ними. В соответствии с заданием было создано собственное прерывание. Была написана программа, выводящая одно сообщение определённое количество раз, а другое – один раз с определённой задержкой.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: *lab5.asm*

```
DATA SEGMENT
    seg_prev dw 0
    ip_prev dw 0
    message_output db 'Message output. $'
    message_final db 'FINAL MESSAGE.$'
DATA ENDS

AStack SEGMENT STACK
    dw 512 dup(?)
AStack ENDS

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

INT_CUSTOM PROC FAR
    push ax  ;  registers storing
    push bx
    push cx  ; numbers of prints in cx
    push dx

    mov ah, 9h  ; print cx times

loop_output:
    int 21h
    loop loop_output

    mov ah, 0   ; delay
    int 1Ah
    add bx, dx

delay:
    mov ah, 0
    int 1Ah
    cmp bx, dx
    jg delay

    mov dx, offset message_final  ; final message output
    mov ah, 9h
    int 21h

    pop dx   ; restoring
    pop cx
    pop bx
    pop ax

    mov al, 20h
    out 20h, al
    iret
INT_CUSTOM ENDP
```

```
Main PROC FAR
    push ds
    sub ax, ax
    push ax
    mov ax, data
    mov ds, ax

    mov ax, 3560h  ; previous interruption storing
    int 21h
    mov seg_prev, es
    mov ip_prev, bx

    push ds    ; custom interruption setting
    mov dx, offset int_custom
    mov ax, seg int_custom
    mov ds, ax
    mov ax, 2560h
    int 21h
    pop ds

    mov dx, offset message_output    ; setting registers using custom
interruption manual
    mov cx, 05h   ; number of messages
    mov bx, 72h  ; delay in ticks of process /seconds/
    int 60h

    CLI   ; previous interruption restoring
    push ds
    mov dx, ip_prev
    mov ax, seg_prev
    mov ds, ax
    mov ax, 251ch
    int 21h
    pop ds
    STI

    ret

main ENDP

CODE ENDS
END Main
```

Название файла: *lab5.lst*

```
Microsoft (R) Macro Assembler Version 5.10                11/5/22 01:26:17
                                                             Page     1-1


 0000                 DATA SEGMENT
 0000  0000                   seg_prev dw 0
 0002  0000                   ip_prev dw 0
 0004  4D 65 73 73 61 67      message_output db 'Message output. $'
       65 20 6F 75 74 70
       75 74 2E 20 24
 0015  46 49 4E 41 4C 20      message_final db 'FINAL MESSAGE.$'
       4D 45 53 53 41 47
```

```
      45 2E 24
0024                DATA ENDS

0000                AStack SEGMENT STACK
0000  0200[                  dw 512 dup(?)
       ????
                ]

0400                AStack ENDS

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

0000                INT_CUSTOM PROC FAR
0000  50                  push ax  ;  registers storing
0001  53                  push bx
0002  51                  push cx  ; numbers of prints in cx
0003  52                  push dx

0004  B4 09                    mov ah, 9h  ; print cx times

0006                loop_output:
0006  CD 21                    int 21h
0008  E2 FC                    loop loop_output

000A  B4 00                    mov ah, 0   ; delay
000C  CD 1A                    int 1Ah
000E  03 DA                    add bx, dx

0010                delay:
0010  B4 00                    mov ah, 0
0012  CD 1A                    int 1Ah
0014  3B DA                    cmp bx, dx
0016  7F F8                    jg delay

0018  BA 0015 R           mov dx, offset message_final  ; final messa
                        ge output
001B  B4 09                    mov ah, 9h
001D  CD 21                    int 21h

001F  5A                  pop dx   ; restoring
0020  59                  pop cx
0021  5B                  pop bx
0022  58                  pop ax

0023  B0 20                    mov al, 20h
```



```
0025  E6 20                    out 20h, al
0027  CF                  iret
0028                INT_CUSTOM ENDP

0028                Main PROC FAR
0028  1E                  push ds
```

```
 0029  2B C0                  sub ax, ax
 002B  50               push ax
 002C  B8 ---- R        mov ax, data
 002F  8E D8                 mov ds, ax

 0031  B8 3560               mov ax, 3560h  ; previous interruption stor
                    ing
 0034  CD 21                 int 21h
 0036  8C 06 0000 R          mov seg_prev, es
 003A  89 1E 0002 R          mov ip_prev, bx

 003E  1E               push ds    ; custom interruption setting
 003F  BA 0000 R        mov dx, offset int_custom
 0042  B8 ---- R        mov ax, seg int_custom
 0045  8E D8                 mov ds, ax
 0047  B8 2560               mov ax, 2560h
 004A  CD 21                 int 21h
 004C  1F               pop ds

 004D  BA 0004 R            mov dx, offset message_output   ; setting r
                    egisters using custom interruption manual
 0050  B9 0005               mov cx, 05h   ; number of messages
 0053  BB 0072               mov bx, 72h  ; delay in ticks of process /s
                    econds/
 0056  CD 60                 int 60h

 0058  FA               CLI   ; previous interruption restoring
 0059  1E               push ds
 005A  8B 16 0002 R          mov dx, ip_prev
 005E  A1 0000 R        mov ax, seg_prev
 0061  8E D8                 mov ds, ax
 0063  B8 251C               mov ax, 251ch
 0066  CD 21                 int 21h
 0068  1F               pop ds
 0069  FB               STI

 006A  CB               ret

 006B                   main ENDP

 006B                   CODE ENDS
                        END Main
```

Microsoft (R) Macro Assembler Version 5.10 11/5/22 01:26:17
Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine Class
ASTACK	0400	PARA	STACK
CODE	006B	PARA	NONE
DATA	0024	PARA	NONE

Symbols:

```
                N a m e         Type  Value     Attr

INT_CUSTOM . . . . . .          F PROC    0000  CODE  Length = 0028
IP_PREV  . . . . . .            L WORD    0002  DATA

LOOP_OUTPUT  . . . . . .        L NEAR    0006  CODE

MAIN . . . . . .                F PROC    0028  CODE  Length = 0043
MESSAGE_FINAL  . . . . . .      L BYTE    0015  DATA
MESSAGE_OUTPUT . . . . . .      L BYTE    0004  DATA

DELAY  . . . . . .              L NEAR    0010  CODE

SEG_PREV . . . . . .            L WORD    0000  DATA

@CPU . . . . . .                TEXT  0101h
@FILENAME  . . . . . .          TEXT  lab5
@VERSION . . . . . .            TEXT  510


     91 Source  Lines
     91 Total   Lines
     16 Symbols

  47930 + 434721 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```