

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.

Студентка гр. 1303

Андреева Е.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Изучить особенности прерываний на языке Ассемблера, написать собственное прерывание.

Задание.

Написать прерывание 08h - прерывание от системного таймера - генерируется автоматически операционной системой 18 раз в сек. Выполнить вывод сообщения на экран заданное число раз, после чего вставить фиксированную задержку и вывести сообщение о завершении обработчика. Отвести в программе под стек не менее 1К байт.

Ход работы.

В сегменте данных DATA содержится две переменных для хранения старого прерывания, содержавшегося по смещению 08h. Также в этом сегменте содержится MESSAGE – сообщение, которое будет выводиться во время работы прерывания, END_MES – сообщение, которое будет выведено после завершения обработчика.

В сегменте Astack выделяется 1Кбайт памяти, то есть db 1024.

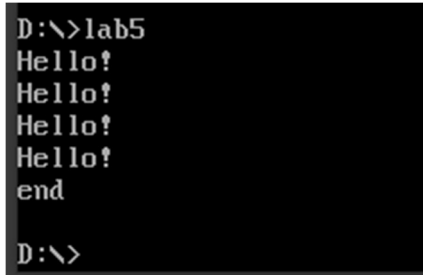
В сегменте кода сначала определяем процедуру для печати сообщения WriteMsg и процедуру пользовательского прерывания FUNC. В процедуре FUNC сначала сохраняются в стеке значения регистров до входа в прерывание. Далее с помощью lp строка из dx выводится заданное в cx количество раз. Далее реализована задержка после вывода строк с помощью прерывания 15h. После выводится сообщение об окончании обработки.

Вызов прерывания происходит в процедуре MAIN. В переменных, указанных в сегменте данных, сохраняется старое прерывание. Новое прерывание FUNC записывается по смещению 08h, с помощью функции 25h прерывания 21h.

Исходный код программы см. в приложении А.

Тестирование.

Работа программы с заданными условиями представлена на рис. 1



```
D:\>lab5
Hello!
Hello!
Hello!
Hello!
end
D:\>
```

Рис.1 – Результат работы программы

Вывод.

В ходе выполнения лабораторной работы были изучены виды прерываний и работа с ними. В соответствии с заданием было разработано собственное прерывание.

Приложение А

Исходный код программы

Название файла: lab5.asm

```
AStack  SEGMENT STACK
        DB 1024 DUP(?)
AStack  ENDS

DATA     SEGMENT
        KEEP_CS DW 0
        KEEP_IP DW 0
        MESSAGE DB 'Hello!', 0dh, 0ah, '$'
        END_MES DB 'end', 0dh, 0ah, '$'
        EMPTY DB ' ', '$'
DATA     ENDS

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

WriteMsg PROC NEAR
        mov AH, 9
        int 21h
        ret
WriteMsg ENDP

FUNC PROC FAR
        push ax
        push bx
        push cx
        push dx
        push ds

        mov dx, OFFSET MESSAGE
        mov cx, 4
        lp:
                call WriteMsg
                loop lp

        mov al, 0
        mov  ah,86h
        xor  cx,cx
        mov  dx,10000
        int  15h

        mov dx, OFFSET END_MES
        call WriteMsg

        pop ds
        pop dx
        pop cx
        pop bx
        pop ax
        mov al, 20h
        out 20h, al
        iret
FUNC ENDP

MAIN PROC FAR
        push ds
        sub ax, ax
        push ax
```

```

    mov ax, DATA
    mov ds, ax

    mov ah, 35h
    mov al, 08h
    int 21h
    mov KEEP_IP, bx
    mov KEEP_CS, es

    push ds
    mov dx, OFFSET FUNC
    mov ax, SEG FUNC
    mov ds, ax
    mov ah, 25h
    mov al, 08h
    int 21h
    pop ds

    int 08h

    cli
    push ds
    mov dx, KEEP_IP
    mov ax, KEEP_CS
    mov ds, ax
    mov ah, 25h
    mov al, 08h
    int 21h
    pop ds
    sti
    mov ah, 4ch
    int 21h
MAIN ENDP
CODE ENDS
    END MAIN

```

Приложение Б
Листинг программы

Название файла: lab5.lst

```
0000          AStack SEGMENT STACK
0000 0400[          DB 1024 DUP(?)
    ??
    ]

0400          AStack ENDS


0000          DATA SEGMENT
0000 0000          KEEP_CS DW 0
0002 0000          KEEP_IP DW 0
0004 48 65 6C 6C 6F 21          MESSAGE DB 'Hello!', 0dh, 0ah, '$'
    0D 0A 24
000D 65 6E 64 0D 0A 24          END_MES DB 'end', 0dh, 0ah, '$'
0013 20 24          EMPTY DB ' ', '$'
0015          DATA ENDS


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


0000          WriteMsg PROC NEAR
0000 B4 09          mov AH, 9
0002 CD 21          int 21h
0004 C3          ret
0005          WriteMsg ENDP


0005          FUNC PROC FAR
```

0005 50	push ax
0006 53	push bx
0007 51	push cx
0008 52	push dx
0009 1E	push ds
000A BA 0004 R	mov dx, OFFSET MESSAGE
000D B9 0004	mov cx, 4
0010	lp:
0010 E8 0000 R	call WriteMsg
0013 E2 FB	loop lp
0015 B0 00	mov al, 0
0017 B4 86	mov ah, 86h
0019 33 C9	xor cx, cx
001B BA 2710	mov dx, 10000
001E CD 15	int 15h
0020 BA 000D R	mov dx, OFFSET END_MES
0023 E8 0000 R	call WriteMsg
0026 1F	pop ds
0027 5A	pop dx
0028 59	pop cx
0029 5B	pop bx
002A 58	pop ax
002B B0 20	mov al, 20h
002D E6 20	out 20h, al


```
002F CF                                ired
0030                                FUNC ENDP

0030                                MAIN PROC FAR
0030 1E                                push ds
0031 2B C0                            sub ax, ax
0033 50                                push ax
0034 B8 ---- R                        mov ax, DATA
0037 8E D8                            mov ds, ax

0039 B4 35                            mov ah, 35h
003B B0 08                            mov al, 08h
003D CD 21                            int 21h
003F 89 1E 0002 R                    mov KEEP_IP, bx
0043 8C 06 0000 R                    mov KEEP_CS, es

0047 1E                                push ds
0048 BA 0005 R                        mov dx, OFFSET FUNC
004B B8 ---- R                        mov ax, SEG FUNC
004E 8E D8                            mov ds, ax
0050 B4 25                            mov ah, 25h
0052 B0 08                            mov al, 08h
0054 CD 21                            int 21h
0056 1F                                pop ds

0057 CD 08                            int 08h
```

0059	FA	cli
005A	1E	push ds
005B	8B 16 0002 R	mov dx, KEEP_IP
005F	A1 0000 R	mov ax, KEEP_CS
0062	8E D8	mov ds, ax
0064	B4 25	mov ah, 25h
0066	B0 08	mov al, 08h
0068	CD 21	int 21h
006A	1F	pop ds
006B	FB	sti
006C	B4 4C	mov ah, 4ch
006E	CD 21	int 21h
0070		MAIN ENDP
0070		CODE ENDS
		END MAIN

Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0400	PARA		STACK
CODE	0070	PARA		NONE
DATA	0015	PARA		NONE

Symbols:

N a m e	Type	Value	Attr
EMPTY	L BYTE	0013	DATA
END_MES	L BYTE	000D	DATA
FUNC	F PROC	0005	CODE Length = 002B
KEEP_CS	L WORD	0000	DATA
KEEP_IP	L WORD	0002	DATA
LP	L NEAR	0010	CODE
MAIN	F PROC	0030	CODE Length = 0040
MESSAGE	L BYTE	0004	DATA

WRITEMSG N PROC 0000 CODE Length =
0005

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

92 Source Lines

92 Total Lines

17 Symbols

48016 + 461291 Bytes symbol space free

0 Warning Errors

0 Severe Errors