

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 8

Студент гр. 0382

Кондратов Ю.А.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2021

Цель работы.

Разработать собственное прерывание, выполняющее функции в соответствии с заданием.

Задание.

Прерывание пользователя – должно генерироваться в программе, должно выполнять вывод сообщения на экран заданное число раз, после чего выставить фиксированную задержку и выводить сообщение о завершении работы.

Выполнение работы.

DATA – сегмент данных в программе. Он содержит: `old_seg`, `old_ip` – переменные для хранения старого прерывания, содержавшегося по смещению `60h`, `out_msg` – сообщение которое будет выводиться прерыванием, `end_msg` – сообщение о завершении работы прерывания.

AStack – сегмент стека в программе.

CODE – сегмент кода в программе.

Процедура пользовательского прерывания называется `CUSTOM_INT`. В ней сначала на стеке сохраняются значения регистров при входе в прерывание. Далее при помощи метки `print_loop` выводится строка, содержащаяся по адресу `DS:DX` количество раз заданное в `CX`.

Пауза после вывода строк реализуется при помощи прерывания `1Ah`. При вызове прерывания в регистре `bx` должна содержаться требуемая задержка (в тактах процессора). К требуемой задержке прибавляется текущее время в программе, которое прерыванием `1Ah` записывается в `CX`, `DX` (в `CX` – старшая часть значения). Далее в цикле происходит сравнение значения `bx` с текущим временем программы, если оно больше времени в `bx`, то производится выход из цикла.

Далее при помощи прерывания `21h` производится вывод завершающего сообщения, хранящегося по адресу `DS:offset end_msg`.

После вывода завершающего сообщения производится восстановление регистров из стека и выход из прерывания.

Вызов прерывания производится в процедуре MAIN. Для этого сначала при помощи прерывания 21h происходит получение прерывания, хранящегося по смещению 60h. Старое прерывание сохраняется в переменных old_seg, old_ip.

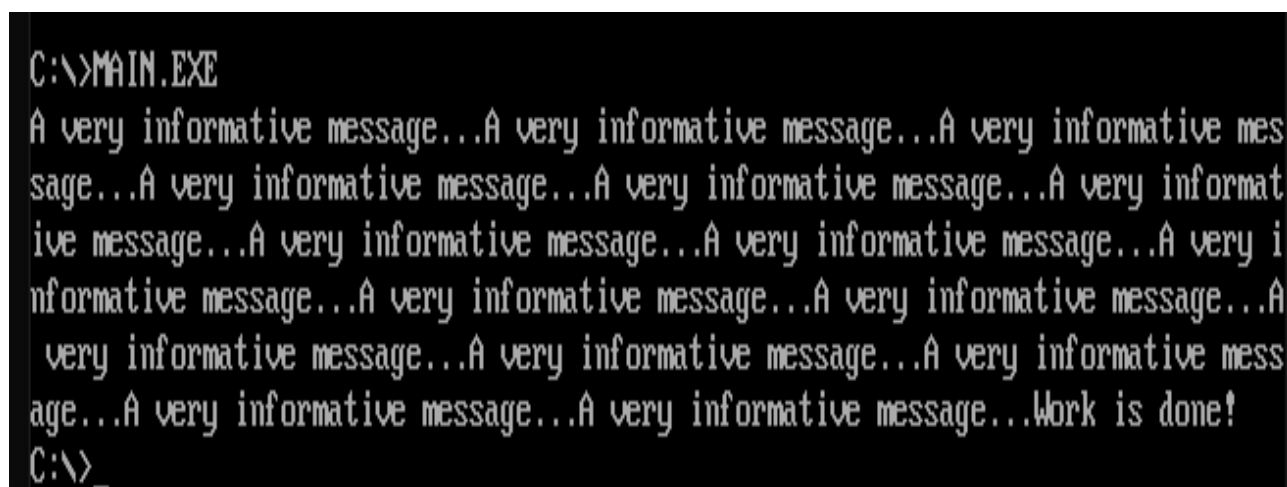
Далее также при помощи прерывания 21h происходит запись по смещению 60h нового прерывания CUSTOM_INT.

Когда прерывание установлено, происходит заполнение регистров в соответствии с инструкцией по использованию прерывания: в ds:dx должна лежать выводимая несколько раз строка, в cx – количество раз сколько нужно вывести строку, в bx – время паузы (в тиках процессора), в ds:offset end_msg – сообщение о завершении.

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

Тестирование.

Для проверки работоспособности программы при вызове прерывания установлены значения регистров: dx – offset out_msg ('A very informative message...'), cx – 10h (16 decimal), bx – 36h (3 секунды), offset end_msg – "Work is done!". Результат работы программы представлен на рисунке 1. Вердикт – программа работает верно.



```
C:\>MAIN.EXE
A very informative message...A very informative message...A very informative mes
sage...A very informative message...A very informative message...A very informat
ive message...A very informative message...A very informative message...A very i
nformative message...A very informative message...A very informative message...A
very informative message...A very informative message...A very informative mess
age...A very informative message...A very informative message...Work is done!
C:\>
```

Рисунок 1 – Результат работы программы

Выводы.

В ходе работы были изучены основные принципы построения собственных прерывания и их вызова из основной программы. Была написана программа, выводящая строку заданное количество раз, после выставяющая задержку на заданное время и выводящая завершающее сообщение.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
main.asm:
DATA SEGMENT
    old_seg dw 0
    old_ip dw 0
    out_msg DB 'A very informative message...$'
    end_msg DB 'Work is done!$'
DATA ENDS

AStack SEGMENT STACK
    DW 512 DUP(?)
AStack ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:AStack
; ds:dx must contain message adress ends with '$'
; cx must contain number of prints
; bx must contain time (in cpu ticks)
; data segment must contain 'end_msg' string
CUSTOM_INT PROC FAR
    ;storing registers
    push ax
    push bx
    push cx
    push dx

    ;print cx times
    mov ah, 9h
print_loop:
    int 21h
    loop print_loop

    ;pause
    mov ah, 0
    int 1Ah
    add bx, dx
pause:
    mov ah, 0
    int 1Ah
    cmp bx, dx
    jg pause

    ;printing end message
    mov dx, offset end_msg
    mov ah, 9h
    int 21h

    ;restoring registers
    pop dx
    pop cx
    pop bx
    pop ax

    ;return
    mov al, 20h
```

```

        out 20h, al
        iret
CUSTOM_INT ENDP

Main PROC FAR
    push DS
    sub ax, ax
    push ax
    mov ax, DATA
    mov ds, ax

    ;storing old int
    mov ax, 3560h
    int 21h
    mov old_seg, es
    mov old_ip, bx

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

    ;setting registers according to custom int manual
    mov dx, offset out_msg
    mov cx, 10h
    mov bx, 36h
    int 60h

    ;restoring old int
    CLI
    push ds
    mov dx, old_ip
    mov ax, old_seg
    mov ds, ax
    mov ax, 251ch
    int 21h
    pop ds
    STI

    ret
Main ENDP

CODE ENDS
    END Main

```

ПРИЛОЖЕНИЕ В

ЛИСТИНГ ПРОГРАММЫ

Microsoft (R) Macro Assembler Version 5.10
12:33:0

11/29/21

Page

1-1

```

0000                                DATA SEGMENT
0000 0000                                old_seg dw 0
0002 0000                                old_ip dw 0
0004 41 20 76 65 72 79                out_msg DB 'A very informative
message...$'
                                20 69 6E 66 6F 72
                                6D 61 74 69 76 65
                                20 6D 65 73 73 61
                                67 65 2E 2E 2E 24
0022 57 6F 72 6B 20 69                end_msg DB 'Work is done!$'
                                73 20 64 6F 6E 65
                                21 24
0030                                DATA ENDS

0000                                AStack SEGMENT STACK
0000 0200[                                DW 512 DUP(?)
                                ????
                                ]

0400                                AStack ENDS

0000                                CODE SEGMENT
                                ASSUME CS:CODE, DS:DATA, SS:AStack
                                ; ds:dx must contain message adress ends with
                                '$'
                                ; cx must contain number of prints
                                ; bx must contain time (in cpu ticks)
                                ; data segment must contain 'end_msg' string
0000                                CUSTOM_INT PROC FAR
                                ;storing registers
0000 50                                push ax
0001 53                                push bx
0002 51                                push cx
0003 52                                push dx

                                ;print cx times
0004 B4 09                                mov ah, 9h
0006                                print_loop:
0006 CD 21                                int 21h
0008 E2 FC                                loop print_loop

                                ;pause
000A B4 00                                mov ah, 0
000C CD 1A                                int 1Ah
000E 03 DA                                add bx, dx
0010                                pause:
0010 B4 00                                mov ah, 0

```

```

0012  CD 1A                int 1Ah
0014  3B DA                cmp bx, dx
0016  7F F8                jg pause

                                ;printing end message
0018  BA 0022 R            mov dx, offset end_msg
001B  B4 09                mov ah, 9h
001D  CD 21                int 21h

```

Microsoft (R) Macro Assembler Version 5.10
12:33:0

11/29/21

Page

1-2

```

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

                                ;return
0023  B0 20                mov al, 20h
0025  E6 20                out 20h, al
0027  CF                iret
0028                                CUSTOM_INT 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

                                ;storing old int
0031  B8 3560                mov ax, 3560h
0034  CD 21                int 21h
0036  8C 06 0000 R            mov old_seg, es
003A  89 1E 0002 R            mov old_ip, bx

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

                                ;setting registers according to custom int
                                manual
004D  BA 0004 R            mov dx, offset out_msg
0050  B9 0010                mov cx, 10h
0053  BB 0036                mov bx, 36h
0056  CD 60                int 60h

                                ;restoring old int
0058  FA                CLI

```



```

0059 1E          push ds
005A 8B 16 0002 R      mov dx, old_ip
005E A1 0000 R      mov ax, old_seg
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

```

Microsoft (R) Macro Assembler Version 5.10
12:33:0

11/29/21

Page

1-3

```

006B          Main ENDP

```

```

006B          CODE ENDS
                END Main

```

Microsoft (R) Macro Assembler Version 5.10
12:33:0

11/29/21

Symbols-1

Segments and Groups:

	N a m e	Length	Align	Combine Class
ASTACK		0400	PARA	STACK
CODE		006B	PARA	NONE
DATA		0030	PARA	NONE

Symbols:

	N a m e	Type	Value	Attr
0028	CUSTOM_INT	F PROC	0000	CODE Length =
	END_MSG	L BYTE	0022	DATA
0043	MAIN	F PROC	0028	CODE Length =
	OLD_IP	L WORD	0002	DATA
	OLD_SEG	L WORD	0000	DATA
	OUT_MSG	L BYTE	0004	DATA
	PAUSE	L NEAR	0010	CODE
	PRINT_LOOP	L NEAR	0006	CODE
	@CPU	TEXT	0101h	
	@FILENAME	TEXT	MAIN	
	@VERSION	TEXT	510	

101 Source Lines
101 Total Lines
16 Symbols

48002 + 459258 Bytes symbol space free

0 Warning Errors
0 Severe Errors