

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: Разработка собственного прерывания

Студент гр. 9383

Гордон Д.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Разработать на языке ассемблер собственное прерывание

Текст задания.

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.

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
```

Две последние строки обработчика прерывания, указанные перед командой IRET выхода из прерывания, необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное. 12

Замечание: в лабораторной работе действиями по обработке прерывания может быть вывод на экран некоторого текста, вставка цикла задержки в вывод сообщения или включение звукового сигнала.

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 соответствии с этим, программа должна содержать следующие инструкции:

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

Для установки адреса нового обработчика прерывания в поле векторов прерываний используется функция 25H прерывания 21H, которая помещает заданные адреса сегмента и смещения обработчика в вектор прерывания с заданным номером.

```
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

Далее может выполняться вызов нового обработчика прерывания.

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 старый вектор прерывания

POP DS

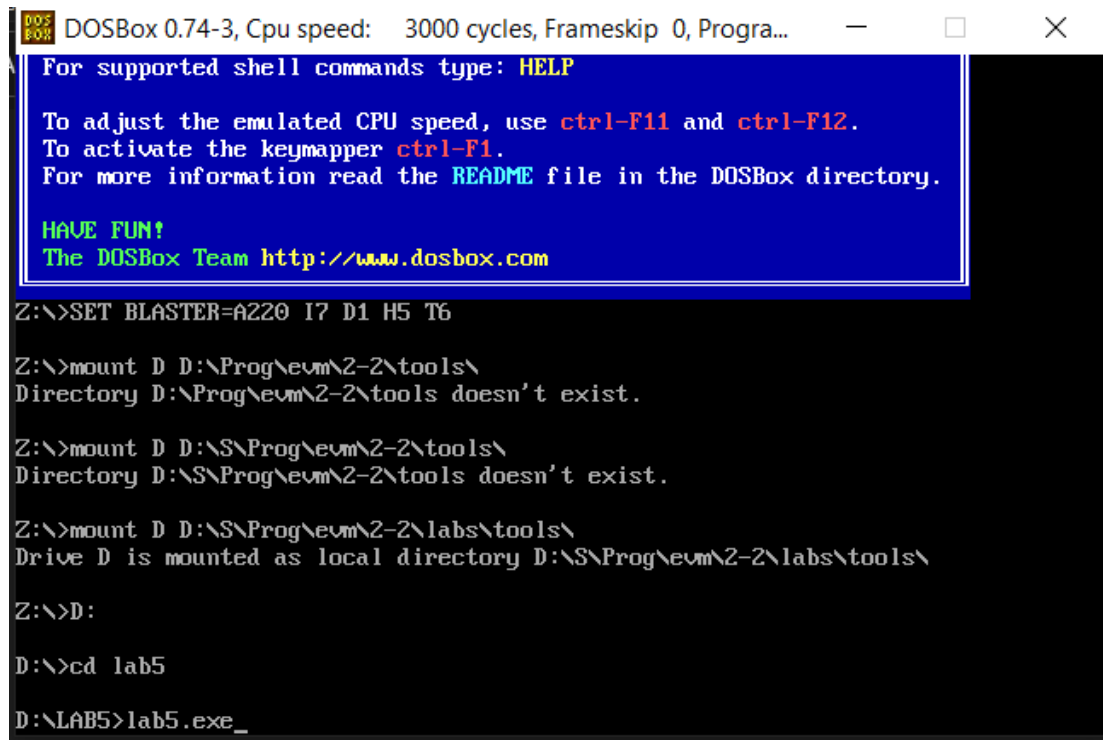
STI

Вариант №3 – шифр 1С

1 - 1Ch - прерывание от часов - генерируется автоматически операционной системой 18 раз в сек;

С - Приостановить вывод на экран (вставить цикл задержки).

ПРОТОКОЛ



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

For supported shell commands type: **HELP**

To adjust the emulated CPU speed, use **ctrl-F11** and **ctrl-F12**.
To activate the keymapper **ctrl-F1**.
For more information read the **README** file in the DOSBox directory.

HAVE FUN!
The DOSBox Team <http://www.dosbox.com>

```
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount D D:\Prog\evm\2-2\tools\
Directory D:\Prog\evm\2-2\tools doesn't exist.

Z:\>mount D D:\S\Prog\evm\2-2\tools\
Directory D:\S\Prog\evm\2-2\tools doesn't exist.

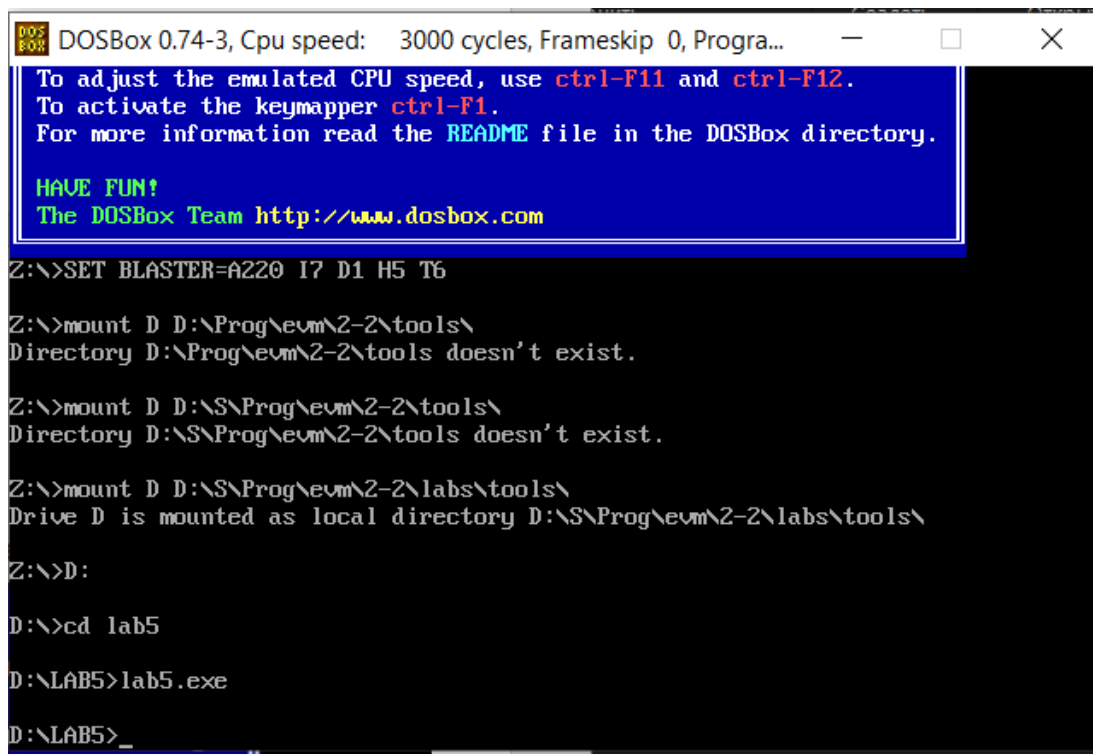
Z:\>mount D D:\S\Prog\evm\2-2\labs\tools\
Drive D is mounted as local directory D:\S\Prog\evm\2-2\labs\tools\

Z:\>D:

D:\>cd lab5

D:\LAB5>lab5.exe_
```

Рисунок 1 – До запуска lab5.exe



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

To adjust the emulated CPU speed, use **ctrl-F11** and **ctrl-F12**.
To activate the keymapper **ctrl-F1**.
For more information read the **README** file in the DOSBox directory.

HAVE FUN!
The DOSBox Team <http://www.dosbox.com>

```
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount D D:\Prog\evm\2-2\tools\
Directory D:\Prog\evm\2-2\tools doesn't exist.

Z:\>mount D D:\S\Prog\evm\2-2\tools\
Directory D:\S\Prog\evm\2-2\tools doesn't exist.

Z:\>mount D D:\S\Prog\evm\2-2\labs\tools\
Drive D is mounted as local directory D:\S\Prog\evm\2-2\labs\tools\

Z:\>D:

D:\>cd lab5

D:\LAB5>lab5.exe

D:\LAB5>_
```

Рисунок 2 – После запуска lab5.exe (прошло 8 секунд)

ВЫВОДЫ

Поставленная задача была выполнена – написано собственное прерывание на языке ассемблер.

ПРИЛОЖЕНИЕ

Lab5.asm:

EOF EQU '\$'

AStack SEGMENT STACK
DB 1024 DUP(?)
AStack ENDS

DATA SEGMENT
KEEP_CS DW 0 ; для хранения сегмента вектора прерывания
KEEP_IP DW 0 ; и смещения вектора прерывания
DATA ENDS

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

SUBR_INT PROC FAR
PUSH AX
PUSH DX
PUSH CX

MOV CX, 0088H
MOV DX, 00FFH ; 0088H и 00FFH - 8 сек
MOV AH, 86H ;delay func
INT 15H ;

MOV AL, 20H
OUT 20H,AL

POP AX
POP CX
POP DX
IRET
SUBR_INT ENDP

MAIN PROC FAR
PUSH DS ; сохранение адреса начала PSP в стеке для последующего
восстановления по команде ret
MOV AX, DATA ; загрузка сегментного регистра данных
MOV DS, AX

```

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 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, 1CH ; номер вектора
INT 21H ; меняем прерывание
POP DS

INT 1CH

CLI
PUSH DS
MOV DX, KEEP_IP
MOV AX, KEEP_CS
MOV DS, AX
MOV AH, 25H
MOV AL, 1CH
INT 21H ; восстанавливаем старый вектор прерывания
POP DS
STI

RET
MAIN ENDP
CODE ENDS
END MAIN

```


Lab5.lst:

```

= 0024                                EOF EQU '$'

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

0400                                AStack ENDS

0000                                DATA SEGMENT
0000 0000                            KEEP_CS DW 0 ; для хранения э
    • 0x00000000 IP вектора прерыван
    ия
0002 0000                            KEEP_IP DW 0 ; и смещения веИ
    0x00000000 прерывания
0004                                DATA ENDS

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

0000                                SUBR_INT PROC FAR
0000 50                                PUSH AX
0001 52                                PUSH DX
0002 51                                PUSH CX

0003 B9 0088                            MOV CX, 0088H
0006 BA 00FF                            MOV DX, 00FFH ; 0088H и 00FFH - 8 сек
0009 B4 86                            MOV AH, 86H ;delay func
000B CD 15                            INT 15H ;

000D B0 20                            MOV AL, 20H
000F E6 20                            OUT 20H,AL

0011 58                                POP AX
0012 59                                POP CX
0013 5A                                POP DX
0014 CF                                IRET
0015                                SUBR_INT ENDP

0015                                MAIN PROC FAR

```

0015	1E	PUSH DS ; сохранение адреса начала PSP в стеке для посл едующего восстановления И
		⌘□команде ret
0016	B8 ---- R	MOV AX, DATA ; загрузка сегмента ДР—Е регистра данных
0019	8E D8	MOV DS, AX
001B	B4 35	MOV AH, 35H ; функция получен ия вектора
001D	B0 1C	MOV AL, 1CH ; номер вектора
001F	CD 21	INT 21H
0021	89 1E 0002 R	MOV KEEP_IP, BX ; запоминание с мещения

```

0025 8C 06 0000 R      MOV KEEP_CS, ES ; и сегмента веИ
                        ъДЎРРпрерывания

0029 1E                PUSH DS
002A BA 0000 R      MOV DX, OFFSET SUBR_INT ; смещение
                        для процедуры в DX
002D B8 ---- R      MOV AX, SEG SUBR_INT ; сегмент прИ
                        ъї□8dQ_
0030 8E D8          MOV DS, AX ; помещаем в DS
0032 B4 25          MOV AH, 25H ; функция установ
                        ки вектора
0034 B0 1C          MOV AL, 1CH ; номер вектора
0036 CD 21          INT 21H ; меняем прерывание
0038 1F            POP DS

0039 CD 1C          INT 1CH

003B FA            CLI
003C 1E            PUSH DS
003D 8B 16 0002 R    MOV DX, KEEP_IP
0041 A1 0000 R      MOV AX, KEEP_CS
0044 8E D8          MOV DS, AX
0046 B4 25          MOV AH, 25H
0048 B0 1C          MOV AL, 1CH
004A CD 21          INT 21H ; восстанавливаем с
                        тарый вектор прерывания
004C 1F            POP DS
004D FB            STI

004E CB            RET
004F                MAIN ENDP
004F                CODE ENDS
                        END MAIN

```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0400	PARA		STACK
CODE	004F	PARA		NONE
DATA	0004	PARA		NONE

Symbols:

N a m e	Type	Value	Attr	
EOF	NUMBER	0024		
KEEP_CS	L WORD	0000	DATA	
KEEP_IP	L WORD	0002	DATA	
MAIN	F PROC	0015	CODE	Length = 003A
SUBR_INT	F PROC	0000	CODE	Length = 0015
@CPU	TEXT	0101h		
@FILENAME	TEXT	lab5		
@VERSION	TEXT	510		

70 Source Lines

70 Total Lines

13 Symbols

48020 + 461287 Bytes symbol space free

0 Warning Errors

0 Severe Errors