

**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»**

Студент группы 1304

\_\_\_\_\_

Завражин Д.Г.

Преподаватель

\_\_\_\_\_

Кириячиков В.А.

Санкт-Петербург  
2022

## **Цель работы**

Освоить азы трансляции, выполнения, модификации и отладки программ на языке Ассемблера процессора Intel X86; обучиться процессу написания собственного прерывания.

## **Основные теоретические положения**

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.

Операнд в команде прерывания, например, INT 12H, содержит тип прерывания, который идентифицирует запрос. Для каждого типа система содержит адрес в таблице векторов прерываний, начинающейся по адресу 0000. Так как в таблице имеется 256 четырехбайтовых элементов, то она занимает первые 1024 байта памяти от шест.0 до шест.3FF. Каждый элемент таблицы указывает на подпрограмму обработки указанного типа прерывания и содержит адрес кодового сегмента и смещение, которые при прерывании устанавливаются в регистры CS и IP соответственно. 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.

## **Экспериментальные результаты**

При выполнении лабораторной работы был использован эмулятор DOSBox версии 0.74.3-2, позволяющий работать с 16-разрядными исполняемыми файлами.

Для выполнения лабораторной работы требуется разработать программу с использованием самостоятельно написанного обработчика некоторого прерывания. Варианту 28 соответствует шифр задания 4g, причём цифра в шифре задает номер и назначение заменяемого вектора прерывания (16h), а буква определяет

следующие действия, реализуемые программой обработки прерываний: «Выполнить ввод и печать заданного количества символов по одному, после чего вывести сообщение о завершении обработчика».

В коде используются следующие введённые при помощи директивы EQU символы:

```
0 vector      EQU 16h
1 get_vector   EQU 3516h
2 set_vector   EQU 2516h
3 endl        EQU '$'
```

Для сохранения с целью последующего восстановления исходных значений из таблицы векторов прерываний, был применён следующий подход с использованием функции 35h прерывания 21h с последующей записью полученных значений в память:

```
28      MOV     AX,get_vector           ; В AX задаются номера функции и изменяемого вектора
29      INT     21h                     ; Вызывается обработчик прерывания 21h
30      MOV     word ptr [vector_ip],BX ; Сохраняется смещение исходного обработчика прерывания
31      MOV     word ptr [vector_ip+2],ES ; Сохраняется сегмент исходного обработчика прерывания
```

Для изменения таблицы векторов прерываний используется вызов функции 25h прерывания 21h, куда через регистр DS передаётся сегмент кода, где расположен новый обработчик прерываний, а через DX – его смещение. Это было реализовано нами следующим образом:

```
33      CLI
34      MOV     AX,SEG Handler          ; В DS загружается сегмент кода, в котором находится Handler
35      MOV     DS,AX
36      MOV     AX,set_vector           ; В AX задаются номера функции и изменяемого вектора
37      MOV     DX,OFFSET Handler       ; В DX загружается смещение обработчика Handler
38      INT     21h                     ; Вызывается обработчик прерывания 21h
39      STI
```

Для восстановления исходных значений в таблице векторов прерываний были использованы следующие команды:

```
43      CLI
44      LDS     DX,dword ptr [vector_ip] ; В DS:DX загружаются сегмент и смещение исходного обраб.
45      INT     21h                     ; Вызывается обработчик прерывания 21h
46      STI
```

Так как значение регистра AX между двумя вызовами функции 25h прерывания 21h не меняется, заново заносить в него номера функции и изменяемого вектора не требуется.

Так как в собственном обработчике прерываний требуется определение скан-кода и ASCII-кода, соответствующих нажатию клавиши, из него целесообразно исходный обработчик прерывания 16h. Для этого был создан следующий макрос,

достигающий этого путём совершения безусловного перехода по записанному в памяти логическому адресу исходного обработчика с предварительной записью в стек значения регистра флагов и логического адреса следующей за безусловным переходом операции в указанном порядке:

```

6  INTC MACRO addr
7      XOR AX,AX ; AX обнуляется (задаётся функция 0)
8      PUSHF    ; На стек кладётся регистр флагов
9      CALL addr ; Вызывается обработчик прерывания
10 ENDM

```

В собственном обработчике ввод и печать заданного скан-кодом введённого с клавиатуры символа количества символов по одному реализована следующим образом:

```

63      INTC vector_ip ; С клавиатуры считывается символ
64      XOR CX,CX      ; CX обнуляется
65      MOV CL,AH       ; Скан-код введённого символа заносится в CL
66  handler_loop:
67      INTC vector_ip ; С клавиатуры считывается символ
68      CMP AL,0       ; Если AL = 0, то это спец. клавиша, и в AH не скан-код, а ASCII-код
69      JE handler_loop
70      MOV DL,AL       ; ASCII-код введённого символа заносится в DL
71      MOV AH,2        ; В AL заносится код функции печати одного символа прерывания 21h
72      INT 21h         ; Вызывается обработчик прерывания 21h
73      LOOP handler_loop
74      MOV DL,10       ; Производится переход на новую строку
75      INT 21h
76      MOV DL,13
77      INT 21h

```

Так как скан-коды положительны, отдельная проверка на нулевое значение не требуется. Полный исходный код программы приведён в Приложении 1.

Для проверки корректности работы обработчика прерываний был составлен набор из тестов, представленных в Таблице 1, в которой проверяется зависимость количества вводимых с клавиатуры символов от значения скан-кода задающего их количество нажатия клавиши, которое не учитывается нами в приводимых значениях, так как получаемый таким образом символ не выводится в консоль.

Таблица 1 – Тесты

Входные данные	Наблюдаемое поведение	Ожидаемое поведение
AH = 2	Ввод/вывод двух символов	Ввод/вывод двух символов
AH = 3	Ввод/вывод трёх символов	Ввод/вывод трёх символов
AH = 4	Ввод/вывод четырёх символов	Ввод/вывод четырёх символов
AH = 5	Ввод/вывод пяти символов	Ввод/вывод пяти символов
AH = 6	Ввод/вывод шести символов	Ввод/вывод шести символов
AH = 7	Ввод/вывод семи символов	Ввод/вывод семи символов

Полный вывод программы на данных тестах приведён на Рисунках 1,2,3,4,5,6 в Приложении 2.

## **Выводы**

В процессе выполнения лабораторной работы были освоены азы трансляции, выполнения, модификации и отладки программ на языке Ассемблера процессора Intel X86. На практике был изучен процесс написания собственного прерывания.

Основным результатом работы стал исполняемый файл *lab5.exe*, реализующий логику работы с собственноручно написанным обработчиком прерывания.

# ПРИЛОЖЕНИЕ 1

## Исходный код программы *lab5.asm*

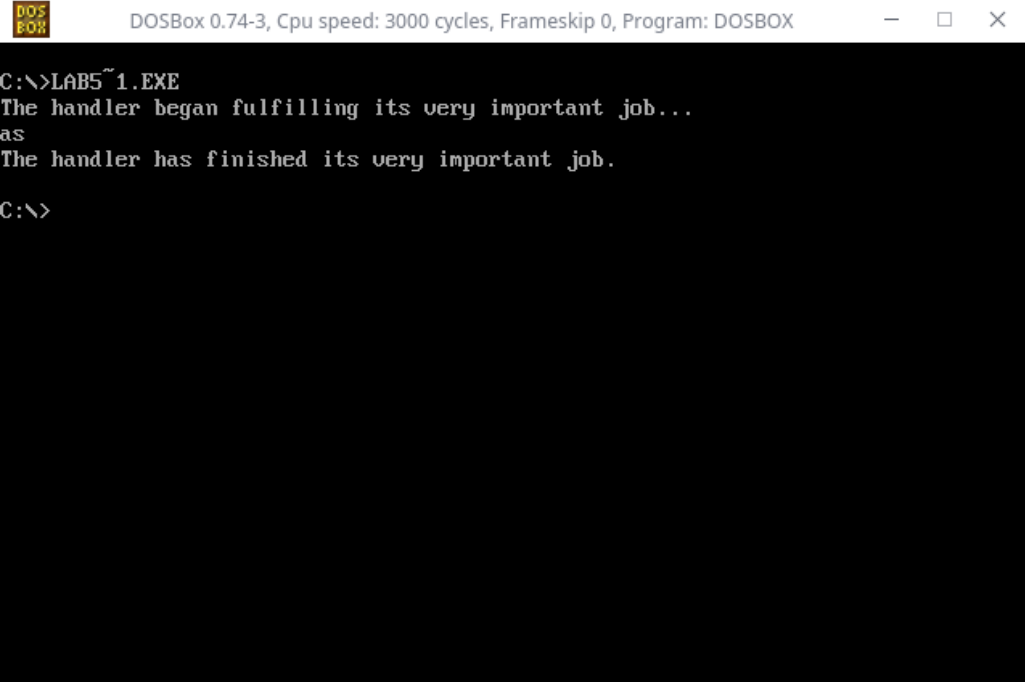
```

1  vector      EQU 16h
2  get_vector  EQU 3516h
3  set_vector  EQU 2516h
4  endl        EQU '$'
5
6  INTC MACRO addr
7      XOR AX,AX
8      PUSHF
9      CALL addr
10  ENDM
11
12  DOSSEG
13  .MODEL SMALL
14  .STACK 400h
15  .DATA
16      vector_ip DD 0
17      initial_message DB 'The handler began fulfilling its very important job...',10,13,endl
18      final_message DB 'The handler has finished its very important job.',9,13,endl
19  .CODE
20  Main PROC FAR
21      PUSH DS
22      SUB AX,AX
23      PUSH AX
24
25      MOV AX,@data
26      MOV DS,AX
27
28      MOV AX,get_vector
29      INT 21h
30      MOV word ptr [vector_ip],BX
31      MOV word ptr [vector_ip + 2],ES
32
33      CLI
34      MOV AX,SEG Handler
35      MOV DS,AX
36      MOV DX,OFFSET Handler
37      MOV AX,set_vector
38      INT 21h
39      STI
40
41      INT 16h
42
43      CLI
44      LDS DX,dword ptr [vector_ip]
45      INT 21h
46      STI
47
48      RET
49  Main ENDP
50
51  Handler PROC FAR
52      PUSH AX
53      PUSH CX
54      PUSH DX
55      PUSH DS
56
57      MOV AX,@data
58      MOV DS,AX
59      MOV AH,9
60      MOV DX,OFFSET initial_message
61      INT 21h
62
63      INTC vector_ip
64      XOR CX,CX
65      MOV CL,AH
66      handler_loop:
67      INTC vector_ip
68      CMP AL,0
69      JE handler_loop
70      MOV DL,AL
71      MOV AH,2
72      INT 21h
73      LOOP handler_loop
74      MOV DL,10
75      INT 21h
76      MOV DL,13
77      INT 21h
78
81      MOV AH,9
82      MOV DX,OFFSET final_message
83      INT 21h
84
85      POP DS
86      POP DX
87      POP CX
88      MOV AL,20h
89      OUT 20h,AL
90      POP AX
91      IRET
92  Handler ENDP
93  END

```

## ПРИЛОЖЕНИЕ 2

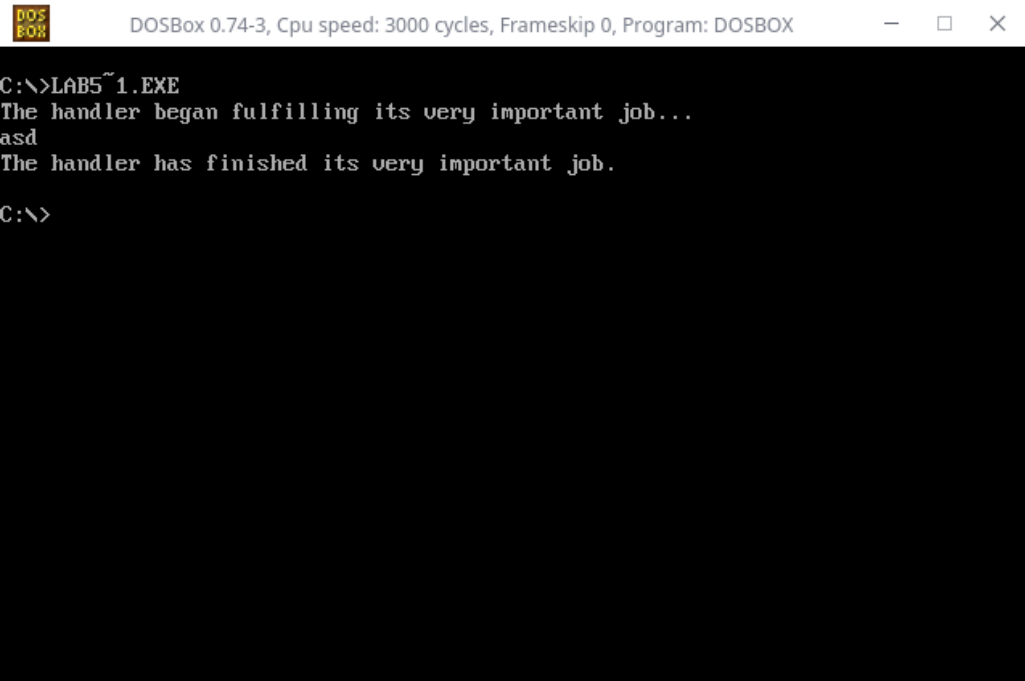
### Полный вывод программы при выполнении тестов



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
C:\>LAB5~1.EXE
The handler began fulfilling its very important job...
as
The handler has finished its very important job.
C:\>
```

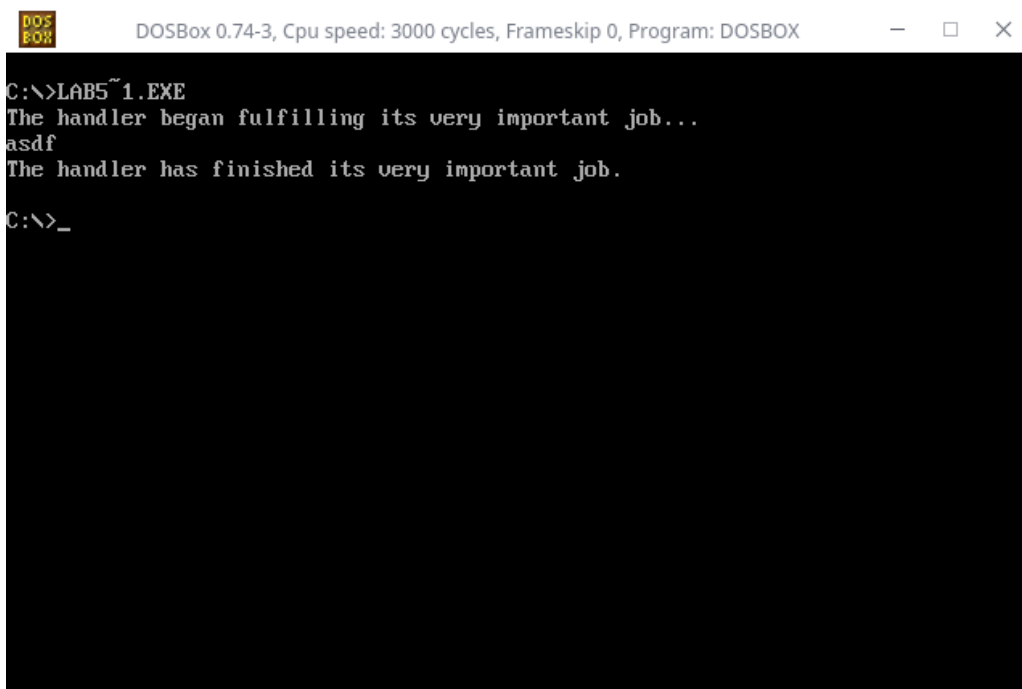
Рис. 1: Полный вывод программы при вводе символов «las» с клавиатуры



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
C:\>LAB5~1.EXE
The handler began fulfilling its very important job...
asd
The handler has finished its very important job.
C:\>
```

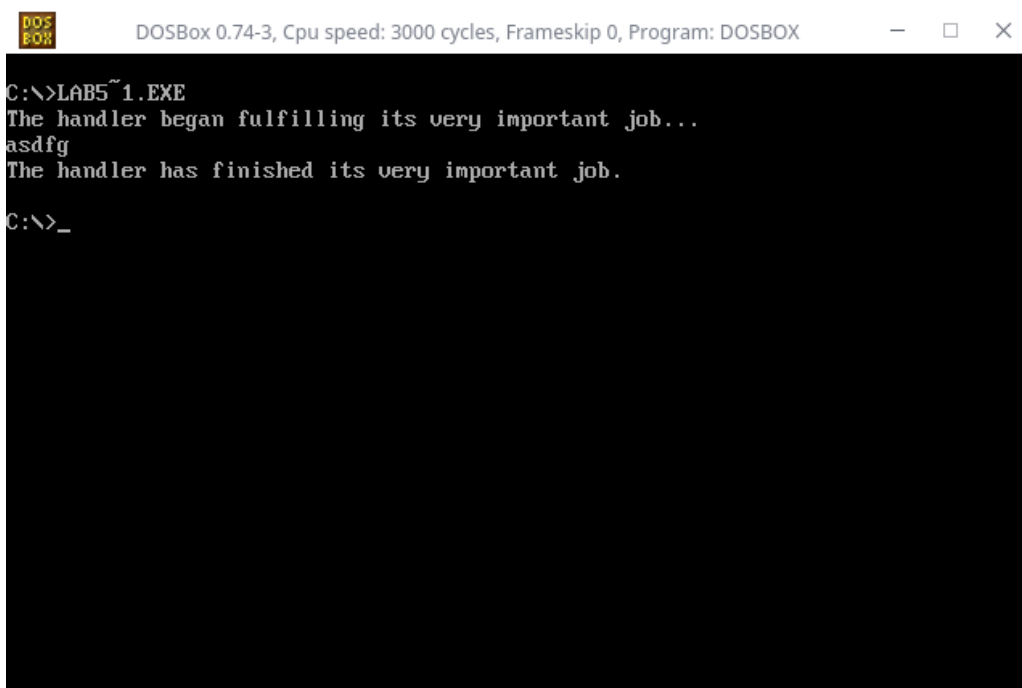
Рис. 2: Полный вывод программы при вводе символов «2asd» с клавиатуры



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
C:\>LAB5~1.EXE
The handler began fulfilling its very important job...
asdf
The handler has finished its very important job.
C:\>_
```

Рис. 3: Полный вывод программы при вводе символов «3asdf» с клавиатуры

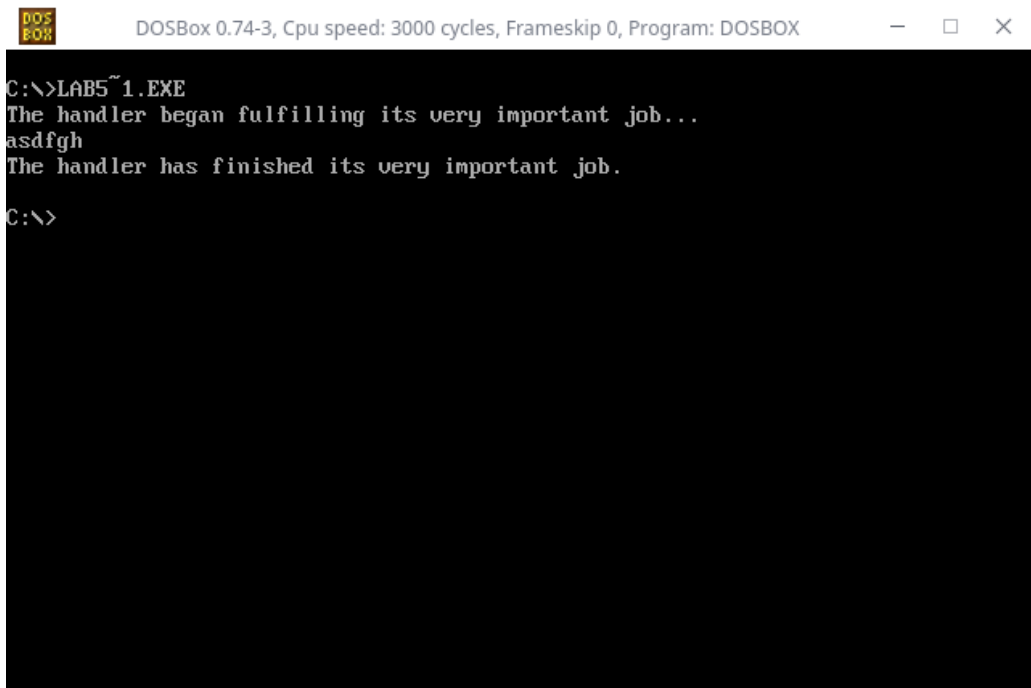


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
C:\>LAB5~1.EXE
The handler began fulfilling its very important job...
asdfg
The handler has finished its very important job.
C:\>_
```

Рис. 4: Полный вывод программы при вводе символов «4asdfg» с клавиатуры

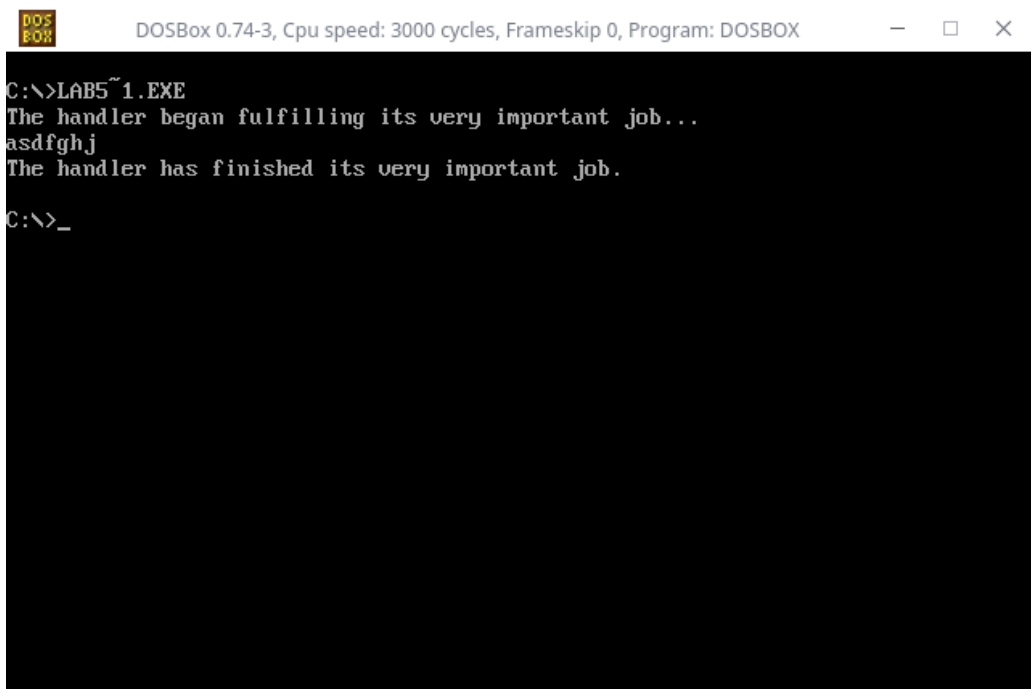




DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
C:\>LAB5~1.EXE
The handler began fulfilling its very important job...
asdfgh
The handler has finished its very important job.
C:\>
```

Рис. 5: Полный вывод программы при вводе символов «5asdfgh» с клавиатуры



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
C:\>LAB5~1.EXE
The handler began fulfilling its very important job...
basdfghj
The handler has finished its very important job.
C:\>_
```

Рис. 6: Полный вывод программы при вводе символов «basdfghj» с клавиатуры