

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 8 соответствует шифр задания 2а, причём цифра в шифре задает номер и назначение заменяемого вектора прерывания (60h), а буква определяет

следующие действия, реализуемые программой обработки прерываний: «Выполнить вывод сообщения на экран заданное число раз, после чего вставить фиксированную задержку и вывести сообщение о завершении обработчика».

В коде используются следующие введённые при помощи директивы EQU символы:

```
0 vector      EQU 60h
1 get_vector  EQU 3560h
2 set_vector  EQU 2560h
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:

```
19      MOV     AX,get_vector          ; В AX задаются номера функции и изменяемого вектора
20      INT     21h                   ; Вызывается обработчик прерывания 21h
21      MOV     word ptr [vector_ip],BX ; Сохраняется смещение исходного обработчика прерывания
22      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:

```
24      CLI
25      MOV     AX,SEG Handler        ; В DS загружается сегмент кода, в котором находится Handler
26      MOV     DS,AX
27      MOV     AX,set_vector         ; В AX задаются номера функции и изменяемого вектора
28      MOV     DX,OFFSET Handler    ; В DX загружается смещение обработчика Handler
29      INT     21h                   ; Вызывается обработчик прерывания 21h
30      STI
```

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

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

Количество повторений выводимой в собственном обработчике прерывания 60h строки передаётся через регистр CX. Вывод строки реализован следующим образом:

```

57     CMP    CX,0 ; ; Производится сравнение CX с нулём
58     JE     skip_repeating ; ; Если ноль, пропускаем участок кода с выводом
59     MOV    AH,9 ; ; В AH загружается номер функции вывода строки
60     MOV    DX,OFFSET repeating_message ; ; В DX загружается смещение выводимой строки
61 repeat:
62     INT     21h ; ; Вызывается обработчик прерывания 21h
63     LOOP   repeat ; ; Вывод повторяется, пока CX не уменьшится до нуля
64 skip_repeating:

```

Задержка была реализована через вызов функции 86h прерывания 15h, обеспечивающую задержку на находящееся в паре регистров CX:DX 32-разрядное количество микросекунд. В частности, для обеспечения задержки на одну секунду, которая равняется 1000000_{10} в десятичной или $F4240_{16}$ в шестнадцатеричной системе счисления микросекунд, в регистр CX требуется занести $000Fh$, а в DX – $4240h$. Таким образом, фиксированная задержка в одну секунду обеспечивается следующим участком кода:

```

66     MOV    AH,86h ; ; В AH загружается номер функции вывода строки
67     MOV    CX,0Fh ; ; В CX:DX загружается значение F4240h (1000000 микросекунд)
68     MOV    DX,4240h
69     INT     15h ; ; Вызывается обработчик прерывания 15h

```

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 появлений повторяющегося сообщения в консоли от хранящегося в регистре CX значения. Фиксированная задержка и вывод заключительной строки наблюдались во всех случаях, чего и следовало ожидать.

Таблица 1 – Тесты

Входные данные	Полученные значения	Ожидаемые значения
CX = 0	Не выводится	Не выводится
CX = 1	Одно повторение	Одно повторение
CX = 2	Два повторения	Два повторения
CX = 3	Три повторения	Три повторения
CX = 4	Четыре повторения	Четыре повторения
CX = 5	Пять повторений	Пять повторений

Полный вывод программы на данных тестах приведён на Рисунках 1,2,3,4,5,6 в Приложении 2. В тестах подразумевается, что обработчик пользовательского прерывания 60h вызывается следующим образом (где N – некоторое число):

```

32     MOV    CX,N
33     INT     vector

```

В приведённом в Приложении 1 полном исходном коде программы прерывание 60h вызывается четыре раза следующим образом:

```
32     MOV     CL,0
33     INT     vector
34     MOV     CL,5
35     INT     vector
36     INT     vector
37     MOV     CH,1
38     INT     vector
```

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*

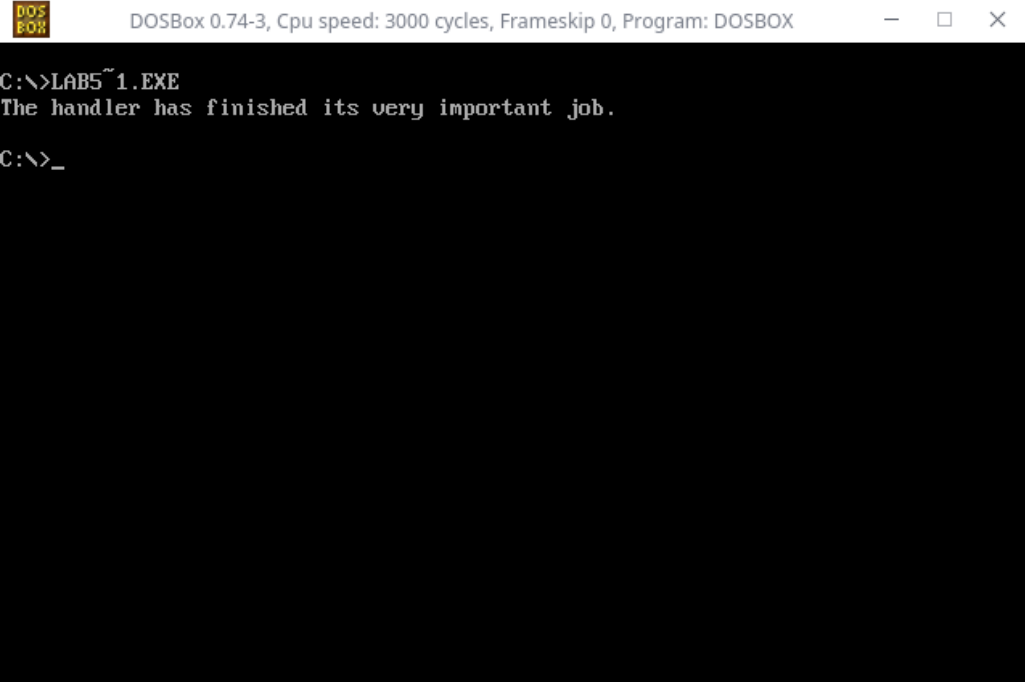
```

1  vector      EQU 60h
2  get_vector  EQU 3560h
3  set_vector  EQU 2560h
4  endl        EQU '$'
5
6  DOSSEG
7  .MODEL SMALL
8  .STACK 400h
9  .DATA
10     repeating_message DB 'This message is to be repeated CX times...',10,13,endl
11     final_message DB 'The handler has finished its very important job.',10,13,endl
12 .CODE
13     Main PROC FAR
14     vector_ip:
15         PUSH DS
16         SUB AX,AX
17         PUSH AX
18
19         MOV AX,get_vector
20         INT 21h
21         MOV word ptr [vector_ip],BX
22         MOV word ptr [vector_ip + 2],ES
23
24         CLI
25         MOV AX,SEG Handler
26         MOV DS,AX
27         MOV AX,set_vector
28         MOV DX,OFFSET Handler
29         INT 21h
30         STI
31
32         MOV CL,0
33         INT vector
34         MOV CL,5
35         INT vector
36         INT vector
37         MOV CH,1
38         INT vector
39
40         CLI
41         LDS DX,dword ptr [vector_ip]
42         INT 21h
43         STI
44
45         RET
46     Main ENDP
47
48     Handler PROC FAR
49         PUSH AX
50         PUSH CX
51         PUSH DX
52         PUSH DS
53
54         MOV AX,@data
55         MOV DS,AX
56
57         CMP CX,0
58         JE skip_repeating
59         MOV AH,9
60         MOV DX,OFFSET repeating_message
61     repeat:
62         INT 21h
63
64     LOOP repeat
65     skip_repeating:
66         MOV AH,86h
67         MOV CX,0Fh
68         MOV DX,4240h
69         INT 15h
70
71         MOV AH,9
72         MOV DX,OFFSET final_message
73         INT 21h
74
75         POP DS
76         POP DX
77         POP CX
78         MOV AL,20h
79         OUT 20h,AL
80         POP AX
81         IRET
82     Handler ENDP
83 END

```

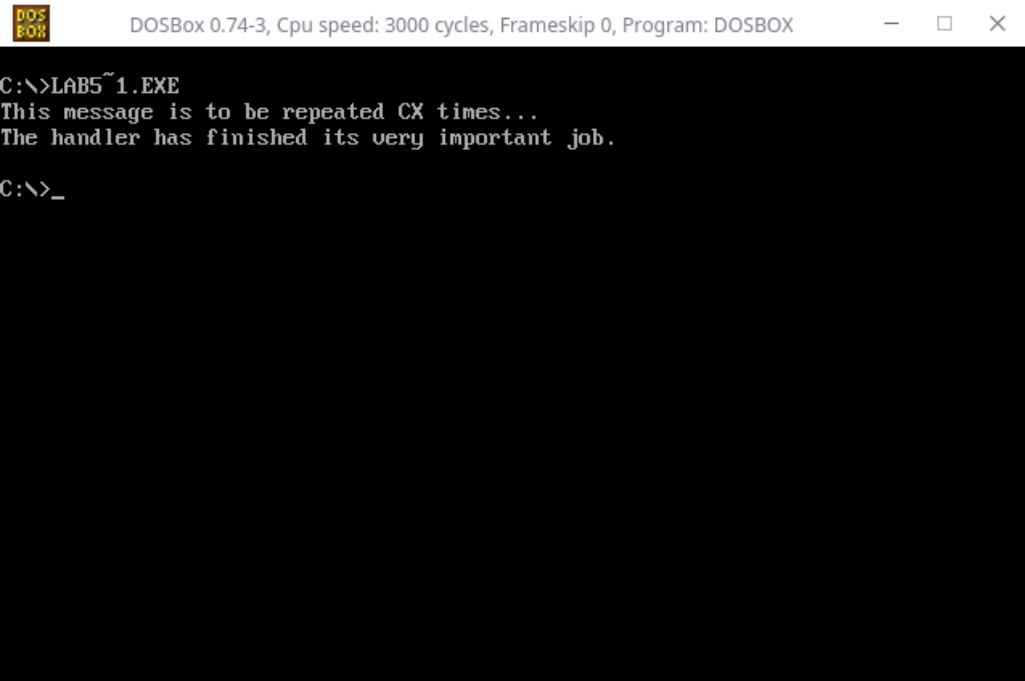
ПРИЛОЖЕНИЕ 2

Полный вывод программы при выполнении тестов



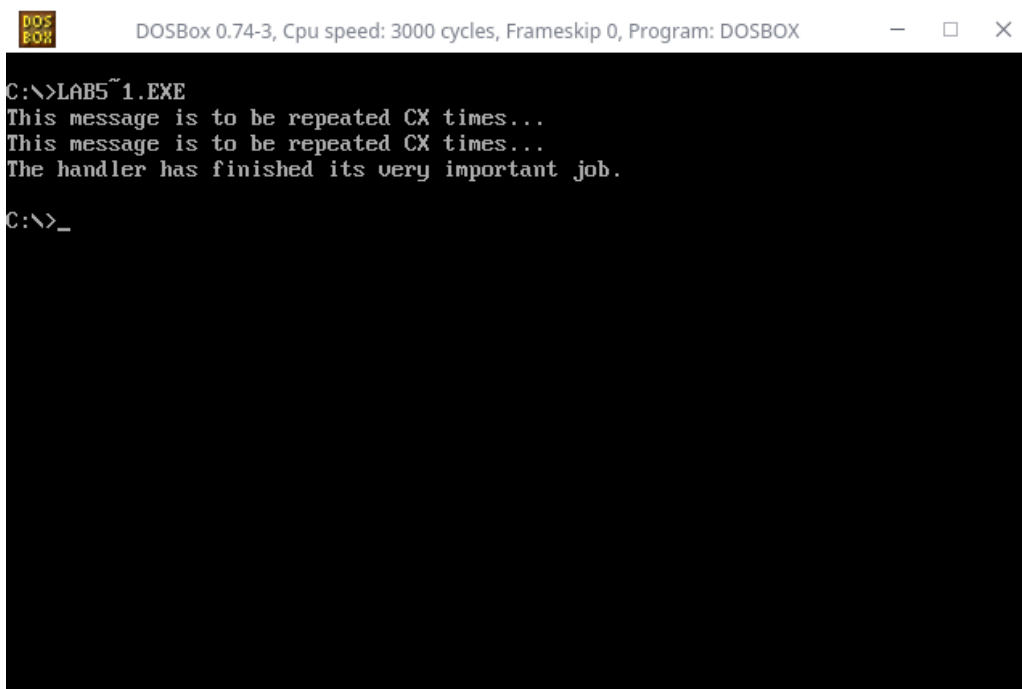
```
DOS
BOX
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>LAB5~1.EXE
The handler has finished its very important job.
C:\>_
```

Рис. 1: Полный вывод программы при выполнении первого теста из Таблицы 1



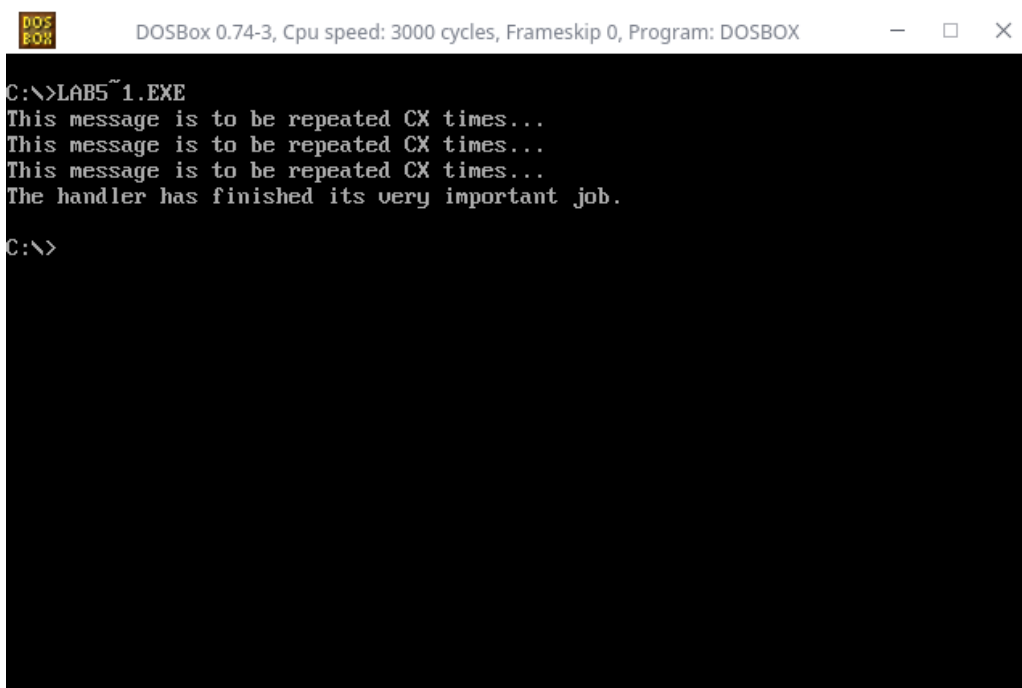
```
DOS
BOX
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>LAB5~1.EXE
This message is to be repeated CX times...
The handler has finished its very important job.
C:\>_
```

Рис. 2: Полный вывод программы при выполнении второго теста из Таблицы 1



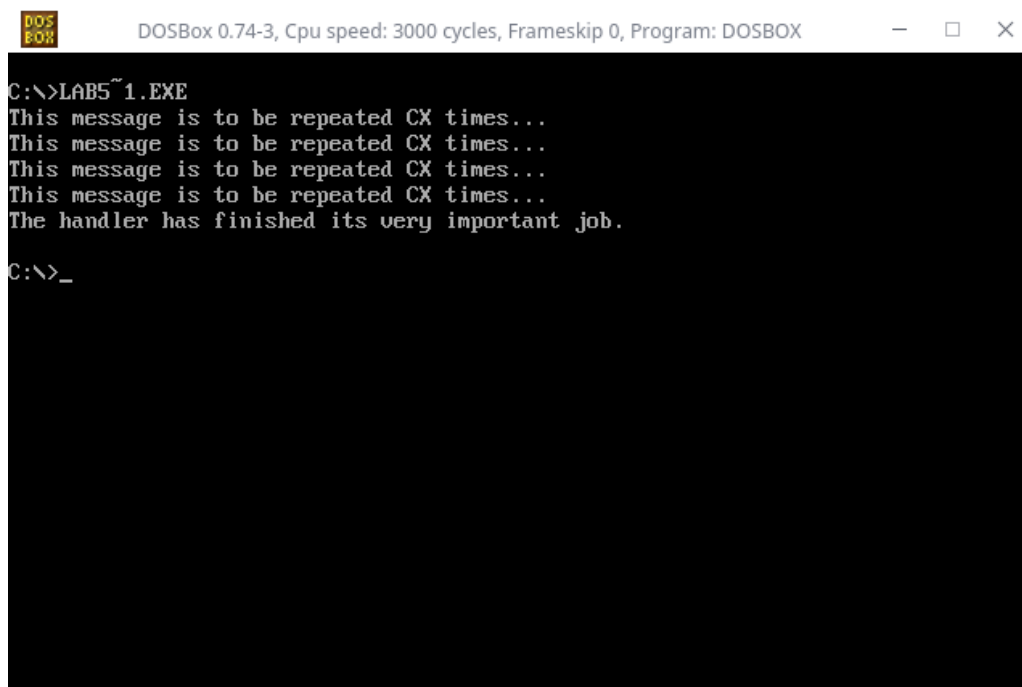
```
DOS
BOX
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>LAB5~1.EXE
This message is to be repeated CX times...
This message is to be repeated CX times...
The handler has finished its very important job.
C:\>_
```

Рис. 3: Полный вывод программы при выполнении третьего теста из Таблицы 1



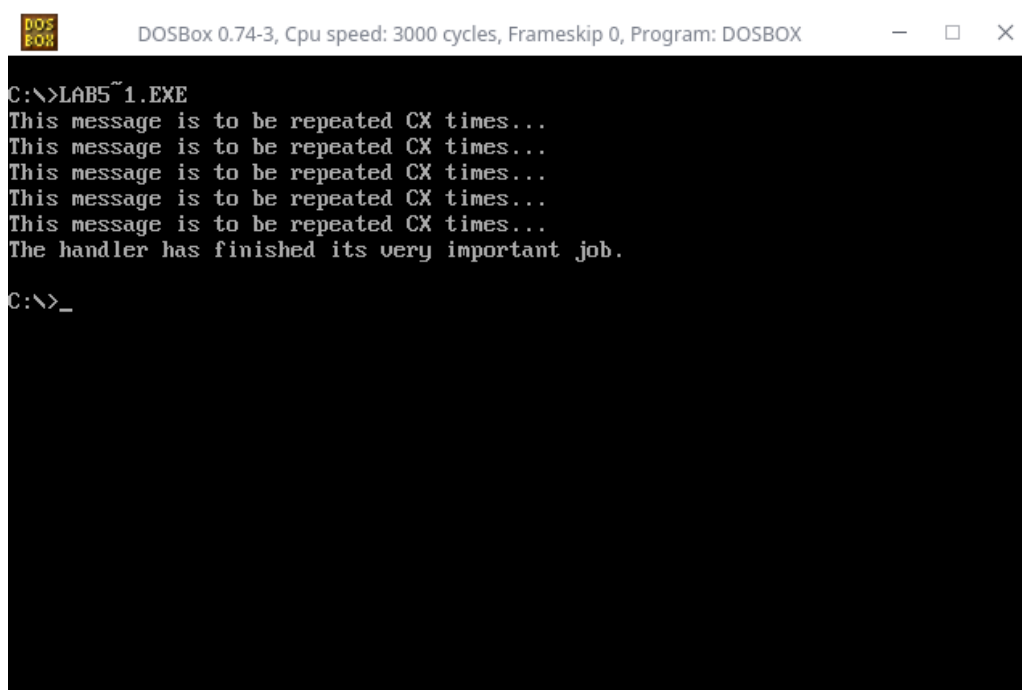
```
DOS
BOX
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>LAB5~1.EXE
This message is to be repeated CX times...
This message is to be repeated CX times...
This message is to be repeated CX times...
The handler has finished its very important job.
C:\>
```

Рис. 4: Полный вывод программы при выполнении четвёртого теста из Таблицы 1



```
DOS FOR      DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>LAB5~1.EXE
This message is to be repeated CX times...
This message is to be repeated CX times...
This message is to be repeated CX times...
This message is to be repeated CX times...
The handler has finished its very important job.
C:\>_
```

Рис. 5: Полный вывод программы при выполнении пятого теста из Таблицы 1



```
DOS FOR      DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>LAB5~1.EXE
This message is to be repeated CX times...
This message is to be repeated CX times...
This message is to be repeated CX times...
This message is to be repeated CX times...
The handler has finished its very important job.
C:\>_
```

Рис. 6: Полный вывод программы при выполнении шестого теста из Таблицы 1