**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине “Архитектура вычислительных систем”

Студент АС-21-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Станиславчук С. М.

(подпись, дата)

Руководитель

Ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Болдырихин О. В.

(подпись, дата)

Липецк 2023

**Цель работы:**

Изучение организации ввода-вывода.

**Задание кафедры: Вариант 27**

Цель работы — изучение порядка функционирования ЭВМ, организация

взаимодействия процессора с памятью и устройствами ввода-вывода.

**Ход работы:**

**1. Листинг**

Turbo Assembler Version 3.1 12/03/23 12:30:54 Page 1

sample.asm

1 0000 .model small

2

3 0000 data\_in segment

4 0000 04\*(00) input db 4 dup(0) ; Массив для ввода данных

5 0004 08\*(00) output db 8 dup(0) ; Массив, содержащий отдельные символы +

6 для каждого числа

7 000C data\_in ends

8

9 0000 data\_out segment

10 0000 00 res1 db 0

11 0001 00 res2 db 0

12 0002 data\_out ends

13

14 0000 stack segment

\*Warning\* sample.asm(13) Reserved word used as symbol: STACK

15 0000 64\*(0000) dw 100 dup(0) ; Stack definition

16 00C8 stack ends

17

18 0000 code segment

19 assume DS:data\_out, ES:data\_in, CS:code, SS:stack

20 ;--------------------------------------------------------

21 0000 near\_conversion proc

22 0000 B8 0000s mov ax, data\_in

23 0003 8E C0 mov es, ax

24 0005 B8 0000s mov ax, data\_out

25 0008 8E D8 mov ds, ax

26

27 000A BE 0000 mov si, 0 ; Initialize index for array traversal

28

29 000D input\_loop:

30 ; Ввод символа с клавиатуры

31 000D B4 01 mov ah, 01h

32 000F CD 21 int 21h

33 0011 2C 30 sub al, '0' ; Convert ASCII to integer

34

35 ; Store the modified value in the input array

36 0013 26: 88 84 0000r mov input[si], al

37

38 0018 46 inc si

39 0019 83 FE 04 cmp si, 4 ; Check if all digits are entered

40 001C 7C EF jl input\_loop

41

42 001E BE 0000 mov si, 0 ; Reset the index for array traversal

43

44

45 ; Convert two-digit numbers and store it in al

46 0021 26: 8A 84 0000r mov al, input[si] ; Load the first digit

47 0026 D0 E0 D0 E0 D0 E0 D0+ shl al, 4 ; Shift left to make room for the second digit

48 E0

49 002E 26: 02 84 0001r add al, input[si + 1] ; Add the second digit

50

51 0033 A2 0000r mov res1, al ; Store the result in res1

52

53

54 ; Move to the next two digits and store it in ah

55 0036 26: 8A A4 0002r mov ah, input[si + 2] ; Load the third digit

56 003B D0 E4 D0 E4 D0 E4 D0+ shl ah, 4 ; Shift left to make room for the fourth digit

Turbo Assembler Version 3.1 12/03/23 12:30:54 Page 2

sample.asm

57 E4

58 0043 26: 02 A4 0003r add ah, input[si + 3] ; Add the fourth digit

59

60 0048 88 26 0001r mov res2, ah ; Store the result in res2

61

62

63 ; Push the modified values (83, 99) to the stack

64 004C 50 push ax

65

66 ; Print the input array for verification

67 004D BE 0000 mov si, 0

68 0050 print\_input:

69 0050 26: 8A 94 0000r mov dl, input[si]

70 0055 80 C2 30 add dl, '0' ; Convert back to ASCII

71 0058 B4 02 mov ah, 02h ; DOS function to print character

72 005A CD 21 int 21h

73

74 005C 46 inc si

75 005D 83 FE 04 cmp si, 4

76 0060 7C EE jl print\_input

77

78 ; Call the far subroutine for further processing

79 0062 9A 00000001sr call far ptr far\_conversion

80

81 0067 C3 ret ; Return from subroutine

82

83 0068 near\_conversion endp

84 ;--------------------------------------------------------

85

86 0068 start:

87 0068 B8 0000s mov ax, stack

88 006B 8E D0 mov ss, ax

89

90 006D E8 FF90 call near\_conversion

91

92 ;--------------------------------------------------------

93 0070 play\_audio:

94 ; загрузка счетчика канала 2 значением 0E24h (нота +

95 "Ми")

96 0070 B0 24 mov al, 24h ; сначала выводится младший байт

97 0072 E6 42 out 42h, al

98 0074 B0 0E mov al, 0Eh ; затем выводится старший байт

99 0076 E6 42 out 42h, al

100 ; включение сигнала и динамика

101 0078 E4 61 in al, 61h

102 007A 0C 03 or al, 00000011b

103 007C E6 61 out 61h, al

104

105 ; выключение сигнала и динамика

106 007E E4 61 in al, 61h

107 0080 24 FC and al, 11111100b

108 0082 E6 61 out 61h, al

109

110 ;--------------------------------------------------------

111

112 0084 B8 4C00 mov ax, 4C00h

113 0087 CD 21 int 21h

Turbo Assembler Version 3.1 12/03/23 12:30:54 Page 3

sample.asm

114

115 0089 code ends

116

117 ;--------------------------------------------------------

118 0000 far\_code segment

119 0000 00 res3 db 0

120 assume CS:far\_code

121 0001 far\_conversion proc far

122 0001 2E: A2 0000r mov res3, al

123 0005 CA 0002 retf 2

124 0008 far\_conversion endp

125 0008 far\_code ends

126 ;--------------------------------------------------------

127

128 end start

**2. Таблица состояния системы**

Составим таблицу состояний системы после выполнения каждой команды (таблица 1)

Таблица 1 – Состояния системы после выполнения команд программы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер команды | Адрес команды | Команда на машинном языке | Регистр  команд | Команда на языке ассемблера | Указатель команд | Содержание изменившихся регистров и ячеек памяти |
| 1 | 0068 | B8BE48 | B8 | mov ax, stack | 006B | ax 48BE |
| 2 | 0006B | 8ED0 | 8ED0 | mov ss, ax | 006D | ss 48BE |
| 3 | 0046 | E890FF | E8 | call near\_conversion | 0000 | sp FFFE ss:sp -> 0070 |
| 4 | 0000 | B8BC48 | B8 | mov ax, data\_in | 0003 | ax 48BC |
| 4 | 0003 | 8EC0 | 8EC0 | mov es, ax | 0005 | es 48BC |
| 5 | 0005 | B8BD48 | B8 | mov ax, data\_out | 0008 | ax 48BD |
| 6 | 0008 | 8ED8 | 8ED8 | mov ds, ax | 000A | ds 48BD |
| 7 | 000A | BE0000 | BE | mov si, 0 | 000D | si 0000 |
| ======INPUT LOOP======  STARTS:  ITERATION 1 | | | | | | |
| 8 | 000D | B401 | B4 | mov ah, 01h | 000F | ax 01BD |
| 9 | 000F | CD21 | CD | int 21h | 0011 | ax 0138 |
| ENTER: 8 | | | | | | |
| 10 | 0011 | 2C30 | 2C | sub al, ‘0’ | 0013 | ax 0108 |
| 11 | 0013 | 2688840000 | 2688840000 | mov input[si], al | 0018 |  |
| 12 | 0018 | 46 | 46 | inc si | 0019 | si 0001 |
| 13 | 0019 | 83FE04 | 83FE | cmp si, 4 | 001C |  |
| 14 | 001C | 7CEF | 7CEF | jl input\_loop | 000D |  |
| ======INPUT LOOP======  ITERATION 2 | | | | | | |
| 15 | 000D | B401 | B4 | mov ah, 01h | 000F |  |
| 16 | 000F | CD21 | CD | int 21h | 0011 | ax 0133 |
| ENTER: 3 | | | | | | |
| 17 | 0011 | 2C30 | 2C | sub al, ‘0’ | 0013 | ax 0103 |
| 18 | 0013 | 2688840000 | 2688840000 | mov input[si], al | 0018 |  |
| 19 | 0018 | 46 | 46 | inc si | 0019 | si 0002 |
| 20 | 0019 | 83FE04 | 83FE | cmp si, 4 | 001C |  |
| 21 | 001C | 7CEF | 7CEF | jl input\_loop | 000D |  |
| ======INPUT LOOP======  ITERATION 3 | | | | | | |
| 22 | 000D | B401 | B4 | mov ah, 01h | 000F |  |
| 23 | 000F | CD21 | CD | int 21h | 0011 | ax 0139 |
| ENTER: 9 | | | | | | |
| 24 | 0011 | 2C30 | 2C | sub al, ‘0’ | 0013 | ax 0109 |
| 25 | 0013 | 2688840000 | 2688840000 | mov input[si], al | 0018 |  |
| 26 | 0018 | 46 | 46 | inc si | 0019 | si 0003 |
| 27 | 0019 | 83FE04 | 83FE | cmp si, 4 | 001C |  |
| 28 | 001C | 7CEF | 7CEF | jl input\_loop | 001D |  |
| ======INPUT LOOP======  ITERATION 4 | | | | | | |
| 29 | 000D | B401 | B4 | mov ah, 01h | 000F |  |
| 30 | 000F | CD21 | CD | int 21h | 0011 | ax 0139 |
| ENTER: 9 | | | | | | |
| 31 | 0011 | 2C30 | 2C | sub al, ‘0’ | 0013 | ax 0109 |
| 32 | 0013 | 2688840000 | 2688840000 | mov input[si], al | 0018 |  |
| 33 | 0018 | 46 | 46 | inc si | 0019 | si 0004 |
| 34 | 0019 | 83FE04 | 83FE | cmp si, 4 | 001C |  |
| 35 | 001C | 7CEF | 7CEF | jl input\_loop | 001E |  |
| ======INPUT LOOP======  ENDS | | | | | | |
| 36 | 001E | BE0000 | BE | mov si, 0 | 0021 | si 0000 |
| 37 | 0021 | 268A840000 |  | mov al, input[si] | 0026 | ax 0108 |
| 38 | 0026 | D0E0 | D0E0 | shl al, 1 | 0028 | ax 0110 |
| 39 | 0028 | D0E0 | D0E0 | shl al, 1 | 002A | ax 0120 |
| 40 | 002A | D0E0 | D0E0 | shl al, 1 | 002C | ax 0140 |
| 41 | 002C | D0E0 | D0E0 | shl al, 1 | 002E | ax 0180 |
| 42 | 002E | 2602840100 | 2602 | add al, input[si+1] | 0033 | ax 0183 |
| 43 | 0033 | A20000 | A2 | mov res1, al | 0036 |  |
| 44 | 0036 | 268AA40200 | 268A | mov ah, input[si+2] | 003B | ax 0983 |
| 45 | 003B | D0E4 | D0E4 | shl ah, 1 | 003D | ax 1283 |
| 46 | 003D | D0E4 | D0E4 | shl ah, 1 | 003F | ax 2483 |
| 47 | 003F | D0E4 | D0E4 | shl ah, 1 | 0041 | ax 4883 |
| 48 | 0041 | D0E4 | D0E4 | shl ah, 1 | 0043 | ax 9083 |
| 49 | 0043 | 2602A40300 | 2602A4 | add ah, input[si+3] | 0048 | ax 9983 |
| 50 | 0048 | 88260100 | 88260100 | mov res2, ah | 004C |  |
| 51 | 004C | 50 | 50 | push ax | 004D | sp FFFC; ss:sp -> 9983 |
| 52 | 004D | BE0000 | BE | mov si, 0 | 0050 |  |
| ======OUTPUT LOOP======  STARTS:  ITERATION 1 | | | | | | |
| 53 | 0050 | 268A940000 | 268A | mov dl, input[si] | 0055 | ax 8408 |
| 54 | 0055 | 80C230 | 80C2 | add dl, ‘0’ | 0058 | dx 8438 |
| 55 | 0058 | B402 | B4 | mov ah, 02h | 005A | ax 0283 |
| 56 | 005A | CD21 | CD | int 21h | 005C | ax 0238 |
| 57 | 005C | 46 | 46 | inc si | 005D | si 0001 |
| 58 | 005D | 83FE04 | 83FE | cmp si, 4 | 0060 |  |
| 59 | 0060 | 7CEE | 7C | jl print\_input | 0050 |  |
| ======OUTPUT LOOP======  ITERATION 2 | | | | | | |
| 60 | 0050 | 268A940000 | 268A | mov dl, input[si] | 0055 | dx 8403 |
| 61 | 0055 | 80C230 | 80C2 | add dl, ‘0’ | 0058 | dx 8433 |
| 62 | 0058 | B402 | B4 | mov ah, 02h | 005A |  |
| 63 | 005A | CD21 | CD | int 21h | 005C | ax 0233 |
| 64 | 005C | 46 | 46 | inc si | 005D | si 0002 |
| 65 | 005D | 83FE04 | 83FE | cmp si, 4 | 0060 |  |
| 66 | 0060 | 7CEE | 7CEE | jl print\_input | 0050 |  |
| ======OUTPUT LOOP======  ITERATION 3 | | | | | | |
| 67 | 0050 | 268A940000 | 268A | mov dl, input[si] | 0055 | dx 8409 |
| 68 | 0055 | 80C230 | 80C2 | add dl, ‘0’ | 0058 | dx 8439 |
| 69 | 0058 | B402 | B4 | mov ah, 02h | 005A |  |
| 70 | 005A | CD21 | CD | int 21h | 005C | ax 0239 |
| 71 | 005C | 46 | 46 | inc si | 005D | si 0003 |
| 72 | 005D | 83FE04 | 83FE | cmp si, 4 | 0060 |  |
| 73 | 0060 | 7CEE | 7CEE | jl print\_input | 0050 |  |
| ======OUTPUT LOOP======  ITERATION 4 | | | | | | |
| 74 | 0050 | 268A940000 | 268A | mov dl, input[si] | 0055 | dx 8409 |
| 75 | 0055 | 80C230 | 80C2 | add dl, ‘0’ | 0058 | dx 8439 |
| 76 | 0058 | B402 | B4 | mov ah, 02h | 005A |  |
| 77 | 005A | CD21 | CD | int 21h | 005C |  |
| 78 | 005C | 46 | 46 | inc si | 005D | si 0004 |
| 79 | 005D | 83FE04 | 83FE | cmp si, 4 | 0060 |  |
| 80 | 0060 | 7CEE | 7CEE | jl print\_input | 0062 |  |
| ======OUTPUT LOOP======  ENDS | | | | | | |
| 81 | 0062 | 9A0100CB48 | 9A01 | call far ptr far\_conversion | 0001 | cs 48CB; sp FFF8; ss:sp -> 0067 |
| 82 | 0001 | 2EA20000 | 2EA2 | mov res3, al | 0005 | cs 48B3; sp FFFE; ss:sp -> 0070 |
| 83 | 0005 | CA0200 | CA02 | retf 2 | 0067 |  |
| 84 | 0067 | C3 | C3 | ret | 0070 | sp 0000; ss:sp -> 0000 |
| ======AUDIO PLAY ALGORITHM======  STARTS | | | | | | |
| 85 | 0070 | B024 | B0 | mov al, 24h | 0072 | ax 0224 |
| 86 | 0072 | E642 | E6 | out 42h, al | 0074 |  |
| 87 | 0074 | B00E | B0 | mov al, 0Eh | 0076 | ax 020E |
| 88 | 0076 | E642 | E6 | out 42h, al | 0078 |  |
| 89 | 0078 | E461 | E4 | in al, 61h | 007A | ax 0230 |
| 90 | 007A | 0C03 | 0C | or al, 00000011b | 007C | ax 0233 |
| 91 | 007C | E661 | E6 | out 61h, al | 007E |  |
| 92 | 007E | E461 | E4 | in al, 61h | 0080 | ax 0231 |
| 93 | 0080 | 24FC | 24FC | and al, 11111100b | 0082 | ax 0230 |
| 94 | 0082 | E661 | E6 | out 61h, al | 0084 | cs 48B3 |
| ======AUDIO PLAY ALGORITHM====== ENDS | | | | | | |
| 95 | 0084 | B8004C | B8 | mov ax, 4C00h | 0087 | ax 4C00; cs 48B3 |
| 96 | 0087 | CD21 | CD | int 21h | - | - |

**3. Проверка работы алгоритма на правильных числах**

Упакованный двоично-десятичный код (Packed Binary Coded Decimal, PBCD) - это способ представления десятичных чисел в формате, где каждая десятичная цифра представлена в виде 4-битного двоичного числа. В упакованном PBCD каждая десятичная цифра (0-9) кодируется с использованием 4 битов, и эти коды объединяются вместе, чтобы представить десятичное число.

В ближней подпрограмме есть цикл, который считывает четыре однозначных числа и записывает их в сегмент ES, в следующем цикле объединяет первые два символа и вторые два символа в результате чего получается два двузначных числа: 83 и 99. После этого происходит перевод и склеивание битов этих чисел с последующим занесением результата в переменную res1 и res2, которые находится в сегменте ES, выводим эти значения в цикле print\_input (так как по логике программы мы должны вывести преобразованные результаты (HEX), это как раз те значения, которые мы подали на входе (DEC) и ожидаем увидеть на выходе: 83h и 99h). А затем этот результат в дальней подпрограмме заносим в сегмент DS, переменную res3. После завершения ближней подпрограммы производим кратковременное проигрывание ноты си. Пример работы программы представлен на рисунке 1.

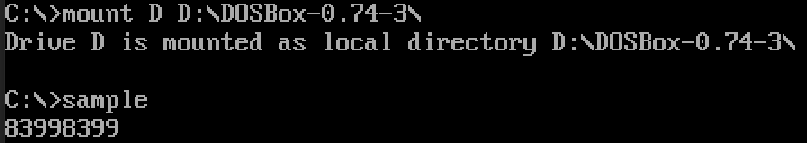


Рисунок 1 – Ввод входных значений “8399” с клавиатуры и вывод результата “8399” на дисплей в консоли Turbo Debugger (sample – название .asm файла)

**5. Вывод**

В ходе выполненной работы ознакомился организацией ввода-вывода значений, научился проигрывать звук.