Московский государственный технический университет им. Н.Э.Баумана

|  |  |  |
| --- | --- | --- |
| Защищено:  Большаков С.А.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |  | Демонстрация ЛР:  Большаков С.А.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

**Отчет по лабораторной работе № 5 по курсу**

**Системное программирование**

#### "Ввод/вывод в адреса и числа"

#### (есть ли дополнительные требования - НЕТ)

8

(количество листов)

Вариант № 11

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы **ИУ5-41Б** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| **Ларкин Б. В.** | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

Москва, МГТУ - 2024

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ**

[1. Цель выполнения лабораторной работы № 4 3](#_Toc162230442)

[2. Порядок и условия проведения работы № 4 3](#_Toc162230443)

[3. Описание ошибок, возникших при отладке № 4 3](#_Toc162230444)

[4. Блок-схема программы 4](#_Toc162230445)

[5. Текст программы на языке Ассемблера (.LST) 5](#_Toc162230446)

[6. Скриншот программы в TD.exe 7](#_Toc162230447)

[7. Результаты работы программы 7](#_Toc162230448)

[8. Выводы по ЛР № 4 8](#_Toc162230449)

# Цель выполнения лабораторной работы № 5

Лабораторная работа №4 выполняется для получения навыков разработки циклических программ и процедур на Ассемблере, получения знаний о перекодировке символов в среде.

# Порядок и условия проведения работы № 5

Разработать и отладить циклическую программу на языке Ассемблер для вывода на экран **20** последовательных прописных букв русского алфавита (начиная с символа “А” или другого символа, введенного с клавиатуры. Символы должны быть представлены в символьном (печатном) и шестнадцатеричном представлении (через черточку) в виде столбчатой таблицы (см. ниже). Каждая буква выводится в виде ее символьного представления и его 2-х разрядного шестнадцатеричного числа на одной строке. Например (СИМВОЛ – Шестнадцатеричный код):

**А – 80h**.

**Б – 81h**.

**В – 82h**.

**Г – 83h**.

**…**

В программе должна быть выполнена автоматическая шестнадцатеричная перекодировка, на основе преобразования машинного представления кода символа.

Шестнадцатеричная перекодировка (перевод одного представления в другое) должна выполняться командой **XLAT** по специальной таблице перекодировки вида: 0123456789ABCDEF. Переведенные представления русских букв выводятся на экран дисплея последовательно. В каждой строке выводиться только одна буква с переводом (например, "**А – 80h**" – пример для кодировки ДОС - ASCII). Для организации цикла использовать команду **LOOP**. Разработать блок-схему программы. Использовать MS VISIO для блок-схемы или другой доступный графический редактор.

После завершения вывода таблицы нужно организовать ожидание ввода нового символа с клавиатуры для вывода новой таблицы (процедура - **GETCH**). Если вводиться заранее предопределенный символ (например, символ “**\***”), то программа должна завершаться с сообщением о своем завершении. В противном случае циклически выводиться новая таблица для нового введенного символа. В программе разработать и использовать **четыре** отдельные процедуры:

* для ввода символа(без эха) (1 - **GETCH**),
* вывода одного символа (2 - **PUTCH**),
* для перевода буквы в двух символьное шестнадцатеричное представление (3-я процедура **HEX**) и перевода строки и возврата “каретки” экрана дисплея (4 - **CLRF**) и
* для очистки экрана (процедура - **CLSSCR**).

Выход из программы выполнить посредством прерывания 21H - 04CH после нажатия любой клавиши, с заданием кода завершения – 5.

# Описание ошибок, возникших при отладке № 5

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Проявление ошибки | Причина ошибки | Способ устранения |
|  | CLRSCR не очищает полностью экран консоли | Переключение режима работы с одного на него же не происходит | Добавление CLRF в тело PRINTCODES позволяет вывести 20 записей на экран без чего-либо лишнего |
|  | Отсутствие функциональности STOSB | Отсутствие включения регистра ES в работу программы | Добавление PUSH ES для задания регистра и включения STOSB |
|  | Вывод мусора вместо надлежащих строк | Недостаточное число символов в инициализации String | Добавление 12 пробелов и ‘$’ для последующего сохранения законченной строки |

# Блок-схема программы

# 5. Текст программы на языке Ассемблера (.LST)

Turbo Assembler Version 3.1 03/25/24 00:44:43 l4.asm

1 0000 MYCODE SEGMENT 'CODE'

2 ;ЛР#4 2024 ЛАРКИН ИУ5-41Б

3 ASSUME CS:MYCODE

4 0000 20 20 20 20 20 20 20+ String db ' $',0

5 20 20 20 20 24 00

6 000D 30 31 32 33 34 35 36+ hext DB '0123456789ABCDEF'

7 37 38 39 41 42 43 44+

8 45 46

9 001D 80 LET DB 'А'

10 ;Символ, по которому будет производится выход

11 001E 2A BREAK\_SYMBOL DB '\*'

12

13 ;ПЕРЕВОД СИМВОЛА В HEX из AL

14 001F HEX PROC

15 001F BF 0000r LEA DI, String

16 0022 BB 000Dr MOV BX, OFFSET hext

17 0025 50 PUSH AX

18 0026 D0 E8 D0 E8 D0 E8 D0+ SHR AL, 4

19 E8

20 002E D7 XLAT

21 002F AA STOSB

22 0030 58 POP AX

23 0031 50 PUSH AX

24 0032 24 0F AND AL, 00001111b

25 0034 D7 XLAT

26 0035 AA STOSB

27 0036 58 POP AX

28 0037 B8 0068 MOV AX, 'h'

29 003A AB STOSW

30 003B B4 09 MOV AH, 09h

31 003D BA 0000r LEA DX, STRING

32 0040 C3 RET

33 0041 HEX ENDP

34

35 0041 CLSSCR PROC

36 0041 B4 00 MOV AH, 0H

37 0043 CD 10 INT 10H

38 0045 C3 RET

39 0046 CLSSCR ENDP

40

41 0046 PUTCH PROC

42 0046 B4 02 MOV AH, 02H

43 0048 CD 21 INT 21H

44 004A C3 RET

45 004B PUTCH ENDP

46

47 004B CLRF PROC

48 004B B2 0A MOV DL, 10

49 004D E8 FFF6 CALL PUTCH

50 0050 B2 0D MOV DL, 13

51 0052 E8 FFF1 CALL PUTCH

52 0055 C3 RET

53 0056 CLRF ENDP

54

55 ;Переносит введенный символ в AL

56 0056 GETCH PROC

57 0056 B4 08 MOV AH, 08H

58 0058 CD 21 INT 21H

59 005A C3 RET

60 005B GETCH ENDP

61

62 ;Выводит таблицу для предварительно заданного LET

63 005B PRINTCODES PROC

64 005B looppr:

65 005B 2E: 8A 16 001Dr MOV DL, LET

66 0060 E8 FFE3 CALL PUTCH

67 0063 B2 2D MOV DL, '-'

68 0065 E8 FFDE CALL PUTCH

69 0068 2E: A0 001Dr MOV AL, LET

70 006C E8 FFB0 CALL HEX

71 006F CD 21 INT 21H

72 0071 E8 FFD7 CALL CLRF

73 0074 2E: 80 06 001Dr 01 ADD LET, 1

74 007A E2 DF loop looppr

75 ;2 \n, так как CLRSCR не чистит экран полностью

76 007C E8 FFCC CALL CLRF

77 007F E8 FFC9 CALL CLRF

78 0082 C3 RET

79 0083 PRINTCODES ENDP

80

81 0083 START:

82 ; Загрузка сегментного регистра данных DS

83 0083 0E PUSH CS

84 0084 1F POP DS

85 0085 1E PUSH DS

86 0086 07 POP ES

87 0087 06 PUSH ES

88 ; Вывод символов на экран

89 0088 E8 FFB6 CALL CLSSCR

90 008B loopmain:

91 008B B9 0014 MOV CX,20

92 008E E8 FFC5 CALL GETCH

93 0091 2E: 3A 06 001Er CMP AL, BREAK\_SYMBOL

94 0096 74 0C JZ term

95 0098 2E: A2 001Dr MOV LET, AL

96 009C E8 FFA2 CALL CLSSCR

97 009F E8 FFB9 CALL PRINTCODES

98 00A2 E2 E7 loop loopmain

99 ; Выход из программы

100 00A4 term:

101 ;Выход с кодом 5

102 00A4 B0 05 MOV AL, 5

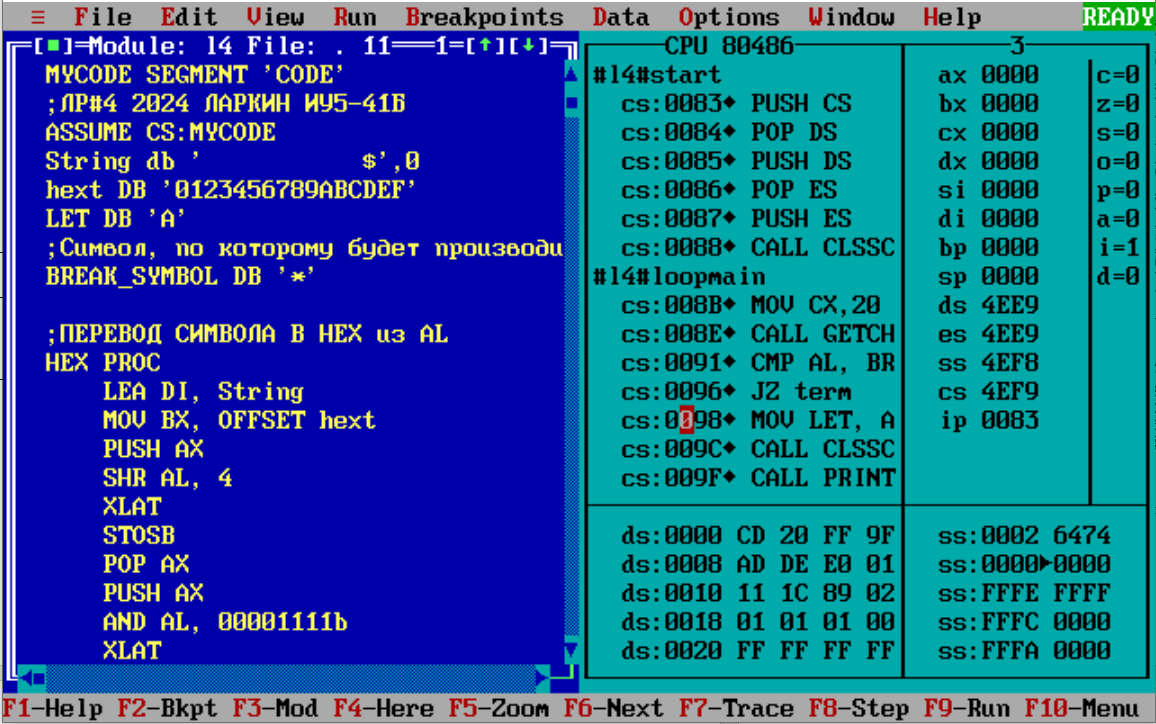
103 00A6 B4 4C MOV AH, 4CH

104 00A8 CD 21 INT 21H

105 00AA MYCODE ENDS

106 END START

# 6. Скриншот программы в TD.exe



# 7. Результаты работы программы

C клавиатуры введен символ ‘a’:

a-61h

b-62h

c-63h

d-64h

e-65h

f-66h

g-67h

h-68h

i-69h

j-6Ah

k-6Bh

l-6Ch

m-6Dh

n-6Eh

o-6Fh

p-70h

q-71h

r-72h

s-73h

t-74h

Введен символ ‘w’:

w-77h

x-78h

y-79h

z-7Ah

{-7Bh

|-7Ch

}-7Dh

~-7Eh

-7Fh

А-80h

Б-81h

В-82h

Г-83h

Д-84h

Е-85h

Ж-86h

З-87h

И-88h

Й-89h

К-8Ah

Введен символ ‘\*’:



# 8. Выводы по ЛР № 5

Разработан файл .ASM и соответствующие файлы приложения и листинга на языке Ассемблер. Программа выполняется в циклическом режиме до ввода ‘\*’, выводя по каждому нажатию клавиши 20 символов на экран в виде “Символ-16-ричный код”, где первый из 20 символов – введенный, остальные 19 – следующие за первым по возрастанию кодировки на 1. Программа работает корректно, мы изучили циклы и перекодировку символов.