Московский государственный технический университет им. Н.Э.Баумана

Защищено:

Большаков С.А.

Демонстрация ЛР:

Большаков С.А.

24 апреля 2023 г.

Отчет по лабораторной работе № 5 по курсу Системное программирование

" Ввод/вывод в адреса и числа "

(есть ли дополнительные требования- ДА/НЕТ)

9
(количество листов)
Вариант № <**10**>

ИСПОЛНИТЕЛЬ:

студент группы **ИУ5-41Б** _____
(подпись)

Никулин Д.Д.

" "_____2023 г.

Москва, МГТУ - 2023

СОДЕРЖАНИЕ

1.Цель выполнения лабораторной работы № 5

Разработать и отладить программу на языке Ассемблер для ввода и буферизации строки символов с клавиатуры (последовательности символов) и затем последовательного их вывода на экран в шестнадцатеричном представлении (через пробел). В данной программе для корректной работы необходимо предусмотреть запоминание строки символов в байтовом массиве. Программа и блок-схема должны содержать вложенные циклы (двойные циклы). Программу оформить в виде исполнимого *.EXE файла.

2. Порядок и условия проведения работы № 5

Признак завершения ввода отдельной строки с клавиатуры – это символ "**$**" (он вводиться с клавиатуры для завершения ввода строки). Между введенной строкой символов и их шестнадцатеричным представлением должен располагаться знак равенства (“=”). Максимальное число вводимых символов не должно превышать 20-ти. В данной программе цикл ввода (с клавиатуры) организуется с помощью команд условного (JE, JNE) перехода и команды безусловного перехода (JMP). После завершения ввода строки выполняется ее автоматический вывод. Организовать цикл ввода строк до ввода специального символа ('*'). Пример результата работы одного цикла программы показан ниже:

```
АБВ$ = 80 81 82
```

Требования к процедурам и их именованию совпадают с требованием предыдущих ЛР. Программа должна работать в циклическом режиме ввода строк (для внешнего цикла используется команда **LOOP**): после ввода одной строки запрашивается следующая (максимальное число вводимых строк для одного запуска программы равно **10**). Завершение цикла ввода строк может быть выполнено при вводе символа звездочка (“*****”), который должен быть введен в первой позиции строки. Вводимые символы строки записываются в символьный массив (буфер символов), максимальное число введенных символов равно 20-ти. Цикл ввода строки организуется командами условного и безусловного перехода. При вводе нужно подсчитать число введенных символов, включая символ доллара (“**$**”). Для вывода организуется цикл с помощью команды цикла (**LOOP**). В программе использовать процедуры предыдущих лабораторных данного цикла (ввода символа, печати, перевода строки и др.).

Для ввода/вывода строки и ее шестнадцатеричного представления разрабатываются дополнительная процедура HEX (см. ЛР №4). Организовать очистку экрана до начала работы программы, а также после ее завершения (С помощью специальной процедуры - CLRSCR).

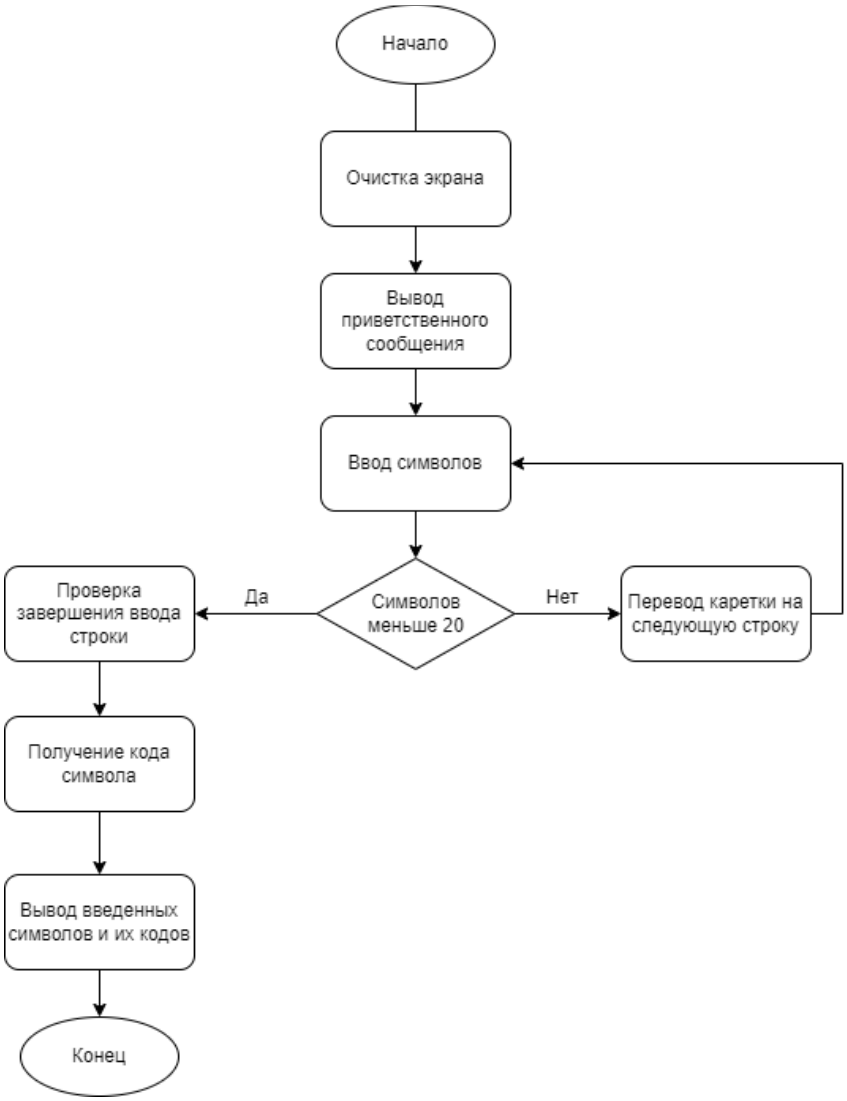
В данной программе необходимо отдельно объявить отдельно сегмент данных (**DTSEG**) и сегмент стека (**STSEG**). Проверить загрузку сегментного регистра данных (**DS**) с помощью команды пересылки (**MOV)** , но через промежуточный регистр (**AX**).

Оформить отчет по ЛР. Для оформления отчета студент должен знать или найти способ для вывода результата работы программы в текстовый файл. Лучше использовать копирование текста из окна командной строки (нежелательно снимать графическую картинку с экрана).

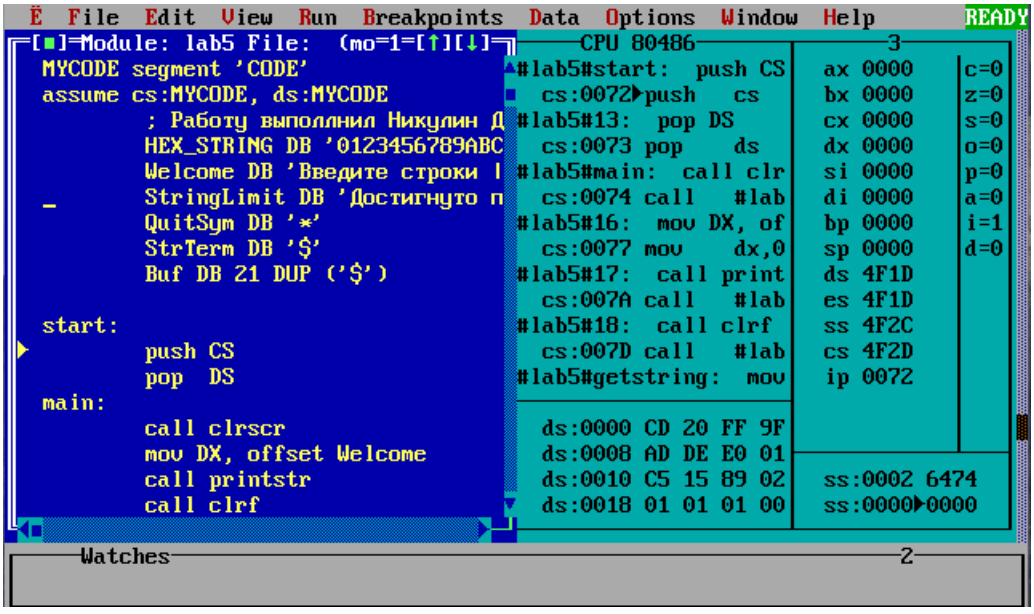
3. Описание ошибок, возникших при отладке № 5

№ п/п	Проявление ошибки	Причина ошибки	Способ устранения
1.	Файл first.exe не открывался при запуске TD	Символьная отладочная информация не была включена в модули	Прописать режимы “/zi” и “/v”
2.	first.exe не запускается в командной строке	Не совместимость с 64-разрядной версией Windows	Использование эмулятора DOSBox

4.Блок-схема программы



5.Скриншот программы в TD.exe



6.Текст программы на языке Ассемблера


```
1    0000                       MYCODE segment 'CODE'
2                               assume cs:MYCODE, ds:MYCODE
3                                       ; Работу выполлнил  Никулин         Данила ИУ5-41Б
4    0000  30 31 32 33 34 35   36+       HEX_STRING          DB '0123456789ABCDEF'
5          37 38 39 41 42 43   44+
6          45 46
7    0010  82 A2 A5 A4 A8 E2         A5+       Welcome DB 'Введите строки | Нажмите * для выхода$'
8          20 E1 E2 E0 AE AA   A8+
9          20 7C 20 8D A0 A6   AC+
10         A8 E2 A5 20 2A 20   A4+
11         AB EF 20 A2 EB E5   AE+
12         A4 A0 24
13   0036  84 AE E1 E2 A8 A3         AD+       StringLimit DB 'Достигнуто    предельное число символов$'
14         E3 E2 AE 20 AF E0   A5+
15         A4 A5 AB EC AD AE         A5+
16         20 E7 A8 E1 AB AE   20+
17         E1 A8 AC A2 AE AB AE+
18         A2 24
19   005B  2A                          QuitSym DB  '*'
20   005C  24                          StrTerm DB  '$'
21   005D  15*(24)                     Buf DB 21 DUP ('$')
22
23   0072                       start:
24   0072  0E                          push CS
25   0073  1F                          pop  DS
26   0074                       main:
27   0074  E8 00A1                     call clrscr
28   0077  BA 0010r                    mov DX, offset Welcome
29   007A  E8 0081                     call printstr
30   007D  E8 008D                     call clrf
31
32   0080                       GetString:
33   0080  BE 0000                     mov SI, 0
34   0083  BB 005Dr                    lea BX, Buf
35
36                                     ;          check 1 sym for being *
37   0086  E8 007F                     call getch
38   0089  89 00                       mov BX[SI], AX
39
40   008B  3A 06 005Br                 cmp AL, QuitSym
41   008F  74 67                       je     Exit
42   0091  3C 24                       cmp AL, '$'
43   0093  74 1E                       je     PrintString
44
45                                     ;          if     not * || $ => print
46   0095  8B D0                       mov DX, AX
47   0097  E8 0069                     call putch
48   009A  46                          inc SI
49
50   009B                       GetSym:
51                                     ;          read sym by sym
52   009B  E8 006A                     call getch
53   009E  89 00                       mov BX[SI], AX
54
55   00A0  3A 06 005Cr                 cmp AL, StrTerm
56   00A4  74 0D                       je     PrintString
57
```

```
00A6  8B D0                     mov DX, AX
00A8  E8 0058                   call putch

00AB  83 FE 13                  cmp SI, 19
00AE  74 35                     je      strlim

                                ;       loop back
00B0  46                        inc SI
00B1  EB E8                     jmp GetSym

00B3                    PrintString:
                                ;       empty line      guard
00B3  8B 07                     mov AX, [BX]
00B5  3C 24                     cmp AL, '$'
00B7  74 27                     je      Handler$

00B9  BA 0020                   mov DX, 32
00BC  E8 0044                   call putch
00BF  BA 003D                   mov DX, '='
00C2  E8 003E                   call putch

00C5                    PrintHex:
                                ;       output sym      by sym
00C5  33 F6                     xor SI, SI
00C7                    PrintHexSym:
                                ;       endline check '$'
00C7  8B 00                     mov AX, BX[SI]
00C9  3C 24                     cmp AL, '$'
00CB  74 13                     je      Handler$
                                ;       print space ' ' between symbols
00CD  BA 0020                   mov DX, 32
00D0  E8 0030                   call putch
                                ;       print hex from lab4
00D3  8B 00                     mov AX, BX[SI]
00D5  53                        push BX
00D6  BB 0000r                  mov BX, offset HEX_STRING
00D9  E8 0043                   call hex
00DC  5B                        pop BX

                                ;       cycle back
00DD  46                        inc SI
00DE  EB E7                     jmp PrintHexSym

00E0                    Handler$:
00E0  E8 002A                   call clrf
00E3  EB 9B                     jmp GetString

00E5                    strlim:
00E5  B8 0024                   mov AX, '$'
00E8  89 00                     mov BX[SI], AX
00EA  E8 0020                   call clrf
00ED  BA 0036r                  mov DX, offset StringLimit
00F0  E8 000B                   call printstr
00F3  E8 0017                   call clrf
00F6  74 CD                     je      PrintHex

00F8                    Exit:
```

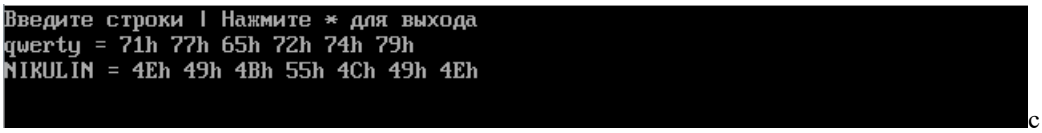
```
                                        ;call clrscr
00F8  B0 00                             mov al, 0
00FA  B4 4C                             mov ah, 4ch
00FC  CD 21                             int 021h

                                ; print string
00FE                                    printstr proc
00FE  B4 09                                     mov ah, 09h
0100  CD 21                                     int 021h
0102  C3                                        ret
0103                                    printstr endp

0103                                    putch proc
0103  B4 02                                     mov ah, 02h
0105  CD 21                                     int 021h
0107  C3                                        ret
0108                                    putch endp

0108                                    getch proc
0108  B4 08                                     mov ah, 08h
010A  CD 21                                     int 021h
010C  C3                                                ret
010D                                    getch endp

                                        ; /n/r
010D                                    clrf proc
010D  B2 0A                                     mov dl, 10
010F  E8 FFF1                                   call putch
0112  B2 0D                                     mov dl, 13
0114  E8 FFEC                                   call putch
0117  C3                                        ret
0118                                    clrf endp

                                ; Clean Srcreen
0118                                    clrscr proc
0118  B4 00                                     mov ah, 00h
011A  B0 02                                     mov al, 02
011C  CD 10                                     int 10h
011E  C3                                        ret
011F                                    clrscr endp

011F                                    hex proc
011F  50                                                push AX
0120  D0 E8 D0 E8 D0 E8         D0+                             shr al, 4
      E8
0128  D7                                                xlat
0129  8A D0                                             mov dl, al
012B  E8 FFD5                                   call putch

012E  58                                                pop ax
012F  24 0F                                     and al, 00001111b
0131  D7                                                xlat
0132  8A D0                                             mov dl, al
0134  E8 FFCC                                   call putch
0137  BA 0068                                   mov dx, 104 ; h
013A  E8 FFC6                                   call putch
013D  C3                                        ret
```

```
172  013E                    hex endp
173
174  013E                MYCODE ends
175                      end start
```


Symbol Name	Type	Value	Cref (defined at #)
??DATE	Text	"04/10/23"	
??FILENAME	Text	"LAB5 "	
??TIME	Text	"11:47:13"	
??VERSION	Number	030A	
@CPU	Text	0101H	
@CURSEG	Text	MYCODE	#1
@FILENAME	Text	LAB5	
@WORDSIZE	Text	2	#1
BUF	Byte	MYCODE:005D	#21 34
CLRF	Near	MYCODE:010D	30 102 108 111 #140
CLRSCR	Near	MYCODE:0118	27 #149
EXIT	Near	MYCODE:00F8	41 #114
GETCH	Near	MYCODE:0108	37 52 #133
GETSTRING	Near	MYCODE:0080	#32 103
GETSYM	Near	MYCODE:009B	#50 66
HANDLER$	Near	MYCODE:00E0	72 86 #101
HEX	Near	MYCODE:011F	94 #156
HEX_STRING	Byte	MYCODE:0000	#4 93
MAIN	Near	MYCODE:0074	#26
PRINTHEX	Near	MYCODE:00C5	#79 112
PRINTHEXSYM	Near	MYCODE:00C7	#82 99
PRINTSTR	Near	MYCODE:00FE	29 110 #121
PRINTSTRING	Near	MYCODE:00B3	43 56 #68
PUTCH	Near	MYCODE:0103	47 59 75 77 89 #127 142 144 162 168 170
QUITSYM	Byte	MYCODE:005B	#19 40
START	Near	MYCODE:0072	#23 175
STRINGLIMIT	Byte	MYCODE:0036	#13 109
STRLIM	Near	MYCODE:00E5	62 #105
STRTERM	Byte	MYCODE:005C	#20 55
WELCOME	Byte	MYCODE:0010	#7 28

Groups & Segments	Bit Size	Align	Combine	Class	Cref (defined at #)
MYCODE	16	013E	Para none	CODE	#1 2 2

7. Результаты работы программы



c

8.Выводы по ЛР № 5

По результату выполнения лабораторной работы №5 была разработана и отлажена программа на языке Ассемблер для ввода и буферизации строки символов с клавиатуры (последовательности символов) и затем последовательного их вывода на экран в шестнадцатеричном представлении (через пробел).