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 № 7 по курсу Системное программирование

" Ввод, вывод и перевод адреса "

(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 № 7

Разработать и отладить программу на языке Ассемблер для ввода с клавиатуры четырехразрядного шестнадцатеричного числа – символами (короткого адреса NEAR) в машинном шестнадцатеричном. Полученное значение выводится затем на экран также в шестнадцатеричном представлении, но заново переведенное из машинного формата. Кроме того, выполняется перевод по схеме Горнера в десятичное представление и на экран выводится в десятичном формате.

2. Порядок и условия проведения работы № 7

Между введенным символьным значением адреса и выводимым шестнадцатеричным представлением должен располагаться знак равенства ("="), а между – формируемыми представлениями пробел (шестнадцатеричным и десятичным).

Например (сначала машинное - 00FEh ,а затем десятичное - 254): Введите число(длинный адрес: HHHH:HHHH)>00FE=00FEh 254

>...

>*

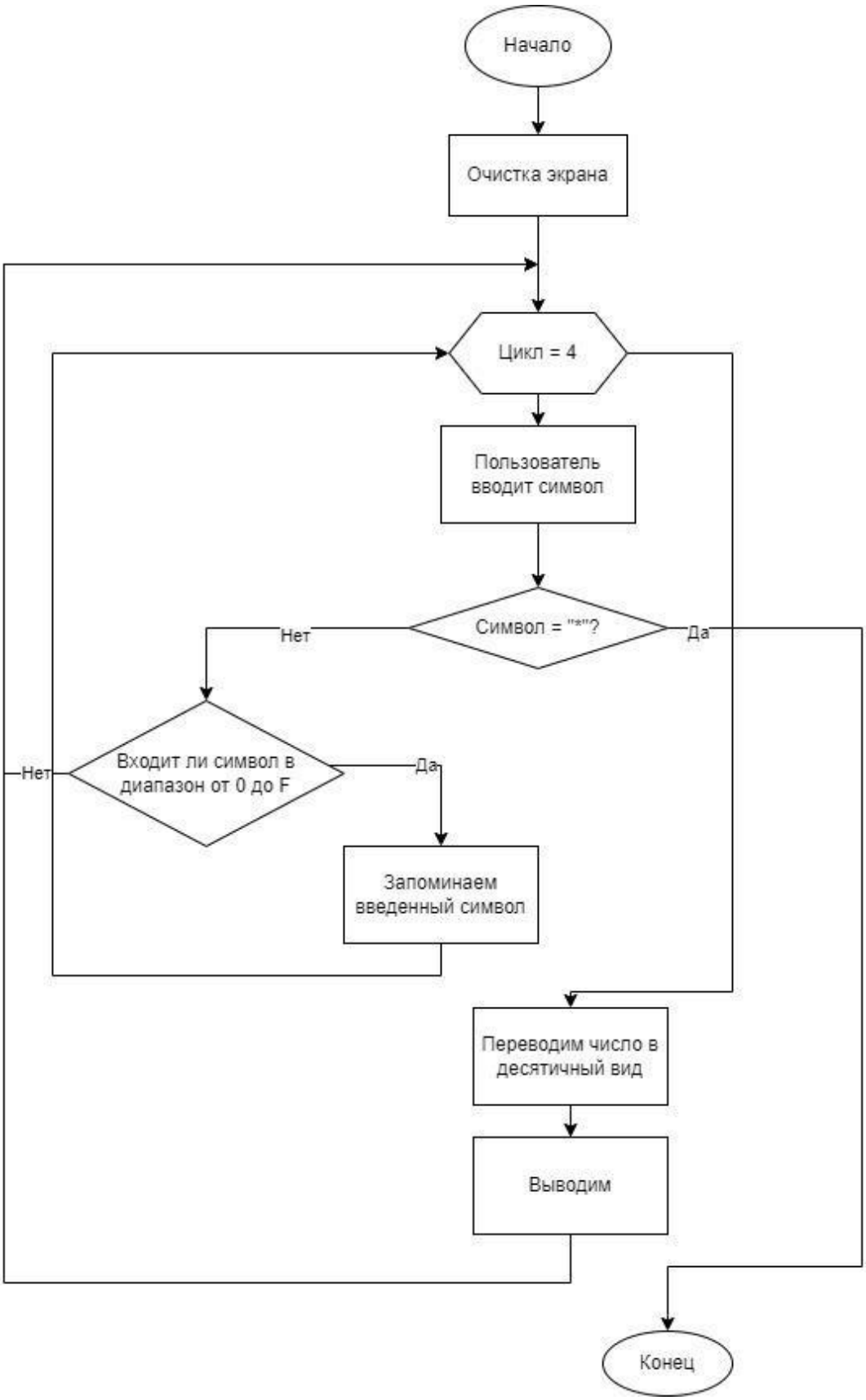
Завершение ввода чисел!

Программа должна работать в циклическом режиме, то есть после ввода одного числа, запрашивается ввод нового. Завершение цикла ввода чисел выполняется по знаку “*” в первой позиции строки ввода. Для ввода и перевода должны быть использованы базовые процедуры. При вводе необходимо проверять вводимые шестнадцатеричные символы (0-9 и A -F)/ Нужно организовать очистку экрана до начала работы программы, и после ее завершения.

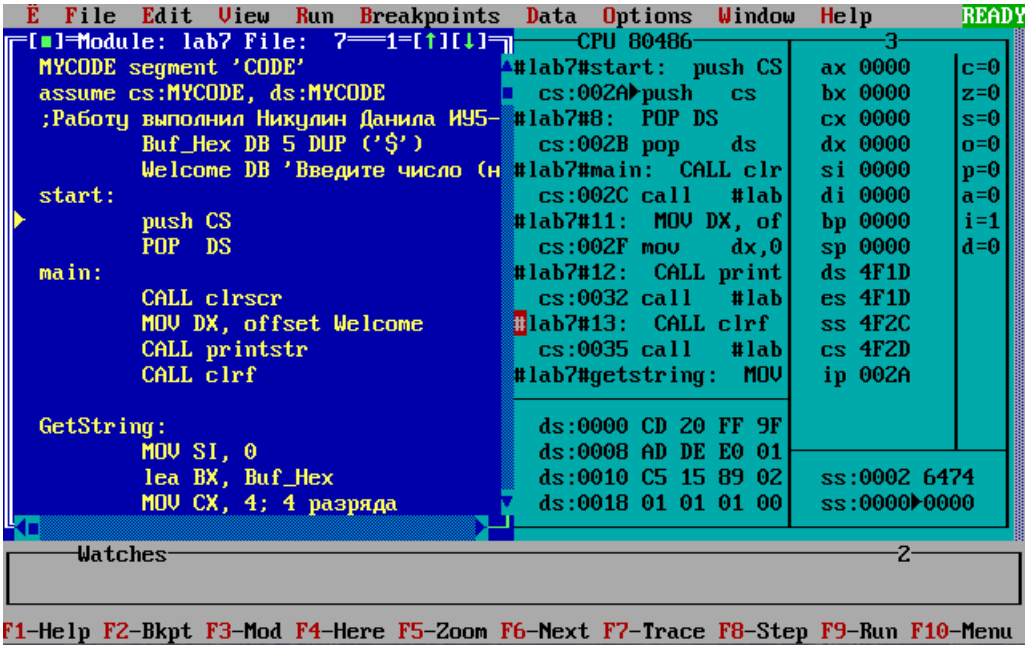
3. Описание ошибок, возникших при отладке № 7

№ п/п	Проявление ошибки	Причина ошибки	Способ устранения
1.	Пользователь вводит числа, превышающие 4-разрядный диапазон и программа обрабатывает только первые 4 цифры, игнорируя оставшиеся	Не было введено ограничений на ввод пользователем символов	Пыыри вводе пользователем 4 символов, сразу жепереходим на следующую строку

4.Блок-схема программы



5.Скриншот программы в TD.exe



6.Текст программы на языке Ассемблера

Turbo Assembler Version 3.1 04/24/23 11:20:41 Page 1
LAB7.ASM

```
 1    0000                    MYCODE segment 'CODE'
 2                            assume cs:MYCODE, ds:MYCODE
 3                            ;Работу выполнил Никулин Данила ИУ5-41Б Лабараторная работа №7
 4    0000  05*(24)                   Buf_Hex DB  5 DUP ('$')
 5    0005  82 A2 A5 A4 A8 E2      A5+         Welcome DB 'Введите число (нажмите          * для выхода)$'
 6          20 E7 A8 E1 AB AE   20+
 7          28 AD A0 A6 AC A8 E2+
 8          A5 20 2A 20 A4 AB   EF+
 9          20 A2 EB E5 AE A4   A0+
10          29 24
11    002A                    start:
12    002A  0E                        push CS
13    002B  1F                        POP  DS
14    002C                    main:
15    002C  E8 00D6                   CALL clrscr
16    002F  BA 0005r                  MOV DX, offset Welcome
17    0032  E8 00B6                   CALL printstr
18    0035  E8 00C2                   CALL clrf
19
20    0038                    GetString:
21    0038  BE 0000                   MOV SI, 0
22    003B  BB 0000r                  lea BX, Buf_Hex
23    003E  B9 0004                   MOV CX, 4;  4 разряда
24    0041  E8 00B6                   CALL clrf
25    0044                    GetSym:
26    0044  33 C0                     xor AX, AX; Очищаем
27    0046  E8 00AC                   CALL getch
28    0049  3C 2A                     cmp AL, '*'; Выходим если символ *
29    004B  74 1F                     je      Exit
30    004D                    Check_if_in_hex: ;      Выходим         если символ не подходит
      под требования
31    004D  3C 46                           cmp AL, 'F'
32    004F  7F F3                           jg      GetSym
```

```
   33    0051  3C 30                          cmp AL, '0'
   34    0053  7C EF                          jl      GetSym
   35    0055  EB 7C 90                 jmp Remember_numbers
   36    0058                           AfterCheck:
   37    0058  88 00                    MOV BX[SI], AL
   38    005A  8A D0                    MOV DL,AL
   39    005C  E8 0091                  CALL putch
   40    005F  83 C6 01                 ADD SI,1
   41    0062  E2 E0                    LOOP GetSym
   42    0064  B2 3D                    MOV DL, '='
   43    0066  E8 0087                  CALL putch
   44    0069  EB 06 90                 jmp Translate_to_dec
   45    006C                           Exit:
   46    006C  E8 009D                  CALL exit_f
   47    006F                           JmpFar:
   48    006F  EB C7                    jmp GetString
   49    0071                           Translate_to_dec:
   50                                   ;Соединяем    в AX все число в hex. Алгоритм такой - добавляем число, а потом
         сдвигаем и так пока +
   51                                   не     получим          в AX изначальное число
   52    0071  58                       POP AX
   53    0072  D1 C8 D1 C8 D1 C8        D1+          ROR AX,4
   54          C8
   55    007A  5E                       POP SI
   56    007B  03 C6                    ADD AX,SI
   57    007D  D1 C8 D1 C8 D1 C8        D1+          ROR AX,4
```



```
   58          C8
   59    0085  5E                       POP SI
   60    0086  03 C6                    ADD AX,SI
   61    0088  D1 C8 D1 C8 D1 C8        D1+          ROR AX,4
   62          C8
   63    0090  5E                       POP SI
   64    0091  03 C6                    ADD AX,SI
   65    0093  D1 C8 D1 C8 D1 C8        D1+          ROR AX,4
   66          C8
   67
   68                                   ;Теперь переводим и выводим число в dec виде, но предворительно выведем hex
вид числа и пробел
   69    009B  50                       push AX
   70    009C  B9 000A                  MOV CX, 10
   71    009F  BE 0000                  MOV SI, 0
   72    00A2  BA 0000r                 lea DX, Buf_Hex
   73    00A5  E8 0043                  CALL printstr
   74    00A8  BA 0068                  MOV DX, 'h'
   75    00AB  E8 0042                  CALL putch
   76    00AE  BA 0020                  MOV DX, ' '
   77    00B1  E8 003C                  CALL putch
   78    00B4  58                       POP AX
   79    00B5                           Convert: ;      Делим на 10 число в шестнадцетеричной системе, пока полностью
не получим его в              +
   80                                   десятичной    системе
   81    00B5  33 D2                    xor DX,DX
   82    00B7  F7 F1                    DIV CX
   83    00B9  52                       push DX
   84    00BA  83 C6 01                 ADD SI,1
   85    00BD  3D 0000                  CMP AX, 0
   86    00C0  74 02                    je      Print_dec
   87    00C2  EB F1                    JMP Convert
```

```
   88   00C4                    Print_dec:; А теперь выводим записанное в стеке число, добавляя 48      и получая
соответсвующий   +
   89                           символ из таблицы ASCII
   90   00C4  5A                        pop DX
   91   00C5  80 C2 30                  ADD DL,48
   92   00C8  E8 0025                   call putch
   93   00CB  4E                        DEC SI
   94   00CC  83 FE 00                  cmp SI, 0
   95   00CF  74 9E                     je      JmpFar
   96   00D1  EB F1                     jmp Print_dec
   97
   98
   99   00D3                    Remember_numbers:
  100                                   ;Запоминаем цифры (не символы) шестднадцетеричных чисел
  101   00D3  3C 41                     cmp AL, 'A'
  102   00D5  7C 0A                     jl      if_less_then_A
  103   00D7  50                        push AX
  104   00D8  2C 37                     sub AL,55;(Пример для символа F(код ASCII - 70, а в HEX-46):
46 - 55 = -15       = -F+
  105                           (тк. минус      не учитывается) = F)
  106   00DA  8A D0                     MOV DL,AL
  107   00DC  58                        POP AX
  108   00DD  52                        push DX
  109   00DE  E9 FF77                   jmp AfterCheck
  110   00E1                            if_less_then_A:
  111   00E1  50                                push AX
  112   00E2  2C 30                             sub AL,48
  113   00E4  8A D0                             MOV DL,AL
  114   00E6  58                                POP AX
```



```
  115   00E7  52                                push DX
  116   00E8  E9 FF6D                   jmp AfterCheck
  117
  118   00EB                    printstr proc
  119   00EB  B4 09                     MOV ah, 09h
  120   00ED  CD 21                     int 021h
  121   00EF  C3                        ret
  122   00F0                    printstr endp
  123
  124   00F0                    putch proc
  125   00F0  B4 02                     MOV ah, 02h
  126   00F2  CD 21                     int 021h
  127   00F4  C3                        ret
  128   00F5                    putch endp
  129
  130   00F5                    getch proc
  131   00F5  B4 08                     MOV ah, 08h
  132   00F7  CD 21                     int 021h
  133   00F9  C3                                ret
  134   00FA                    getch endp
  135
  136   00FA                    clrf proc
  137   00FA  B2 0A                     MOV dl, 10
  138   00FC  E8 FFF1                   CALL putch
  139   00FF  B2 0D                     MOV dl, 13
  140   0101  E8 FFEC                   CALL putch
  141   0104  C3                        ret
  142   0105                    clrf endp
  143
  144   0105                    clrscr proc
```

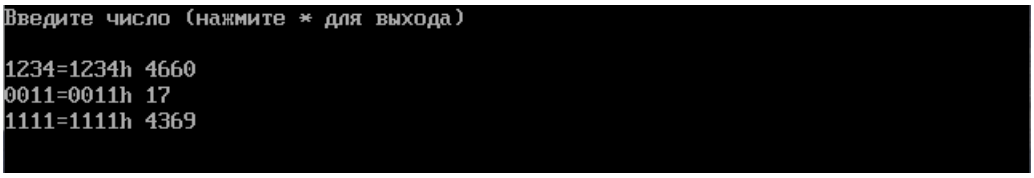
```
145   0105  B4 00                       MOV ah, 00h
146   0107  B0 02                       MOV al, 02
147   0109  CD 10                       int 10h
148   010B  C3                          ret
149   010C                        clrscr endp
150
151   010C                        exit_f proc
152   010C  B0 00                       MOV al, 0
153   010E  B4 4C                       MOV ah, 4ch
154   0110  CD 21                       int 021h
155   0112                        exit_f endp
156
157   0112                  MYCODE ends
158                         end start
```


Symbol Table

Symbol Name	Type	Value	Cref (defined at #)
??DATE	Text	"05/09/23"	
??FILENAME	Text	"LAB7 "	
??TIME	Text	"16:31:41"	
??VERSION	Number	030A	
@CPU	Text	0101H	
@CURSEG	Text	MYCODE	#1
@FILENAME	Text	LAB7	
@WORDSIZE	Text	2	#1
AFTERCHECK	Near	MYCODE:0058	#36 109 116
BUF_HEX	Byte	MYCODE:0000	#4 22 72
CHECK_IF_IN_HEX	Near	MYCODE:004D	#30
CLRF	Near	MYCODE:00FA	18 24 #136
CLRSCR	Near	MYCODE:0105	15 #144
CONVERT	Near	MYCODE:00B5	#79 87
EXIT	Near	MYCODE:006C	29 #45
EXIT_F	Near	MYCODE:010C	46 #151
GETCH	Near	MYCODE:00F5	27 #130
GETSTRING	Near	MYCODE:0038	#20 48
GETSYM	Near	MYCODE:0044	#25 32 34 41
IF_LESS_THEN_A	Near	MYCODE:00E1	102 #110
JMPFAR	Near	MYCODE:006F	#47 95
MAIN	Near	MYCODE:002C	#14
PRINTSTR	Near	MYCODE:00EB	17 73 #118
PRINT_DEC	Near	MYCODE:00C4	86 #88 96
PUTCH	Near	MYCODE:00F0	39 43 75 77 92 #124 138 140
REMEMBER_NUMBERS	Near	MYCODE:00D3	35 #99
START	Near	MYCODE:002A	#11 158
TRANSLATE_TO_DEC	Near	MYCODE:0071	44 #49
WELCOME	Byte	MYCODE:0005	#5 16

Groups & Segments	Bit Size	Align	Combine	Class	Cref (defined at #)
MYCODE	16	0112	Para none	CODE	#1 2 2

7. Результаты работы программы



8.Выводы по ЛР № 7

В ходе выполнения данной лабораторной работы на ассемблере я освоил навыки разработки и отладки программ, которые вводят числа с клавиатуры, переводят их из шестнадцатеричной системы в десятичную и выводят результат на экран. Также я научился проверять вводимые символы, чтобы программа работала корректно. Циклический режим работы позволяет повторно вводить новые числа, что также было реализовано.