Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук

Департамент

Программной инженерии

Микропроект по
дисциплине
«Архитектура вычислительных систем»

Тема работы: Вариант 3. Сравнение двух ASCII-последовательностей

Выполнил: студент группы БПИ191
Архаров Дмитрий Павлович

тел. +7 9852702653
e-mail адрес: dparkharov@edu.hse.ru

Преподаватель: Легалов Александр Иванович

Тестирование программы

Для того, чтобы запустить программу, нужно запустить её из командной строки с параметрами (две строки) через пробел (Пример: strcmp03.exe wow yes).

Существуют заранее подготовленные .bat файлы, для быстрого тестирования отдельных случаев, возникающих в программе, которые проводят к различным исходам.

Список .bat файлов и их назначение:

1. executeEqual.bat – передает две одинаковых последовательности, что влечет к ответу Equal
2. executeLower.bat – передает две последовательности, где первая лексикографически расположена позже другой
3. executeGreater.bat – передает две последовательности, где первая лексикографически расположена раньше другой
4. executeDifferentLength – передает две последовательности разной длины

Ответ представлен в виде всплывающего MessageBox.

Ход работы программы

1. Программа принимает на вход две последовательности
2. Проверяется длина последовательностей.
3. Если длина разная, то считается, что больше та, что длиннее.
4. Если длина одинакова, то вызывается функция, вычисляющая, находится ли первая строка лексикографически раньше второй.

Код программы

```
; BPI 191, Var 3: Arkharov Dmithry
; The application compares two strings and returns if the first is "greater" than second
format  PE Console
include "C:\Users\wsb.bart\Desktop\Assembler\INCLUDE\WIN32AX.INC" ; WIN32AX.INC location

entry main

section '.data' data readable writeable ; here is the data stored

        endl FIX 10, 13 ; \n

        ; Fields to store necessary data
        argc dd ?
        argv dd ?
        env dd ?

        str_buffer  DB  256 dup (0) ; str buffer
mb_titledb          "BPI191 Arkharov Dmithry strcmp.exe", 0 ; messagebox title (just not to rewrite it several times)

section '.text' code readable executable ; here is were the executable code starts

proc main
        invoke GetCommandLine  ; getting args
        cinvoke getmainargs,argc,argv,env,0 ;
        mov esi,[argv] ; storing from argv

        mov eax, dword[esi+4] ;first str
        mov ebx, dword[esi+8] ;second str
```

```
32.
33.         ;checking if lengths are equal
34.         mov ecx, eax
35.         push ecx
36.         call strlen
37.         mov edx, EAX
38.         mov ecx, ebx
39.         push ecx
40.         call strlen
41.         cmp edx, EAX
42.         je .compare ;if lengths are equal we can call strcmp
43.         jl .lower   ;otherwise comparing lengths
44.         jg .greater
45.
46.         .compare:
47.         mov eax, dword[esi+4]
48.         mov ebx, dword[esi+8]
49.         push eax
50.         push ebx
51.
52.         call strcmp
53.
54.         cmp ecx, 0
55.         je .equal
56.
57.         cmp ecx, 1
58.         je .lower
59.
60.         cmp ecx, -1
61.         je .greater
62. endp
63.
64. .equal:
65.
66. invoke  wsprintf,str_buffer, "%s", "Equal"; buffer for messagebox
67. invoke  MessageBox,0,str_buffer, mb_title,MB_OK ; displaying messagebox
68. jmp .finish
69.
70. .greater:
71.
72. invoke  wsprintf,str_buffer, "%s", "Greater"; buffer for messagebox
73. invoke  MessageBox,0,str_buffer, mb_title,MB_OK ; displaying messagebox
74. jmp .finish
75.
76. .lower:
```

```
77.
78. invoke  wsprintf,str_buffer, "%s", "Lower"; buffer for messagebox
79. invoke  MessageBox,0,str_buffer, mb_title,MB_OK ; displaying messagebox
80. jmp .finish
81.
82. ; finish section
83. .finish:
84.         invoke ExitProcess,0
85.
86. ; Show results and other information
87. proc show_info
88.         enter 0, 0
89. invoke  wsprintf,str_buffer, str_format, [num], [answer]; getting correct
    buffer for MessageBox to show in
90.         invoke  MessageBox,0,str_buffer, mb_title,MB_OK ; Showing messagebox
91.         leave
92.         ret
93. endp
94.
95. ;returns length of str
96. proc    strlen lpStr:DWORD
97.         push    edi ecx
98.
99.         cld
100.              mov     edi,[lpStr]
101.              xor     ecx,ecx
102.              dec     ecx
103.              xor     eax,eax
104.              repne   scasb
105.              not     ecx
106.              dec     ecx
107.              mov     eax,ecx
108.
109.              pop     ecx edi
110.              ret
111.      endp
112.
113.      ;strcmp func
114.      proc strcmp
115.          mov ecx, 0
116.          strcmp_loop:
117.              mov byte dl,[eax+ecx]
118.              mov byte dh,[ebx+ecx]
119.              inc ecx
120.              cmp dl,0
```

```
121.              je strcmp_end_0
122.              cmp byte dl,dh
123.              je strcmp_loop
124.              jl strcmp_end_1
125.              jg strcmp_end_2
126.          strcmp_end_0:
127.              cmp dh,0
128.              jne strcmp_end_1
129.              xor ecx,ecx
130.              ret
131.          strcmp_end_1:
132.              mov ecx,1
133.              ret
134.          strcmp_end_2:
135.              mov ecx,-1
136.              ret
137.      endp
138.
139.      section '.idata' import data readable ; section for imports
140.              library
   user32,'user32.dll',kernel32,'kernel32.dll',msvcrt,'msvcrt.dll',shell32,'shell3
   2.dll'
141.              'include '\API\USER32.INC'
142.              include '\API\KERNEL32.INC'
143.              include '\API\SHELL32.INC'
144.
146.              import
   msvcrt,\_getmainargs,'__getmainargs',sscanf,'sscanf';,strcmp,'strcmp'
```