

Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук

Пояснительная записка к микропроекту

По дисциплине

“Архитектура вычислительных систем”

На тему

**“Разработка программы, вычисляющей число вхождений
символов в заданной ASCII-строке”**

Работу выполнил

Студент группы БПИ-194

_____ М. Г. Савинов

подпись, дата

Работу проверил

_____ А.И. Легалов

подпись, дата

Москва 2020

Содержание

Постановка задачи.....	3
Описание задачи.....	3
Пример работы программы	4
Приложение	7
Список использованной литературы	11

Постановка задачи

Необходимо разработать программу, вычисляющую число вхождений символов в заданной ASCII-строке.

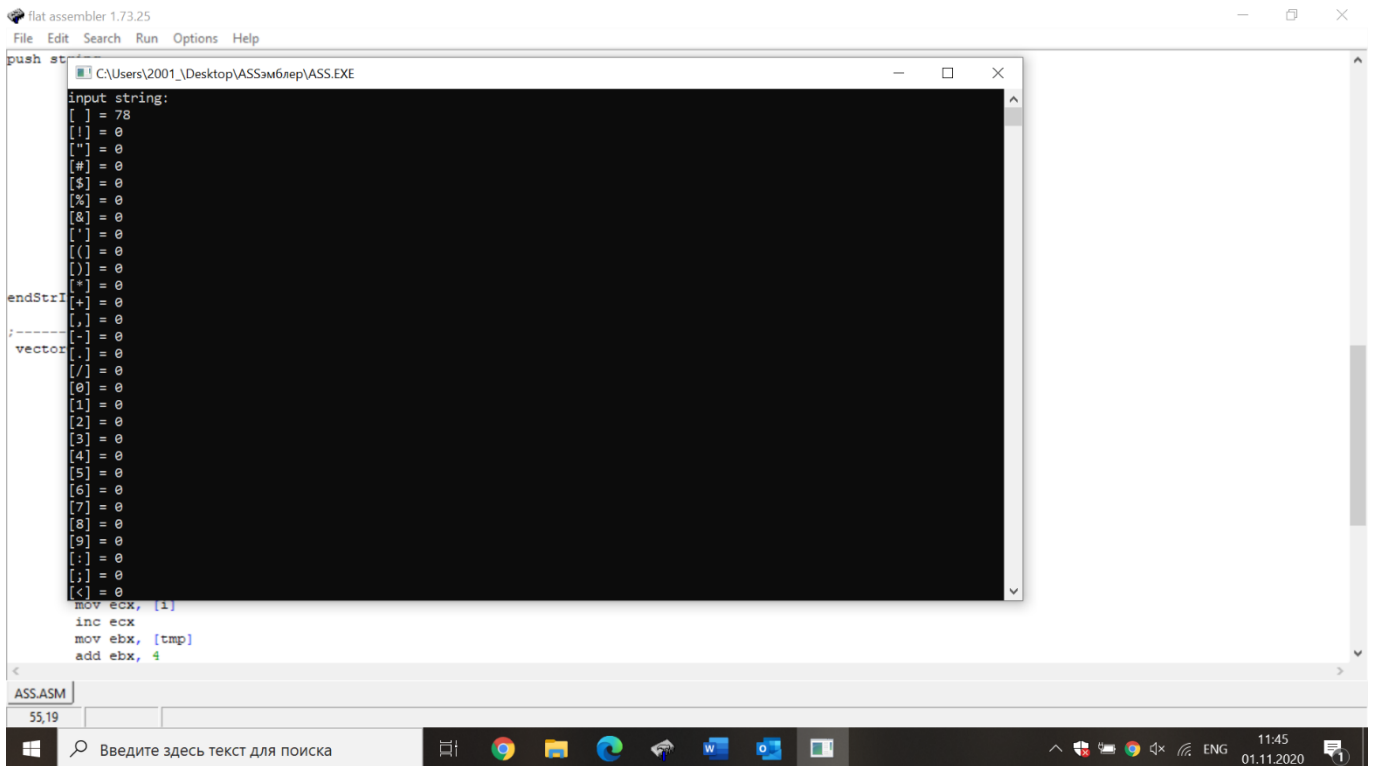
Задание следует выполнять с использованием цепочечных инструкций и соответствующих им команд-префиксов, управляющих изменением содержимого регистров **SI** и **DI** (Source Index и Destination Index соответственно). Для реализации циклов следует применять подходящие по смыслу модификации инструкции **LOOP** с контролем числа итераций по значению регистра **CX** (если число итераций априори известно) или расширенными методами (с выходом из цикла по условиям, отличающимся от классического **CX==0**).

Описание задачи

- 1) Написать программу на языке программирования Assembler (FASM), которая по заданной ASCII строке вычисляет количество вхождений каждого символа в этой строке. Поиск количества символов реализован с помощью цикла с условием, путем посимвольного чтения элементов заданной строки из регистра `esi` с использованием команды `lodsb`:
 - a. Создадим новый нулевой массив для учета количества элементов длиной 95 (32-126 элементы ASCII)
 - b. Вводим символы строки пока не будет нажата клавиша Enter
 - c. Записываем введенный символ в регистр `esi`
 - d. С помощью команды `lodsb` получаем введенный элемент `char` и уменьшаем его на 32, так как индексы нашего для записи 0-94
 - e. Получаем ASCII-код нашего символа через регистр `ax`
 - f. Увеличиваем количество вхождений этого символа на 1 в нашем массиве.
- 2) Программа принимает на вход строку, записанную поэлементно.
- 3) Программа выдает массив с количеством вхождения каждого элемента строки.

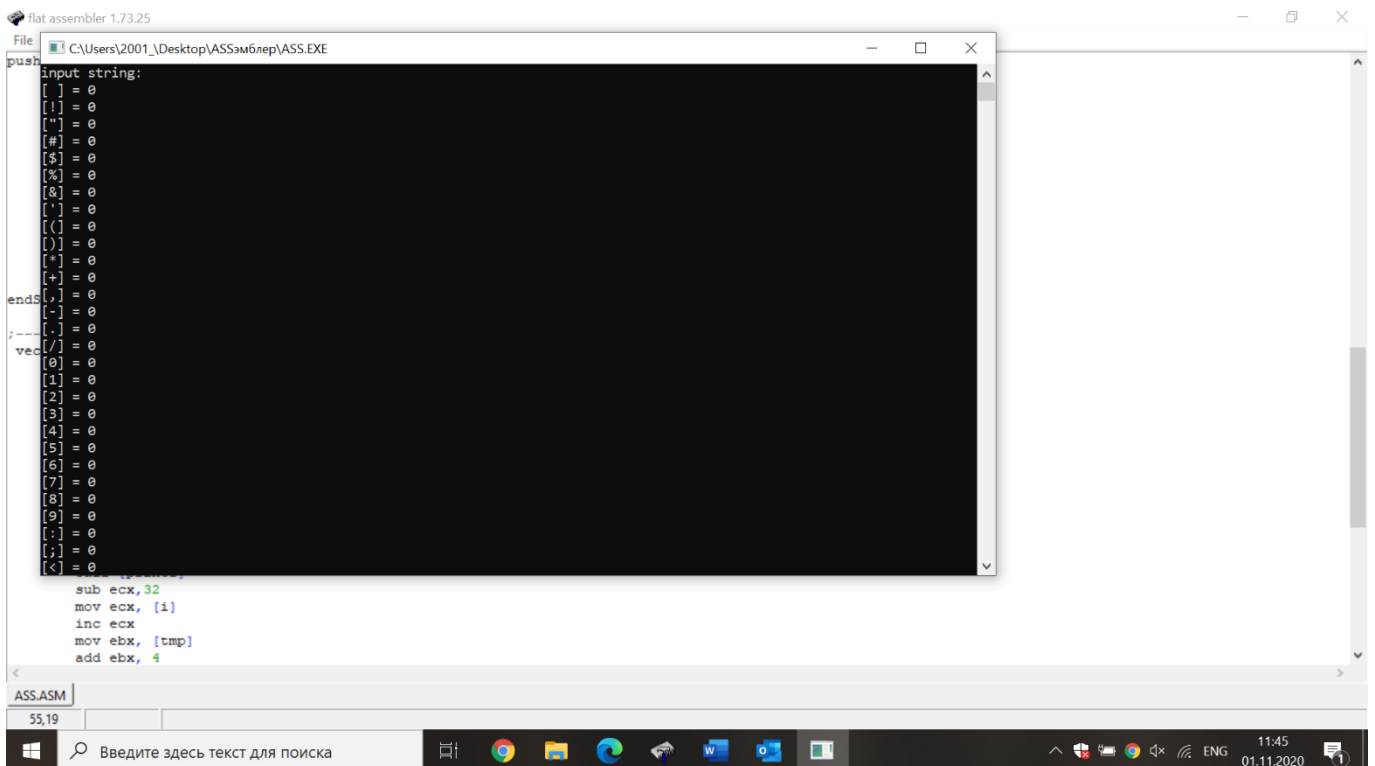
Пример работы программы

1) Введено 78 пробелов



```
push string:
[ ] = 78
[!] = 0
["] = 0
[#] = 0
[$] = 0
[%] = 0
[&] = 0
['] = 0
[(] = 0
[)] = 0
[*] = 0
[+] = 0
[,] = 0
[.] = 0
[-] = 0
[/] = 0
[0] = 0
[1] = 0
[2] = 0
[3] = 0
[4] = 0
[5] = 0
[6] = 0
[7] = 0
[8] = 0
[9] = 0
[:] = 0
[;] = 0
[<] = 0
mov ecx, [1]
inc ecx
mov ebx, [tmp]
add ebx, 4
ASS.ASM
55,19
```

2) Ничего не введено



```
push string:
[ ] = 0
[!] = 0
["] = 0
[#] = 0
[$] = 0
[%] = 0
[&] = 0
['] = 0
[(] = 0
[)] = 0
[*] = 0
[+] = 0
[,] = 0
[.] = 0
[-] = 0
[/] = 0
[0] = 0
[1] = 0
[2] = 0
[3] = 0
[4] = 0
[5] = 0
[6] = 0
[7] = 0
[8] = 0
[9] = 0
[:] = 0
[;] = 0
[<] = 0
sub ecx, 32
mov ecx, [1]
inc ecx
mov ebx, [tmp]
add ebx, 4
ASS.ASM
55,19
```

```

C:\Users\2001\Desktop\ASSam6rep\ASS.EXE
input string: %55
[ ] = 0
[!] = 0
["] = 0
[#] = 0
[$] = 0
[%] = 129
[&] = 0
['] = 0
[(] = 0
[)] = 0
[+] = 0
[,] = 0
[-] = 0
[.] = 0
[/] = 0
[0] = 0
[1] = 0
[2] = 0
[3] = 0
[4] = 0
[5] = 2
[6] = 0
[7] = 0
[8] = 0
[9] = 0
[:] = 0
[;] = 0
[<] = 0
[=] = 0
[>] = 0
[?] = 0
[@] = 0
[A] = 0
[B] = 0
[C] = 0
[D] = 0
[E] = 0
[F] = 0
[G] = 0

```

[illegible]

Приложение

Код программы:

```
format PE console
entry start
include 'win32a.inc'
;-----
section '.data' data readable writable
    mes1 db 'input string: ', 0
    strScan db '%c', 0
    p db 'pause', 0
    strVecElemOut db '[%c] = %d', 10, 0
    string dd ?
    i dd ?
    tmp dd ?
    mas rd 100
    vec_size dd 95 ;because we use only 32-126 ASCII-symbols
;-----
section '.code' code readable executable
start:
; 1) filling vector mas with 0
    call getVector
; 2) input string and count symbols
    call strInput
; 3) vector mas out
    call vectorOut
finish:
    call [getch]
    push 0
    call [ExitProcess]
;-----
getVector: ;filling mas with 0
```

```

    xor ecx, ecx
    mov ebx, mas
getVecLoop:
    mov [tmp], ebx
    cmp ecx, [vec_size]
    jge endFillingVector
    mov [i], ecx
    mov ebx, 0
    mov ecx, [i]
    inc ecx
    mov ebx, [tmp]
    add ebx, 4
    jmp getVecLoop
endFillingVector:
    ret
;-----
strInput:                ;invite to input string
    push mes1
    call [printf]
    add esp, 4
checkNewElem:            ;check what is input symbol
    push string
    push strScan
    call [scanf]
    add esp, 8
    mov esi, string
    lodsb                 ;read elem in register esi
    cmp al, 0x0A           ;while key Enter didn't press
    je endStrInput
    sub al, 32              ;because we use only 32-126 ascii symbols
    mov edi, eax
    mov eax, [mas+edi*4]   ;increase count of input element in mas by 1

```



```

    inc eax
    mov [mas+edi*4],eax
    jmp checkNewElem
endStrInput:
    ret
;-----
vectorOut:
    xor ecx,ecx           ;cout vector
    mov ebx,mas
putVecLoop:
    mov [tmp], ebx
    cmp ecx,[vec_size]
    je endOutputVector
    mov [i], ecx
    add ecx,32           ;because we use only 32-126 ascii symbols
    push dword [ebx]
    push ecx
    push strVecElemOut
    call [printf]
    sub ecx,32
    mov ecx, [i]
    inc ecx
    mov ebx, [tmp]
    add ebx, 4
    jmp putVecLoop
endOutputVector:
    cinvoke system,p
;-----
section '.idata' import data readable
    library kernel, 'kernel32.dll',\
        msvcrt, 'msvcrt.dll',\
        user32,'USER32.DLL'

```

```
include 'api\user32.inc'
include 'api\kernel32.inc'
import kernel,\
    ExitProcess, 'ExitProcess',\
    HeapCreate, 'HeapCreate',\
    HeapAlloc, 'HeapAlloc'
include 'api\kernel32.inc'
import msvcrt,\
    printf, 'printf',\
    scanf, 'scanf',\
    getch, '_getch',\
    system, 'system'
```

Список использованной литературы

1. Flat Assembler 1.64 – Мануал Программера <http://flatassembler.narod.ru/fasm.htm#2-1-10>