МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Организация ЭВМ и системы»
Тема: Представление и обработка символьной информации с использованием строковых команд

Студента гр. 1383 _____ Самулевич С.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

ФОРМУЛИРОВКА ЗАДАНИЯ

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение. Необходимые датчики псевдослучайных чисел находятся в каталоге Tasks\RAND_GEN (при его отсутствии программу датчика получить у преподавателя). Далее должен вызываться ассемблерный модуль(модули) для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину. Результирующий массив частотного распределения чисел по интервалам, сформированный на ассемблерном уровне, возвращается в программу, реализованную на ЯВУ, и затем сохраняется в файле и выводится на экран средствами ЯВУ. Исходные данные. 1. Длина массива псевдослучайных целыхчисел - NumRanDat (<= 16K, К=1024) 2. Диапазон изменения массива псевдослучайных целых чисел [Xmin, Xmax] , значения могут быть биполярные; 14 3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (<=24) 4. Массив левых границ интервалов разбиения LGrInt (должны принадлежать интервалу [Xmin, Xmax]). Результаты: 1. Текстовый файл, строка которого содержит: - номер интервала, - левую границу интервала, - количество псевдослучайных чисел, попавших в интервал. Количество строк равно числу интервалов разбиения.

Выполнение работы

Сначала на ЯВУ реализованы создания входного, выходного массивов ***input*** и ***output***.

Далее реализована программа на зыке ассемблера, переводящая символы русского алфавита в соответствующие латинские по правилам транслитерации. Для этого в регистры ***esi*** и ***edi*** выгружаются адреса исходных массивов ввода и вывода, после чего происходит считывание символа при помощи команды ***lodsb***. Далее символ проверяется по коду на попадание в диапазон русского алфавита и на букву «ё». При несовпадении символ записывается в выходной массив без изменений при помощи команды ***stosb***.

При попадании в диапазон код символа сравнивается с теми, буквам которых соответствует отличное от 1 количество символов, запись которых происходит с помощью двух или более вызовов ***stosb***.

Программный код см. в приложении А

Выводы

В ходе работы были изучены способы представления строк и разработана программа их обработки.

Приложение А исходный код программы

```
#include <stdio.h>
#include <windows.h>

char input[100];
char output[100];
char lat[] =
"ABVGDEZhZIJKLMNOPRSTUFHTsChShShchYEYuYaabvgdezhzijklmnoprstufhtsch
shshchyeyuya";

int main() {
SetConsoleCP(1251);
SetConsoleOutputCP(1251);
fgets(input, 100, stdin);

_asm {
push ds
pop es
sub cx, cx
mov eax, 0
mov ecx, 0
mov esi, offset input
mov edi, offset output
jmp start

mov0 :
jmp start

mov1 :
lodsb
stosb
jmp start

mov2 :
lodsb
stosb
```

```
lodsb
stosb
jmp start

mov4 :
lodsb
stosb
lodsb
stosb
lodsb
stosb
lodsb
stosb
jmp start

start :
mov esi, offset input
add esi, ecx
lodsb
add cx, 1
cmp al, '\0'
je end1

cmp al, 184
jne nYOs
mov al, 101
stosb
jmp start

nYOs:

cmp al, 168
jne nYOb
mov al, 69
stosb
jmp start

nYOb:
```

```
sub al, 192

cmp al, 0
jge aA
add al, 192
stosb
jmp start
aA :
cmp al, 63
jle bYa
add al, 192
stosb
jmp start
bYa :

mov bl, al
cmp al, 32
jge sm
jmp read
sm:
add al, 7
sub bl, 32

read:
mov esi, offset lat
add esi, eax

cmp bl, 6
je mov2
jl end
add esi, 1

cmp bl, 22
je mov2
jl end
add esi, 1

cmp bl, 23
je mov2
```

```
jl end
add esi, 1

cmp bl, 24
je mov2
jl end
add esi, 1

cmp bl, 25
je mov4
jl end
add esi, 3

cmp bl, 26
je start
jl end
sub esi, 1

cmp bl, 28
je start
jl end
sub esi, 1


cmp bl, 30
je mov2
jl end
add esi, 1

cmp bl, 31
je mov2
jl end

end :
jmp mov1

end1 :

}
```

```
printf("%s", output);

}
```