МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Организация ЭВМ и систем»

Тема: Организация связи Ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы.

Студент гр. 1303 Иванов А. С.

Преподаватель Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Написать программу, которая считывает исходные данные и генерирует массив случайных чисел через ЯВУ и подсчитывает кол-во чисел в интервалах через модуль, написанный на языке ассемблера.

Задание.

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение.

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.

Выполнение работы.

Для того, чтобы компилятор понимал, что function определена в другом модуле и таким образом он связывается с основной программой.

```
extern "C" {void function (int* Array, int len, int* LGrInt, int NInt, int* answer); }
```

Далее считываются исходные данные:

кол-во чисел в массиве, который надо будет сгенерировать случайным образом, пределы разброса значений в этом массиве, кол-во интервалов и их левые границы (правой границей интервала по умолчанию считается следующая левая граница).

Далее генерируется массив и передается в function, определенную в ассемблерном модуле.

В данном модуле реализован цикл, проходящийся по всем элементам массива чисел. Каждое число сравнивается со всеми левыми границами последовательно, до тех пор, пока число не окажется меньше текущей левой границы. Тогда в массиве result в ячейке с индексом, соответствующим этой левой границе, значение увеличивается на 1 и цикл повторяется для уже следующего числа.

Далее результат с помощью ЯВУ выводятся в файл.

Разработанный программный код см. в приложении А.

Вывод.

Написана программа, состоящая из двух модулей: c++ и ассемблера. Программа генерирует массив случайных чисел и подсчитывает количество чисел в каждом интервале.

ПРИЛОЖЕНИЕ А

Название файла: lab6.asm

```
.586p
.MODEL FLAT, C
.CODE
handleNumbers PROC C USES EDI ESI, array:dword, len:dword,
leftBord:dword, n_int:dword, result:dword
	push eax
	push ebx
	push ecx
	push edi
	push esi
	mov ecx, len
```

```
      mov esi, array
      mov edi, leftBord
      mov eax, 0
lp:
      mov ebx, 0 ; индекс текущего интервала
      handle:
            cmp ebx, n_int ; цикл пока i < n_int
            jge after_handle ; выход из цикла
            push eax
            mov eax, [esi + 4 * eax] ; кладем в eax элемент массива
            cmp eax, [edi + 4 * ebx] ; сравниваем этот элемент с текущей
левой границей
            pop eax          ; возвращаем eax нулевое значение
            jl after_handle ; выходим из цикла если элемент меньше
            inc ebx          ; переход к след. интервалу если элемент
больше
            jmp handle ; повторяем пока не выйдем

      after_handle:          ; после завершения while
            dec ebx          ; уменьшаем индекс интервала на 1
            cmp ebx, -1            ; проверка на выход за пределы массива
интервалов
            je to_next_num   ; если условие сработало то переходим к след.
числу
            mov edi, result
            push eax
            mov eax, [edi + 4 * ebx]    ; в eax помещаем элемент массива
result с индексом ebx
            inc eax
            mov [edi + 4 * ebx], eax    ; устанавливаем полученное значение
обратно в массив result
            pop eax
            mov edi, leftBord                ; возвращаем в edi leftBord
для дальнейших итераций

      to_next_num:
            inc eax          ; увеличиваем индекс рассматриваемого числа
```

```
loop lp                    ;цикл  выполняется пока длина не окажется
нулевой
pop esi
pop edi
pop ecx
pop ebx
pop eax
ret

handleNumbers ENDP
END
```

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <random>
#include <string>

using namespace std;

extern "C" {void handleNumbers(int* Array, int len, int* LGrInt, int
NInt, int* answer); }

int main() {
	int NumRamDat, minX, maxX, NInt;
	cout << "Enter array length\n";
	cin >> NumRamDat;
	if (NumRamDat <= 0 || NumRamDat >= 16 * 1024) {
		std::cout << "Incorrect NumRamDat\n";
		return 1;
	}
	cout << "Enter minX and maxX\n";
	cin >> minX;
```

```
cin >> maxX;
if (minX >= maxX) {
      std::cout << "Invalid Xmin and Xmax values\n";
      return 1;
}
cout << "Enter number of intervals\n";
cin >> NInt;

if (NInt <= 0 || NInt > 24) {
      cout << "The number of intervals must be in [0; 24)\n";
      return 1;
}



cout << "Enter left borders\n";
auto intervals = new int[NInt];
for (int i = 0; i < NInt; ++i) {
      cin >> intervals[i];
      if ((intervals[i] < minX || intervals[i] > maxX) ||
            (i > 0 && intervals[i] <= intervals[i - 1])) {
            printf("Incorrect left border\n");
            return 1;
      }
}



auto numbers = new int[NumRamDat];
random_device rd;
mt19937 generator(rd());
uniform_int_distribution<> dist(minX, maxX);
for (int i = 0; i < NumRamDat; i++) {
      numbers[i] = dist(generator);
}
```

```
	auto result = new int[NInt];

	for (int i = 0; i < NInt; i++) {
		result[i] = 0;
	}

	handleNumbers(numbers, NumRamDat, intervals, NInt, result);

	ofstream file("result.txt");
	file << "Numbers:\n";
	for (int i = 0; i < NumRamDat; ++i) {
		file << numbers[i] << " ";
	}
	file << "\nResult:\n";
	for (int i = 0; i < NInt; ++i) {
		file << "Border - " << "'" << intervals[i] << "'" << " has
amount of numbers " << result[i] << "\n";
	}
	file.close();
	return 0;
}
```