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 _____ Попандопуло А. Г.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Практическое изучение работы ассемблерного модуля с программой на ЯВУ. Согласно условию, разработать программу генерации псевдослучайных чисел, с подсчетом их попаданий в заданные интервалы.

Задание.

Вариант 1

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.

Подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде одного ассемблерного модуля, сразу формирующего требуемое распределение и возвращающего его в головную программу, написанную на ЯВУ;

Выполнение работы

Часть программы, использующая ЯВУ осуществляет исчитывание необходимых данных: длину массива псевдослучайных чисел, диапозон их изменения, количество интервалов разбиения, массив левых границ интервалов разбиения. Названия переменных соответствуют условию, под них выделяется память, также реализованы некоторые проверки вводимых значений.

Псевдослучайные числа генерируются с помощью функции стандартной библиотеки языки Си.

Ассемблерный модуль представляет собой процедуру, получающую на вход вышеперечисленные исходные данные и массив hits, длины количества интервалов разбиения. В ней для каждого числа сгенерированного массива осуществляется проверка на соответствующие интервалы. Таким образом, количество попаданий в первый интервал соответствует первому элементу массива hits и так далее.

Результат работы программы, согласно условию, выводится как в файл, так и экран.

Тестирование:

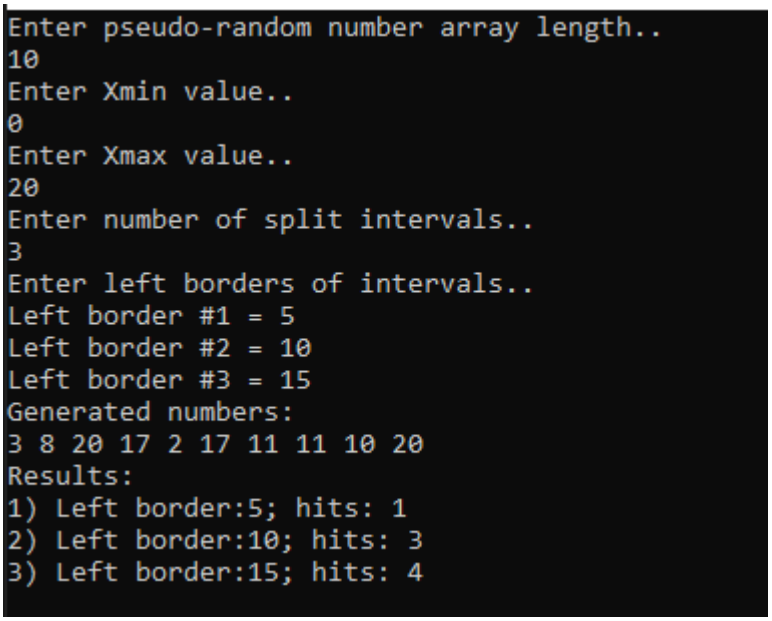

Рис. 1

Вывод: в ходе выполнения лабораторной работы, на практике была изучена организация работы ЯВУ с модулем на языке Ассемблера; удалось реализовать программу генерации псевдослучайных чисел с проверкой и подсчетом на соответствие заданным интервалам.

Приложение А.

Исходный код программы.

Название файла: lb6.cpp

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <cstdio>


extern "C" {void process_hits(int* Array, int len, int* LGrInt, int NInt, int* hits); }
std::ofstream file("result.txt");

int main() {
	srand(static_cast <unsigned int> (time(nullptr)));

	int NumRanDat = 0;
	int x_min = 0;
	int x_max = 0;
	int NInt = 0;

	std::cout << "Enter pseudo-random number array length..\n";
	std::cin >> NumRanDat;

	if (NumRanDat > 16 * 1024 || NumRanDat <= 0) {
		std::cout << "Wrong number array lenght!\n";
		return 1;
	}

	std::cout << "Enter Xmin value..\n";
	std::cin >> x_min;
	std::cout << "Enter Xmax value..\n";
	std::cin >> x_max;

	if (x_min >= x_max) {
		std::cout << "Xmin can't be more than Xmax!\n";
		return 1;
	}

	std::cout << "Enter number of split intervals..\n";
	std::cin >> NInt;

	if (NInt > 24 || NInt <= 0) {
		std::cout << "Invalid number of intervals\n";
```

```
            return 1;
        }

        int* n_arr = new int[NumRanDat];
        int* int_arr = new int[NInt];

        std::cout << "Enter left borders of intervals..\n";
        for (int i = 0; i < NInt; i++) {
            std::cout << "Left border #" << i + 1 << " = ";
            std::cin >> int_arr[i];
            if ((i > 0 && int_arr[i] <= int_arr[i - 1]) || (int_arr[i] < x_min ||
int_arr[i] > x_max)) {
                printf("Some border is wrong!\n");
                delete[] n_arr;
                delete[] int_arr;
                return 1;
            }
        }

        int rand_val = x_max - x_min + 1;
        for (int i = 0; i < NumRanDat; i++) {
            n_arr[i] = x_min + rand() % rand_val;
        }

        int* hits_arr = new int[NInt] {0};
        process_hits(n_arr, NumRanDat, int_arr, NInt, hits_arr);

        std::cout << "Generated numbers:\n";
        file << "Generated numbers:\n";
        for (int i = 0; i < NumRanDat; ++i) {
            std::cout << n_arr[i] << " ";
            file << n_arr[i] << " ";
        }
        std::cout << "\nResults:\n";
        file << "\nResults:\n";

        for (int i = 0; i < NInt; i++) {
            std::cout << i + 1 << ") Left border:" << int_arr[i] << "; hits: " <<
hits_arr[i] << "\n";
            file << i + 1 << ") Left border:" << int_arr[i] << "; hits: " << hits_arr[i]
<< "\n";
        }

        file.close();
        delete[] n_arr;
```

```
	delete[] int_arr;
	delete[] hits_arr;

	return 0;
};
```

Название файла: module.asm

```
.586p
.MODEL FLAT, C
.CODE
process_hits PROC C USES EDI ESI, array:dword, len:dword, LBorders:dword,
NInt:dword, hits:dword
	push eax
	push ebx
	push ecx
	push edi
	push esi

	mov ecx, len
	mov esi, array
	mov edi, LBorders
	mov eax, 0

process:
	mov ebx, 0
	intervals_cycle:
		cmp ebx, NInt
		jge over_interval;
		push eax
		mov eax, [esi + 4*eax]
		cmp eax, [edi + 4*ebx]
		pop eax
		jl over_interval
		inc ebx
		jmp intervals_cycle

	over_interval:
		dec ebx
		cmp ebx, -1
		je next_num
		mov edi, hits
		push eax
		mov eax, [edi + 4*ebx]
		inc eax
		mov [edi + 4*ebx], eax
```

```
            pop eax
            mov edi, LBorders
        next_num:
            inc eax

loop process

pop esi
pop edi
pop ecx
pop ebx
pop eax

ret
process_hits ENDP
END
```