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 распределения попаданий псевдослучайных целых чисел в заданные интервалы.

Студент гр. 0382 _____ Злобин А.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить работу с организацией связи Ассемблера с ЯВУ. Написать программу построения частотного распределения попаданий псевдослучайных целых чисел в интервалы, определённые индивидуальным заданием.

Задание.

На языке С программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих заданный закон распределения. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND_GEN (при его отсутствии получить у преподавателя).

Следует привести числа к целому виду с учетом диапазона изменения.

Далее должны вызываться 1 или 2 ассемблерных процедуры для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Ассемблерные процедуры должны вызываться как независимо скомпилированные модули. Передача параметров в процедуру должна выполняться через кадр стека.

Результирующий массив частотного распределения чисел по интервалам, сфор-мированный на ассемблерном уровне, возвращается в программу, реализованную на ЯВУ, и затем сохраняется в файле и выводится на экран средствами ЯВУ.

Вариант 9

Равномерное распределение случайных чисел, одна ассемблерная процедура

$$\text{Nint} < D_x, \ \text{Lg1} > X_{min}, \ \text{ПГпосл} > X_{max}.$$

Выполнение работы.

Реализовано считывание количества генерируемых чисел, граничных значений генерируемых чисел, количества интервалов разбиения и левых границ интервалов на языке C++. Случайные числа генерируются и заносятся в массив, левые границы интервалов заносятся в отдельный массив, создается результирующий массив, в который в дальнейшем по i-тому индеку будет заносится количество чисел, попавший в i-тый интервал. В ассемблерный модуль в процедуру FUNC передаются указатель на массив сгенерированных чисел, его размер, указатель на массив левых границ интервалов и его размер, указатель на результирующий массив. В процедуре совершается цикл по всем элементам массива сгенерированных чисел, для каждого находится интервал, в который оно попадает и в результирующем массиве инкрементируется соответствующий элемент. После того, как процедура из ассемблерного модуля завершила работу, на экран и в файл out.txt выводится текстовая таблица, содержащая номера интервалов, их левые границы и количество чисел, попавших в каждый интервал

Тестирование.

Таблица 1. Проверка работы программы с отладочным выводом сгенерированных чисел.

Исходные данные	Результат	Примечание
NumRanDat = 10 xMax = 1 xMin = -1 Nint = 2 LGrInt = {0; 1}	0 1 1 -1 1 1 0 -1 1 0 1 0 3 2 1 5	Верно

Выводы.

В ходе выполнения данной лабораторной работы была изучена организация связи ассемблера с ЯВУ. Была реализована программа частотного распределения случайных чисел по заданным интервалам на языке C++ с использованием ассемблерного модуля.

Приложение А

Исходный код программы

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <random>
using namespace std;
extern "C" void FUNC(int* array, int array_size, int* left_boarders, int intervals_size, int* result_array);
int main() {
setlocale(0, "");
std::ofstream file("out.txt");
int array_size;
cout << "Введите число генерируемых чисел: ";
cin >> array_size;
int xMin, xMax;
cout << "Введите минимальное значение: ";
cin >> xMin;
cout << "Введите максимальное значение: ";
cin >> xMax;
if (xMax < xMin) {
	cout << "Неверно введены максимальное и минимальное значения";
	return 0;
}
int intervals_size;
cout << "Введите количество интервалов: ";
cin >> intervals_size;
if (intervals_size <= 0) {
	cout << "Неверно введено количество интервалов";
	return 0;
}

int* left_boarders = new int[intervals_size];
cout << "Введите левые границы:";
for (int i = 0; i < intervals_size; i++)
```

```
            cin >> left_boarders[i];

      for (int i = 0; i < intervals_size-1; i++) {
            for (int j = i + 1; j < intervals_size; j++) {
                  if (left_boarders[j] < left_boarders[i]) {
                        swap(left_boarders[j], left_boarders[i]);
                  }
            }
      }

      std::random_device rd;
      std::mt19937 gen(rd());
      std::uniform_int_distribution<> dis(xMin, xMax);
      int* array = new int[array_size];
      for (int i = 0; i < array_size; i++) array[i] = dis(gen);

      file << "Сгенерированные числа: ";
      for (int i = 0; i < array_size; i++) file << array[i] << ' ';
      file << '\n';

      int* result_array = new int[intervals_size];
      for (int i = 0; i < intervals_size; i++)
            result_array[i] = 0;

      FUNC(array,     array_size,     left_boarders,     intervals_size,
result_array);

      cout << "Номер интервала \tЛевая граница интервала \tКоличество
чисел в интервале" << '\n';
      file << "Номер интервала \tЛевая граница интервала \tКоличество
чисел в интервале" << '\n';
      for (int i = 0; i < intervals_size; i++) {
            cout << "\t" << i + 1 << "\t\t\t" << left_boarders[i] <<
"\t\t\t\t" << result_array[i] << '\n';
            file << "\t" << i + 1 << "\t\t\t" << left_boarders[i] <<
"\t\t\t\t" << result_array[i] << '\n';
      }
      system("pause");
          return 0;
      }
```

Название файла: module.asm

```
      .586
      .MODEL FLAT, C
      .CODE
      FUNC  PROC  C  array:dword,  array_size:dword,  left_boarders:dword,
intervals_size:dword, result_array:dword
      push ecx
      push esi
      push edi
      push eax
```

```
    push ebx ; сохранение регистров

    mov ecx, array_size
    mov esi, array
    mov edi, left_boarders
    mov eax, 0     ; индекс рассматриваемого числа
    l1:             ; цикл по всем сгенерированным числам в массиве
    mov ebx, 0  ; индекс рассматриваемого интервала
    boarders:   ; цикл нахождения интервала, в который попадает число
              cmp ebx, intervals_size ; если дошли до последнего
интервала, выходим из цикла
          jge boarders_exit
          push eax
          mov eax, [esi+4*eax]
          cmp eax, [edi+4*ebx]
          pop eax
          jl boarders_exit
          inc ebx
          jmp boarders
    boarders_exit:
    dec ebx          ; на выходе получили индекс интервала, в который
попало число

    cmp ebx, -1     ; если индекс -1, то число не попало ни в один
интервал
    je skip
    mov edi, result_array
    push eax
    mov eax, [edi+4*ebx]
    inc eax
    mov [edi+4*ebx], eax
    pop eax
    mov edi, left_boarders
    skip:
    inc eax       ; переход к следующему числу
    loop l1

    pop ebx ; восттановление регистров
    pop eax
    pop edi
    pop esi
    pop ecx
    ret
    FUNC ENDP
    END
```