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. 1381 _____ Смирнов Д. Ю.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Научиться связывать язык Ассемблера и язык высокого уровня так, чтобы функции ассемблерного модуля вызывались из программы на ЯВУ. Написать программу построения частного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

Задание.

На языке высокого уровня программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение.

Далее должны вызываться две 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, 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 целых чисел – *NumRanDat*
2. Диапазон изменения массива псевдослучайных целых чисел $[X_{min}, X_{max}]$, могут быть биполярные;
3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - *NInt* (<=24)
4. Массив левых границ интервалов разбиения *LGrInt* (должны принадлежать интервалу $[X_{min}, X_{max}]$).

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.

Программа написана с использованием языка программирования C++. В файле *main.cpp* написаны функции для считывания входных данных, генерации псевдослучайных чисел, а также вывод выходных данных. В модуле, написанном на языке Ассемблера, обрабатывается массив псевдослучайных чисел. Для этого используются инструкция *loop*: пока не будут обработаны все числа из массива array, функция будет обрабатывать числа. Для каждого числа поочередно ищется соответствующий ему интервал (в метке *find_border*): если текущее число больше левой границы, то берется следующая граница, пока число не будет меньше текущей границы, тогда ее интервал – предыдущая граница - переходим по метку *out_of_border*, где соответствующий результат увеличивается на единицу. После этого переходим к следующему элементу массива array.

Результат работы программы выводится в файл result.txt.

Исходный код программы представлен в приложении А.

Тестирование.

Результаты тестирования представлены на рисунках 1 и 2.

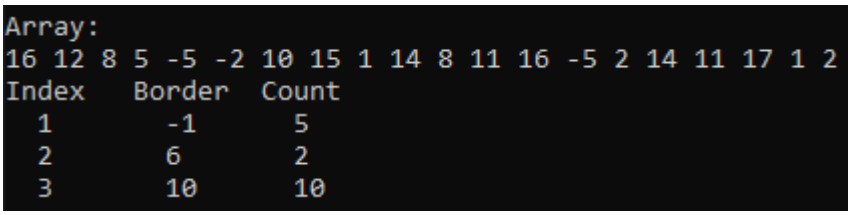

Рис. 1 – Тест 1: 20 случайных чисел ([-5, 20])

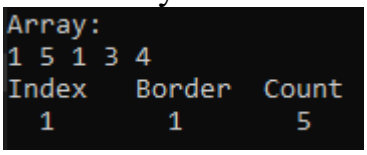

Рис. 2 – Тест 2: 5 случайных чисел ([1, 5])

Выводы.

Изучены принципы организации связи Ассемблера с ЯВУ, а также разработана программа, которая строит частотное распределение попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

Название файла: main.cpp

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

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

std::ofstream file_output;

void generate_array(int*& array, int length, int min, int max) {
    std::random_device seed;
    std::mt19937 gen(seed());
    std::uniform_int_distribution<int> dist{ min, max };
    for (int i = 0; i < length; i++) {
        array[i] = dist(gen);
    }

    file_output << "Pseudo-random array:\n";
    for (int i = 0; i < length; i++)
    {
        file_output << array[i] << " ";
    }
    file_output << "\n";
}

void get_data(int& NumRamDat, int*& array, int& min, int& max, int& NInt,
int*& LGrInt) {
    std::cout << "Input the length of the array of pseudo-random
numbers.\n";
    std::cin >> NumRamDat;

    array = new int[NumRamDat];

    std::cout << "Enter a range of random numbers:\nFrom:";
    std::cin >> min;
    std::cout << "To:";
    std::cin >> max;

    while (min >= max) {
        std::cout << "Incorrect Xmax, input again:";
        std::cin >> max;
    }
    generate_array(array, NumRamDat, min, max);

    std::cout << "Enter the number of split intervals:";
    std::cin >> NInt;
    while (NInt < 0 || NInt > 24) {
        std::cout << "Incorrect NInt, input again:";
        std::cin >> NInt;
    }
```

```
    LGrInt = new int[NInt];

    std::cout << "Enter intervals in ascending order\n";
    for (int i = 0; i < NInt; i++)
    {
        std::cout << "Border" << i + 1 << ": ";
        std::cin >> LGrInt[i];
        while (LGrInt[i] > max || LGrInt[i] < min) {
            std::cout << "Incorrect border, input again:";
            std::cin >> LGrInt[i];
        }
    }
    std::sort(LGrInt, LGrInt + NInt);
}

void print_data(int NInt, int NumRamDat, int* & Array, int*& LGrInt, int*&
answer) {
    std::cout << "Array:\n";
    for (int i = 0; i < NumRamDat; i++)
    {
        std::cout << Array[i] << " ";
    }
    std::cout << "\n";
    file_output << "\n";
    std::cout << "Index\t" << "Border\t" << "Count\n";
    file_output << "Index\t" << "Border\t" << "Count\n";
    for (int i = 0; i < NInt; i++) {
        std::cout << "   " << i + 1 << "\t   " << LGrInt[i] << "\t   " <<
answer[i] << '\n';
        file_output << "   " << i + 1 << "\t   " << LGrInt[i] << "\t   " <<
answer[i] << '\n';
    }

}

int main()
{
    file_output.open("result.txt", std::ios_base::out);
    int NumRamDat;
    int Xmin;
    int Xmax;
    int NInt;

    int* Array;
    int* LGrInt;

    get_data(NumRamDat, Array, Xmin, Xmax, NInt, LGrInt);

    int* answer_arr = new int[NInt] {0};

    mod_function(Array, NumRamDat, LGrInt, NInt, answer_arr);

    std::cout << "\n\n";

    print_data(NInt, NumRamDat, Array, LGrInt, answer_arr);
```

```
    delete[] Array;
    delete[] LGrInt;
    delete[] answer_arr;
    file_output.close();
}
```

Название файла: module.asm

```
.586p
.MODEL FLAT, C
.CODE
mod_function PROC C USES EDI ESI, array:dword, len:dword, LGrInt:dword,
NInt:dword, answer:dword

      push eax
      push ebx
      push ecx
      push edi
      push esi


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

loop_:
      mov ebx, 0
            find_border:
                  cmp ebx, NInt
                  jge out_of_border

                  push eax
                  mov eax, [esi + 4 * eax]
                  cmp eax, [edi + 4 * ebx]
                  pop eax
                  jl out_of_border
                  inc ebx
                  jmp find_border

            out_of_border:
                  dec ebx

                  cmp ebx, -1
                  je to_next_num
                  mov edi, answer
                  push eax
                  mov eax, [edi + 4 * ebx]
                  inc eax
                  mov [edi + 4 * ebx], eax
                  pop eax
                  mov edi, LGrInt

      to_next_num:
            inc eax

loop loop_
```

```
pop esi
pop edi
pop ecx
pop ebx
pop eax

ret

mod_function ENDP
END
```