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

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. 9383 ___ Рыбников Р.А.

Преподаватель ___ Ефремов М.А.

Санкт-Петербург
2020

Цель работы

Изучение особенностей организации связи Ассемблера с ЯВУ на примере программы построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

Задача

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, K=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:

Текстовый файл, строка которого содержит: - номер интервала, - левую границу интервала, - количество псевдослучайных чисел, попавших в интервал. Количество строк равно числу интервалов разбиения.

Выполнение работы

В файле main.cpp пользователь вводит все данные, которые понадобятся для дальнейшей работы программы. Длина массива с псевдослучайными числами, минимум и максимум значений, количество интервалов, а также нижние границы интервалов.

Далее происходит генерация случайных чисел в массиве, которые равномерно распределены.

Создаётся результирующий массив, в которые будут записано число попаданий чисел в интервал, после вызывается ассемблерный модуль в котором будет распределение количества попаданий псевдослучайных целых чисел в заданные интервалы.

В ассемблерном модуле происходит проходка по массиву из псевдослучайных чисел, в процессе чего идёт сравнение элементов с левыми границами . Если элемент больше или равен этого значения, то он отправляется в интервал, и счетчик количества элементов возрастает на 1.

В модуле ЯВУ происходит вывод результата в поток и файл.

Тестирование:

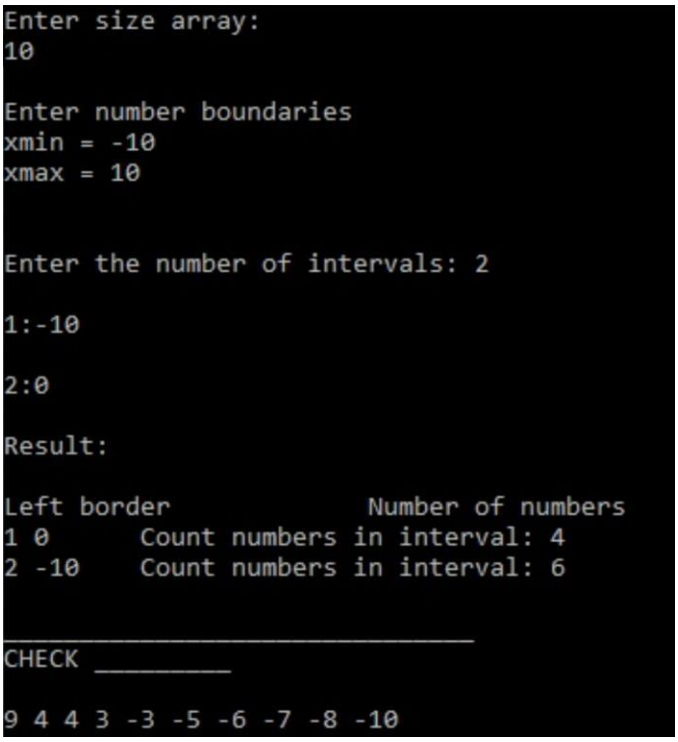

Рисунок 1 - Тест 1

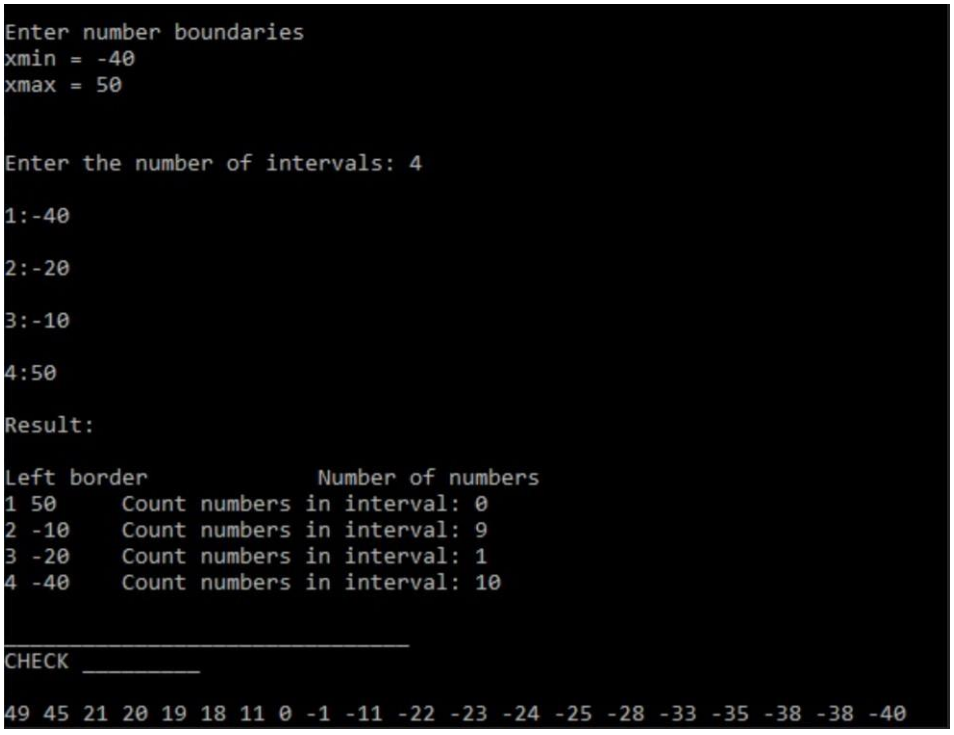

Рисунок 2 - Тест 2

Выводы.

Были изучены особенности организации связи Ассемблера с ЯВУ на примере программы построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

Приложение А

Исходный код

Файл main.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <time.h>
#include <random>
#include <clocale>

#define PATH
"/Users/ramka178rus/Desktop/Any_Poject_XC/asm_lab6/asm_lab6/output.txt"
#define PATH_VS "output.txt"
using namespace std;

extern "C" {
   void ASSEMBLY_ARR(unsigned short* result, short* arrLeftInt, short* array,
unsigned short countInterval, unsigned short sizeArray);
}


int main() {
   srand(time(0));


   unsigned short sizeArray = 0;      // длина массива
   short xmin = 0;                    // диапазон
```

```
    short xmax = 0;
    short* array;                    // массив заданных чисел
    unsigned short countInterval = 0;            // кол-во интеравалов
    short* arrLeftInt;               // массив левых границ
    unsigned short* result;                  // кол-во чисел в каждом интеравле

    std::ofstream out("output.txt");

#pragma region INPUT
    cout << "Enter size array: ";
    cin >> sizeArray;
    while (sizeArray > 16000 || sizeArray < 0){
        cout << "Error size, try again\n";
        cin >> sizeArray;
    }
    cout << endl;

    cout << "Enter number boundaries" << endl;
    cout << "xmin = ";
    cin >> xmin;

    cout << "xmax = ";
    cin >> xmax;

    cout << endl;

    while(xmax <= xmin) {
        cout << "Error xmin/xmax, try again\n";
        cout << "xmin = ";
        cin >> xmin;

        cout << "xmax = ";
        cin >> xmax;
    }

    cout << endl;

    cout << "Enter the number of intervals: ";
    cin >> countInterval;
    while(countInterval > 24 || countInterval < 0){
        cout << "Error countInterval, try again\n";
        cin >> countInterval;
    }
```

```
    cout << endl;
#pragma endregion

    array = new short[sizeArray];
    arrLeftInt = new short[countInterval];

    for (int i = 0; i < countInterval; i++)
    {
        std::cout << i + 1 << ":";
        std::cin >> arrLeftInt[i];
        std::cout << '\n';
    }


    /*for(int i = 1; i < countInterval - 1; ++i){
        do{
            cout << i + 1 << ": ";
            cin >> arrLeftInt[i];
            cout << endl;
        } while(arrLeftInt[i] < xmin || arrLeftInt[i] > xmax);
    }*/
    // отсортировали массив границ
    for(int i = 0; i < countInterval - 1; ++i){
        for (int j = 0; j < countInterval - i - 1; ++j){
            if(arrLeftInt[j] < arrLeftInt[j + 1]) {
                short temp = arrLeftInt[j];
                arrLeftInt[j] = arrLeftInt[j + 1];
                arrLeftInt[j + 1] = temp;
            }
        }
    }


    // равномерно распределим
    for(int i = 0; i < sizeArray; ++i){
        array[i] = xmin + rand() % (xmax - xmin);
    }




    result = new unsigned short[countInterval];
    for(int i = 0; i < countInterval; ++i){
```

```
        result[i] = 0;
    }

    ASSEMBLY_ARR(result, arrLeftInt, array, countInterval, sizeArray);

    //out << "Result:\n" << endl;
   // out << "Left border\tNumber of numbers" << endl;
    cout << "Result:\n" << endl;
    cout << "Left border\t\tNumber of numbers" << endl;

    for (int i = 0; i < countInterval; i++)
    {
        cout << i + 1 << ' ' << arrLeftInt[i] << '\t' << " Count numbers in interval: " <<
result[i] << '\n';
        out << i + 1 << "\t" << arrLeftInt[i] << "; Count numbers in interval: " <<
result[i] << '\n';
    }

    /*for(int i = 0; i < countInterval; ++i){
        out << i + 1 << ' ' << arrLeftInt[i] << ' ' << result[i] << endl;
        cout << i + 1 << ' ' << arrLeftInt[i] << ' ' << result[i] << endl;
    }
    */
    cout << endl;
    cout << "_____  _____" << endl;


    cout << "CHECK_____" << endl << endl;

    for (int i = 0; i < sizeArray - 1; ++i) {
        for (int j = 0; j < sizeArray - i - 1; ++j) {
            if (array[j] < array[j + 1]) {
                short temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }


    for(int i = 0; i < sizeArray; ++i){
        cout << array[i] << ' ';
        out << array[i] << ' ';
    }
```

```
    /*cout << "массив до сортирвоки" << endl;
    for(int i = 0; i < countInterval; ++i){
        cout << arrLeftInt[i] << ' ';
    }

    cout << endl;
    cout << "массив после сортировки" << endl;
    // сортировка
    int temp;
    for(int i = 0; i < countInterval - 1; i++) {
        for(int j = i + 1; j < countInterval; j++) {
            if (arrLeftInt[i] < arrLeftInt[j]) {
                temp = arrLeftInt[i];
                arrLeftInt[i] = arrLeftInt[j];
                arrLeftInt[j] = temp;
            }
        }
    }

    for(int i = 0; i < countInterval; ++i){
        cout << arrLeftInt[i] << ' ';
    }
    */


    delete[] array;
    delete[] arrLeftInt;
    delete[] result;
    out.close();
    return 0;
}
```

Файд lab6.asm

```
.MODEL FLAT, C
 .DATA
 elem DW 0
 .CODE

ASSEMBLY_ARR PROC C
```

```
        mov eax,[esp+4] ; result
        mov ebx,[esp+8] ; arrLeftInt
        mov edx,[esp+12]; array
        mov edi, [esp+16] ; countInterval
        mov cx, [esp+20] ; sizeArray


        and ecx,0000ffffh ; обнуляем верхние 16 бит (счетчики)
        and edi,0000ffffh

func_1:
        push ax                         ; запихивает ax в стек
        mov ax, [edx]                   ; текущий элемент
        mov elem, ax
        pop ax

        mov esi, 0                      ; обнулили индекс результирующих
массивов
        push ecx
        mov ecx, edi                    ; счетчик

func_2:
        push ebx
        mov bx, [ebx+esi]       ; положили элемент из левых границ arrLeftInt
        cmp elem, bx                    ; сравнили с элементом
        jge func_3                      ; если больше или равно, то подходит для
интервала
        add esi, 2                      ; увеличиваем индекс на 2
        pop ebx
        loop func_2                     ; цикл пока не пройдем все текущие
интервалы


func_3:
        mov bx, [eax+esi]       ; взяли элемент из массива количества элементов
        inc bx
        mov [eax+esi],bx        ; увеличиваем кол-во, т.к нашли элемент в
границе
        pop ebx
        pop ecx                         ; счетчик для func_1
        add edx,2                       ; сдвигаемся на 2
        loop func_1                     ; цикл
```

```
ret

ASSEMBLY_ARR ENDP
END
```