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.

Студентка гр. 9383 _____ Лысова А.М,

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Научиться реализовывать связь Ассемблера и ЯВУ. Написать программу построения частотного распределения попаданий псевдослучайных чисел в заданные интервалы.

Задание.

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;

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]).

Для бригад с четным номером: подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде двух ассемблерных модулей, первый из которых формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ как промежуточный результат. Это распределение должно выводиться в текстовом виде для контроля. Затем вызывается второй ассемблерный модуль, который по этому промежуточному распределению формирует окончательное распределение псевдослучайных целых чисел по интервалам произвольной длины (с заданными границами).

Ход работы.

В ходе работы была реализована программа из 3-х модулей, 1 на C++ (ЯВУ) и 2 других на ассемблере.

На ЯВУ написан main.cpp, который собирает от пользователя входную информацию и перенаправляет ее в ассемблерные модули. Так же здесь осуществляется вывод данных в консоль и файл.

На ассемблере написано 2 модуля. Первый реализует распределение чисел(заданных массивом) по единичным отрезкам. Это сделано с помощью зацикливания(loop) действия по записи в новый массив количества повторений каждого числа.

Второй модуль формирует распределение тех же чисел, но уже по заданным интервалам. Это происходит благодаря нескольким циклам, в которых левые границы переводятся в неотрицательные числа и сопоставляются числам с таким же индексом из массива, полученного в первом модуле.

Связь между модулями осуществлена с помощью спецификатора extern, который позволяет выполнять раздельную компиляцию модулей.

Исходный код см. в приложении А.

Тестирование.

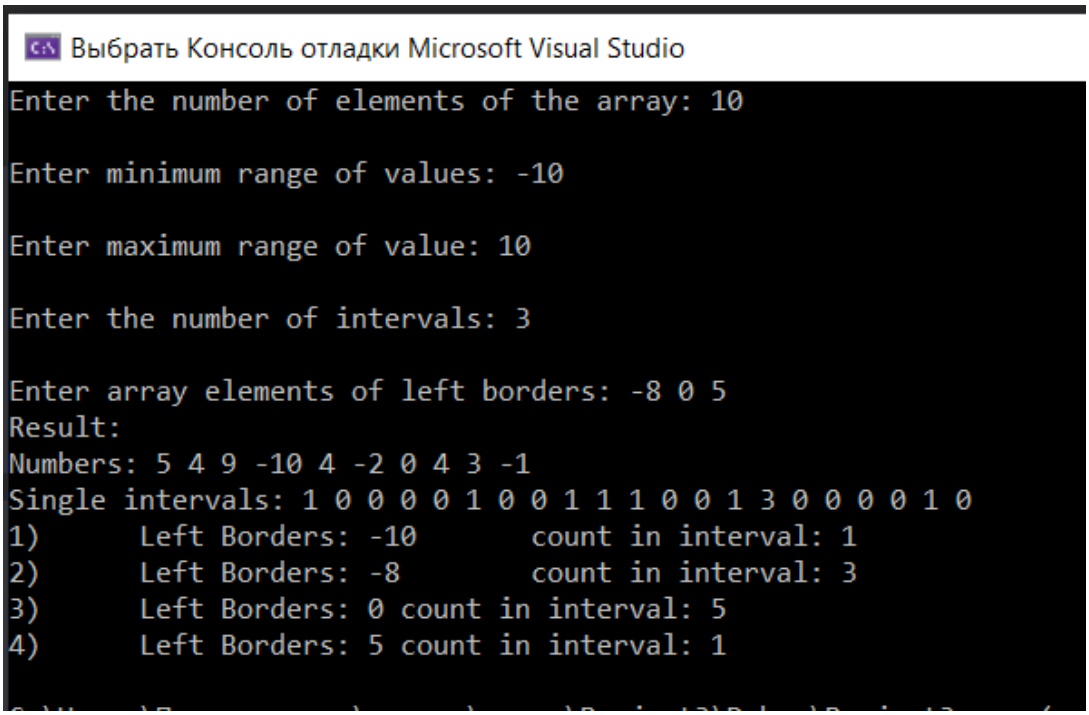

Рисунок 1

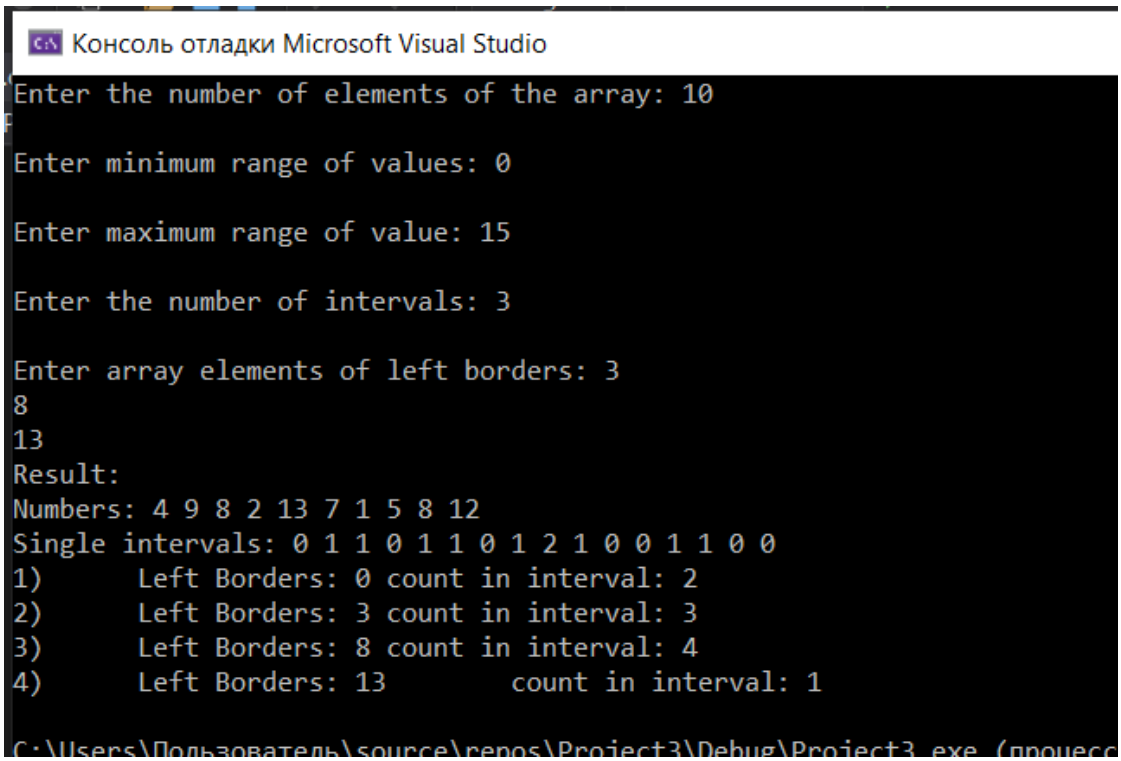

Рисунок 2

Выводы.

Была реализована связь модуля на ЯВУ и модулей на Ассемблере.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД ПРОГРАММЫ.

Название файла: lab6.cpp

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

extern "C" {
    void single(int* result, int* arr, int lenArr, int minX);
    void multiply(int* result, int lenResult, int* leftBorders, int nInt, int
minX, int maxX, int* res);
}

int main() {
    int lenArr;
    int minX, maxX;
    int nInt;

    srand(time(NULL));

    std::cout << "Enter the number of elements of the array: ";
    std::cin >> lenArr;

    if (lenArr > 16 * 1024) {
        std::cout << "\nError of size of the array!\n";
        return 0;
    }

    std::cout << "\nEnter minimum range of values: ";
    std::cin >> minX;

    std::cout << "\nEnter maximum range of value: ";
    std::cin >> maxX;

    std::cout << "\nEnter the number of intervals: ";
    std::cin >> nInt;

    if (nInt <= 0) {
        std::cout << "\nNumber of intervals don't be <= 0!!!\n";
```

```
    return 0;
}

std::cout << "\nEnter array elements of left borders: ";
int* leftBorders = new int[nInt];
for (int i = 0; i < nInt; i++) {
    std::cin >> leftBorders[i];
    if (leftBorders[i] > maxX || leftBorders[i] < minX) {
        std::cout << "\nBorder out of range array!\n";
        return 0;
    }
}

int* arr = new int[lenArr];
for (int j = 0; j < lenArr; j++)
    arr[j] = minX + rand() % (maxX - minX + 1);

int* result = new int[abs(maxX - minX) + 1];
for (int k = 0; k <= abs(maxX - minX); k++)
    result[k] = 0;

int lenResult = abs(maxX - minX) + 1;

single(result, arr, lenArr, minX);

int* res = new int[nInt + 1];

for (int k = 0; k <= nInt; k++)
    *(res + k) = 0;

multiply(result, lenResult, leftBorders, nInt, minX, maxX, res);

std::ofstream file;
file.open("./output.txt");

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

std::cout << "Numbers: ";
file << "Numbers: ";
```

```
for (int i = 0; i < lenArr; i++) {
    std::cout << arr[i] << ' ';
    file << arr[i] << ' ';
}
std::cout << '\n';
file << '\n';

for (int j = 0; j < lenResult; j++) {
    std::cout << result[j] << ' ';
    file << result[j] << ' ';
}

std::cout << '\n';
file << '\n';

for (int i = 1; i < nInt; i++) {
    std::cout << i << ") \t";
    file << i << ") \t";

    std::cout << "Left Border: " << leftBorders[i - 1] << " ";
    file << "Left Border: " << leftBorders[i - 1] << " ";

    int count = 0;
    for (int j = 0; j <= abs(maxX - minX); j++) {
        if (i == nInt)
            if (res[j] >= leftBorders[i - 1]) {
                count++;
                continue;
            }
        if (res[j] >= leftBorders[i - 1] && res[j] < leftBorders[i])
            count++;
    }

    std::cout << "\tcount in interval: " << count << "\n";
    file << "\tcount in interval: " << count << "\n";

}

for (int i = 0; i < abs(maxX - minX); i++) {
    std::cout << arr[i] << ' ';
    file << arr[i] << ' ';
```

```
    }

    std::cout << '\n';

    file.close();

    delete[] leftBorders;
    delete[] arr;
    delete[] result;
    delete[] res;

    return 0;
}
```

Название файла: single.asm

```
.686
.MODEL flat, C
.DATA
.CODE
PUBLIC C single
single PROC C result: dword, arr: dword, lenarr: dword, minx: dword

    push esi
    push edi
    push ebp

    mov edi, result
    mov esi, arr
    mov ecx, lenarr
    mov eax, minx


    for_loop:

    mov ebx, [esi]
    sub ebx, eax
    mov ebp, [edi + 4*ebx]
    inc ebp
    mov [edi + 4*ebx], ebp
    add esi, 4
```

```
    loop for_loop

    pop ebp
    pop edi
    pop esi

    ret

single ENDP
END
```

Название файла: multiply.asm

```
.686
.MODEL flat, C
.DATA
.CODE
PUBLIC C multiply
multiply PROC C result: dword, lenresult: dword, leftborders: dword, nint:
dword, minx: dword, maxx: dword, res: dword

    push esi
    push edi
    push ebp

    mov esi, leftborders
    mov edx, minx
    mov ecx, nint
    mov eax, 0

    for_loop_1:

        mov eax, [esi]
        add eax, maxx
        mov [esi], eax
        mov esi, 4

    loop for_loop_1

    mov edi, result
```

```
mov ecx, nint
mov esi, leftborders
sub ebx, ebx
mov eax, [esi]

for_loop_2:

    push ecx
    mov ecx, eax
    push esi
    mov esi, res

    for_loop_3:

        mov eax, [edi]
        add [esi+4*ebx], eax
        add edi, 4

    loop for_loop_3

    pop esi

    mov eax, [esi]
    add esi, 4
    sub eax, [esi]
    neg eax

    inc ebx
    pop ecx

loop for_loop_2

mov esi, res
mov ecx, nint
sub eax, eax

for_loop_4:

    add eax, [esi]
    add esi, 4
```

```
    loop for_loop_4

    mov esi, res
    sub eax, lenresult
    neg eax
    add [esi+4*ebx], eax

    pop ebp
    pop edi
    pop esi

multiply ENDP
END
```