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 распределения попаданий псевдослучайных целых чисел в заданные интервалы

Студент гр. 9382 _____ Субботин М.О.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

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

Основные теоретические положения.

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.

Ход выполнения:

В C++ вводятся и подготавливаются данные. Основную работу выполняет метод, написанный на языке Ассемблера.

В Ассемблере реализуется метод, который проходится по всему массиву чисел и определяет их в нужный интервал и увеличивает количество элементов в интервале на единицу.

Результаты работы программы выводятся в файл.

Исходный код программы:

Cpp-файл:

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

```
#include <random>
using namespace std;
int64_t getRandomNumber(int64_t min, int64_t max)
{
    std::random_device rd;
    std::mt19937 mt(rd());
    std::uniform_int_distribution<int> dist(min, max);
    return dist(mt);
}

extern "C"
{
    void MAS_INTERVAL(int64_t* left_boarders, int64_t* res_arr, int64_t*
arr, int64_t array_size);
}

int main()
{
    srand(static_cast<unsigned int>(time(0)));
    int64_t array_size = 0;
    cout << "Введите длину массива: ";
    cin >> array_size;
    if (array_size > 16 * 1024) {
        cout << "Слишком много элементов!";
        return 0;
    }

    int64_t x_min = 0;
    cout << "Введите нижний диапазон: ";
    cin >> x_min;
```

```
int64_t x_max = 0;
cout << "Введите верхний диапазон: ";
cin >> x_max;

int64_t intervals_number = 0;
cout << "Введите количество диапазонов(<=24): ";
cin >> intervals_number;
if (intervals_number > 24) {
   cout << "Диапазон слишком много!";
   return 0;
}

int64_t *left_boarders = new int64_t[intervals_number];
cout << "Введите " << intervals_number - 1 << " нижних границ интервалов ";
for (int64_t i = 0; i < intervals_number - 1; i++) {
   cin >> left_boarders[i];
   if (left_boarders[i] > x_max || left_boarders[i] < x_min) {
      cout << "Введеное граница не входит в заданные промежутки!";
      return 0;
   }
}
left_boarders[intervals_number - 1] = x_max;

int64_t *arr = new int64_t[array_size];
for (int64_t i = 0; i < array_size; i++) {
   arr[i] = getRandomNumber(x_min, x_max);
}
```

```
        int64_t *res_arr = new int64_t[intervals_number];
        for (int64_t i = 0; i < intervals_number; i++) {
            res_arr[i] = 0;
        }

        MAS_INTERVAL(left_boarders, res_arr, arr, array_size);

        ofstream myfile("out.txt", std::ios::out);
        for (int64_t i = 0; i < array_size; i++) {
            cout << arr[i] << " ";
            myfile << arr[i] << " ";
        }
        myfile << endl;

        if (myfile) {
            myfile << "N_interval\tL_borders\tN_number\n";
            for (int64_t i = 0; i < intervals_number; i++) {
                int64_t res = i != 0 ? left_boarders[i - 1] : x_min;
                myfile << "     " << i+1 << "\t\t     " << res << "\t\t    " << res_arr[i] <<
endl;
            }
        }
        cout << "\nN_interval\tL_borders\tN_number\n";
        for (int64_t i = 0; i < intervals_number; i++) {
                int64_t res = i != 0 ? left_boarders[i - 1] : x_min;
                cout << "    " << i+1 << "\t\t    " << res << "\t\t   " << res_arr[i] << endl;
            }
    }
```

Asm-файл:

```
.intel_syntax noprefix
```

```
.global _MAS_INTERVAL
.text
_MAS_INTERVAL: # edi : left_borders, esi : res_arr , edx : arr, ecx : array_size
        mov rax,rcx
        mov rcx, 0 # счетчик для прохода по массиву чисел
        mov rbx, rdx # ebx указывает на начало массива чисел arr

        traverse_numbers:
                push rax
                mov rax,[rbx] #в eax лежит текущий элемент
                push rbx #сохраняем указатель на текущий элемент
                mov rbx,0 #обнуляем указатель

        traverse_borders: #здесь ebx - счетчик границ
                mov rdx,rbx # в edx лежит текующий индекс массива границ
                shl rdx,3 # этот индекс умножаем на 8, т.е. каждый элемент по 8 байт
                cmp rax,[rdi+rdx] # сравниваем текующий элемент с текующей левой границей (left_boarders + 4*i), i -номер элемента
                jle matched_interval # если число меньше либо равно левой границе, то идем в matched_interval

                inc rbx # инкрементируем указатель
                jmp traverse_borders # т.к. наше число больше левой

        matched_interval:
                add rdx,rsi # edx - сдвиг для left_boarders, после сложения edx указывает на элемент в res_arr который нужно инкрементировать
```

```
        mov rax,[rdx] #достаем количество подходящей левой границы
        inc rax #прибавляем к ней единицу
        mov [rdx],rax #вставляем ее обратно

        pop rbx #достаем текущий сдвиг для массива чисел

        add rbx,8 #перемещаем указатель на следующий элемент массива чисел
        inc rcx #инкрементируем количество разобранных элементов
        pop rax
        cmp rcx,rax #смотрим, рассмотрели ли мы все элементы
        jl traverse_numbers #если еще не все, то продолжаем
    ret
```

Тестирование.

№	Входные данные	Выходные данные
1	92 -77 -43 -16 90 -23 -96 56 13 -64	N_interval L_borders N_number 1 -100 3 2 -50 4 3 50 3
2	97 56 64 53 86 29 92 11 22 48 17 61 63 56 42 64 70 47 59 46	N_interval L_borders N_number 1 0 0 2 10 8 3 50 12
3	936 199 238 830 -665 768 -729 -590 880 -671	N_interval L_borders N_number 1 -1000 0 2 -750 4 3 -500 2 4 500 0 5 750 4

Выводы.

Была изучена организация связи Ассемблера с ЯВУ на примере программы построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.