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.

Студент(ка) гр. 9382 _____ Голубева В.П.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Научиться создавать программы на ЯВУ, используя в них ассемблерный модуль.

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.

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.

В программе на C++ происходит ввод данных.

Обработка введённых данных происходит в функции, которая написана на Ассемблере. Там реализуется функция, которая проходится по массиву

чисел и определяет их в нужный интервал, увеличивая количество элементов в интервале на единицу. Результаты работы программы выводятся в файл.

Тестирование.

Тестирование представлено в таблице 1.

Таблица 1. Тестирование программы

Номер	Входные данные	Выходные данные
1.	Введите длину массива: 34 Введите нижний диапазон: 34 Введите верхний диапазон: 87 Введите количество диапазонов(<=24): 2 Введите 1 нижних границ интервалов 50 Сгенерированные псевдослучайные числа: 82 36 55 82 66 81 41 58 40 63 76 82 51 48 69 48 55 46 63 58 70 36 76 64 86 47 83 82 55 54 70 82 57 70	Numer_interval Left_borders Count_number 1 34 8 2 50 26
2.	Введите длину массива: 3 Введите нижний диапазон: 10 Введите верхний диапазон: 20 Введите количество диапазонов(<=24): 5 Введите 4 нижних границ интервалов 13 15 17 19 Сгенерированные псевдослучайные числа: 13 16 17	Numer_interval Left_borders Count_number 1 10 1 2 13 0 3 15 2 4 17 0 5 19 0

3.	Введите длину массива: 10 Введите нижний диапазон: 3 Введите верхний диапазон: 30 Введите количество диапазонов(<=24): 7 Введите 6 нижних границ интервалов 4 8 12 16 20 24 28 Сгенерированные псевдослучайные числа: 22 19 21 28 17 28 22 15 9 13	Numer_interval Count_number		Left_borders
		1	3	0
		2	4	0
		3	8	1
		4	12	2
		5	16	2
		6	20	3
		7	24	2
4.	Введите длину массива: 15 Введите нижний диапазон: 1 Введите верхний диапазон: 90 Введите количество диапазонов(<=24): 5 Введите 4 нижних границ интервалов 6 12 24 48 Сгенерированные псевдослучайные числа: 38 34 42 29 20 78 76 67 24 49 60 23 36 75 33	Numer_interval Count_number		Left_borders
		1	1	0
		2	6	0
		3	12	3
		4	24	6
		5	48	6

Выводы.

Было освоено инегрирование программ на ЯВУ и ассемблерных модулей. Была разработана программа, которая строит частотное распределение попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: c.cpp

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

int64_t getRandomNumber(int64_t min, int64_t max)
{
    return min+rand()%(max-min);
}

extern "C" void DISTRIBUTION_INTERVAL(int64_t* left_boarders, int64_t* res_array, int64_t* array, int64_t size);

int main()
{
    int64_t size = 0;
    std::cout << "Введите длину массива: ";
    std::cin >> size;
    while (size > 16 * 1024) {
          std::cout << "Слишком много элементов! Введите длину, которая меньше или равно 16*1024\n";
        std::cin>>size;
    }

    int64_t Xmin = 0;
    std::cout << "Введите нижний диапазон: ";
    std::cin >> Xmin;

    int64_t Xmax = 0;
    std::cout << "Введите верхний диапазон: ";
    std::cin >> Xmax;

    int64_t intervals_count = 0;
```

```
        std::cout << "Введите количество диапазонов(<=24): ";
        std::cin >> intervals_count;
        while (intervals_count > 24) {
                  std::cout << "Диапазонов слишком много! Введите количество интервалов, которое меньше или равно 24\n";
            std::cin>>intervals_count;
        }

        int64_t *left_boarders = new int64_t[intervals_count];
         std::cout << "Введите " << intervals_count - 1 << " нижних границ интервалов ";
        for (int64_t i = 0; i < intervals_count - 1; i++) {
            std::cin >> left_boarders[i];
               while (left_boarders[i] > Xmax || left_boarders[i] < Xmin) {
                 std::cout << "Введеная граница "<<left_boarders[i]<<" не входит в заданные промежутки! Введите снова\n";
                 std::cin>>left_boarders[i];
            }
        }
        left_boarders[intervals_count - 1] = Xmax;

        int64_t *array = new int64_t[size];
        for (int64_t i = 0; i < size; i++) {
            array[i] = getRandomNumber(Xmin, Xmax);
        }

        int64_t *res_array = new int64_t[intervals_count];
        for (int64_t i = 0; i < intervals_count; i++) {
            res_array[i] = 0;
        }

        DISTRIBUTION_INTERVAL(left_boarders, res_array, array, size);

        std::ofstream res_file("res.txt");
        std::cout<<"Сгенерированные псевдослучайные числа: ";
        res_file<<"Сгенерированные псевдослучайные числа: ";
```

```
        for (int64_t i = 0; i < size; i++) {
            std::cout << array[i] << " ";
            res_file << array[i] << " ";
        }
        res_file <<"\n";
        std::cout<<"\n";
          std::cout  <<  "\nNumer_interval\tLeft_borders\tCount_number\
n";
        res_file << "\nNumer_interval\tLeft_borders\tN_number\n";
        for (int64_t i = 0; i < intervals_count; i++) {
                int64_t res = i != 0 ? left_boarders[i - 1] : Xmin;
                 res_file << "    " << i+1 << "\t\t    " << res << "\
t\t   " << res_array[i] << "\n";
                 std::cout << "    " << i+1 << "\t\t    " << res << "\
t\t   " << res_array[i] << "\n";
             }
     }
```

Название файла: a.asm

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

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

```
        passing_borders:    # ebx - счетчик границ
                mov rdx,rbx     # в edx лежит текущий индекс массива границ
                shl rdx,3       # этот индекс умножаем на 8, т.е. каждый элемент по 8 байт
                cmp rax,[rdi+rdx]    # сравниваем текущий элемент с текущей 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
                jmp passing_borders     # т.к. наше число больше левой границы

        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                 # увеличиваем количетво обработанных элементов
                pop rax
                cmp rcx,rax             #смотрим, рассмотрели ли мы все элементы
```

```
                jl  passing_numbers            #если  еще  не  все,  то
продолжаем
           ret
```