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 _____ Гладких А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Применить на практике знания по организации связи Ассемблера с ЯВУ. Написать программу, строящую частотное распределение попаданий псевдослучайных целых чисел в заданные интервалы.

Текст задания.

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.

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). Это распределение возвращается в головную

программу и выдается как основной результат в виде текстового файла и, возможно, графика.

Ход работы.

В ходе работы была разработана программа, которая строит частотное распределение попаданий псевдослучайных целых чисел в заданные интервалы.

Сперва, в функции main() файла Source.cpp происходит считывание исходных данных - длина массива псевдослучайных целых чисел, диапазон изменения массива псевдослучайных целых чисел, количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел и массив левых границ интервалов разбиения.

После этого происходит вызов ассемблерной функции, связанной спецификатором extern, asm_mod1, которая считает количество повторений чисел с помощью цикла loop.

После этого происходит подсчет чисел в заданных интервалах с помощью ассемблерной функции asm_mod2, которая делает это с помощью двух циклов loop.

После завершения работы функции asm_mod2, полученные результаты выводятся на экран и в файл output.txt. Функции main() завершается, предварительно удалив выделенную динамически память.

Исходный код программы представлен в приложении А.

Примеры работы программы.

Таблица 1 — Примеры работы программы.

№	Входные данные	Выходные данные
1	Введите количество цифр: 10 Введите xmin и xmax: -5 5 Введите число левых границ: 3 Введите все левые границы: -3 0 2	Рандомизированные значения -1 3 -1 -1 -3 2 4 -5 -1 2 Подсчитаем количество повторений каждого отдельного числа: 1 0 1 0 4 0 0 2 1 1 0 Результат: № Лев.Гр. Кол-во чисел 1 -5 1 2 -3 5 3 0 0 4 2 4
2	Введите количество цифр: 10 Введите xmin и xmax: -10 10 Введите число левых границ: 3 Введите все левые границы: 1 3 4	Рандомизированные значения 7 7 -7 7 9 8 9 -5 -7 -10 Подсчитаем количество повторений каждого отдельного числа: 1 0 0 2 0 1 0 0 0 0 0 0 0 0 0 0 0 3 1 2 0 Результат: № Лев.Гр. Кол-во чисел 1 -10 4 2 1 0 3 3 0 4 4 6
3	Введите количество цифр: 100 Введите xmin и xmax: -1 1	Рандомизированные значения 0 0 -1 -1 0 0 0 -1 -1 -1 0 -1 -1 -1 -1 -1 0 -1 -1

	Введите число левых границ: 1 Введите все левые границы: 0	0 -1 -1 0 0 -1 0 0 -1 0 -1 0 -1 0 0 0 -1 0 -1 0 -1 0 0 -1 0 -1 0 0 -1 -1 -1 0 -1 0 0 0 0 0 0 -1 -1 0 0 0 0 -1 0 0 -1 0 0 0 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 0 0 0 0 0 -1 -1 0 0 -1 0 0 0 0 0 0 0 -1 Подсчитаем количество повторений каждого отдельного числа: 45 55 0 Результат: № Лев.Гр. Кол-во чисел 1 -1 45 2 0 55

Выводы.

Были применены на практике знания по организации связи Ассемблера с ЯВУ. Была написана программа, строящая частотное распределение попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: source.cpp

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

using namespace std;

extern "C" void asm_mod1(int* numbers, int numbers_size, int* res, int
res_xmin);
extern "C" void asm_mod2(int* input, int input_size, int input_xmin, int
input_xmax, int* intervals, int intervals_size, int* res_final);


int main() {
      setlocale(0, "Russian");
      srand(time(NULL));
      ofstream result("result.txt");

      int amountNumbers = 0;
      int xmin, xmax;
      int* numbers;
      int* bordersArr;
      int numberOfIntervals;
      int* asm_mod1_res;
      int* asm_mod2_res;

      cout << "Введите количество цифр: ";
      cin >> amountNumbers;
      cout << "Введите xmin и xmax: ";
      cin >> xmin >> xmax;
      cout << "Введите число левых границ: ";
      cin >> numberOfIntervals;

      numbers = new int[amountNumbers];
      bordersArr = new int[numberOfIntervals];

      int len_asm_mod1_res = abs(xmax - xmin) + 1;
      asm_mod1_res = new int[len_asm_mod1_res];
      for (int i = 0; i < len_asm_mod1_res; i++) {
            asm_mod1_res[i] = 0;
      }

      asm_mod2_res = new int[numberOfIntervals + 1];
      for (int i = 0; i < numberOfIntervals + 1; i++) {
            asm_mod2_res[i] = 0;
      }

      cout << "Введите все левые границы: ";
```

```
	for (int i = 0; i < numberOfIntervals; i++) {
		cin >> bordersArr[i];
	}


	for (int i = 0; i < amountNumbers; i++) {
		numbers[i] = xmin + rand() % (xmax - xmin);
	}
	cout << '\n';

	cout << "Рандомизированные значения\n";
	result << "Рандомизированные значения\n";
	for (int i = 0; i < amountNumbers; i++) {
		cout << numbers[i] << ' ';
		result << numbers[i] << ' ';
	}
	cout << '\n';
	cout << '\n';
	result << '\n';
	result << '\n';

	cout << "Подсчитаем количество повторений каждого отдельного
числа:\n";
	result << "Подсчитаем количество повторений каждого отдельного
числа:\n";

	asm_mod1(numbers, amountNumbers, asm_mod1_res, xmin);

	for (int i = 0; i < len_asm_mod1_res; i++) {
		cout << asm_mod1_res[i] << ' ';
		result << asm_mod1_res[i] << ' ';
	}
	cout << '\n';
	cout << '\n';
	result << '\n';
	result << '\n';

	asm_mod2(asm_mod1_res, amountNumbers, xmin, xmax, bordersArr,
numberOfIntervals, asm_mod2_res);


	cout << "Результат:\n";
	result << "Результат:\n";
	cout << "№\tЛев.Гр.\tКол-во чисел" << endl;
	result << "№\tЛев.Гр.\tКол-во чисел" << endl;
	cout << "1" << "\t" << xmin << '\t' << asm_mod2_res[0] << endl;
	result << "1" << "\t" << xmin << '\t' << asm_mod2_res[0] << endl;
	for (int i = 0; i < numberOfIntervals; i++) {
		cout << i + 2 << "\t" << bordersArr[i] - xmax << '\t' <<
asm_mod2_res[i + 1] << endl;
		result << i + 2 << "\t" << bordersArr[i] - xmax << '\t' <<
asm_mod2_res[i + 1] << endl;

	}

	delete[] numbers;
	delete[] bordersArr;
	delete[] asm_mod1_res;
```

```
delete[] asm_mod2_res;
return 0;
}
```

Название файла: asm_mod1.asm

```
.586p
.MODEL FLAT, C
.CODE
PUBLIC C asm_mod1
asm_mod1  PROC  C  numbers:  dword,  numbers_size:  dword,  res:  dword,
res_xmin: dword

push esi
push edi

mov edi, numbers
mov ecx, numbers_size
mov esi, res

for_numbers:
mov eax, [edi]
sub eax, res_xmin
mov ebx, [esi + 4*eax]
inc ebx
mov [esi + 4*eax], ebx
add edi, 4
loop for_numbers

pop edi
pop esi

ret
asm_mod1 ENDP
END
```

Название файла: asm_mod2.asm

```
.586p
.MODEL FLAT, C
.CODE
PUBLIC C asm_mod2
asm_mod2 PROC C input: dword, init_size: dword, input_xmin: dword, input_xmax: dword, intervals: dword, intervals_size: dword, res_final: dword

push esi
push edi
push ebp

mov edi, input

mov esi, intervals
mov ecx, intervals_size


align_intervals:
mov eax, [esi]
add eax, input_xmax
mov [esi], eax
add esi, 4
loop align_intervals

mov esi, intervals
mov ecx, intervals_size
sub ebx, ebx

mov eax, [esi]

for_loop:

push ecx

mov ecx, eax
```

```
push esi

mov esi, res_final

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

    pop esi
    inc ebx

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


pop ecx

loop for_loop

mov esi, res_final
mov ecx, intervals_size
sub eax, eax

final_interval:
add eax, [esi]
add esi, 4
loop final_interval

mov esi, res_final
sub eax, init_size
neg eax

add [esi + 4 * ebx], eax

pop ebp
```

```
pop edi
pop esi

ret
asm_mod2 ENDP
END
```