МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения 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 гр. 9383 Поплавский И.

Преподаватель Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

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

Постановка задачи.

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение.

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]).

Результаты:

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

2. График, отражающий распределение чисел по интервалам. (необязательный результат)

Для бригад с нечетным номером: 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.

Сначала происходит инициализация всех данных в C++ (запрашиваем у пользователя длину массива, xmin, xmax, количество интервалов).

После равномерно распределяем числа и передаем все данные в асм модуль. Считаем количество попаданий в каждый интервал и убираем результаты в массив res.

Возвращаемся в C++ и выводим результаты.

Тестирование:

№	Исходные данные	Результат		
1	NumDatRan=100 xmin=-50 xmax=50 NInt=10 LGrInt={-50, -43, -30, 0, 10, 12, 15, 30, 40, 49 }	№ 1 2 3 4 5 6 7 8 9 10	Лев.гр. -50 -43 -30 0 10 12 15 30 40 49	Кол-во чисел 4 14 28 7 1 6 18 11 11 0

2	NumDatRan=40 xmin=-20 xmax=20 NInt=4 LGrInt={-20, -10, 0, 10 }	№ 1 2 3 4	Лев.гр. -20 -10 0 10	Кол-во чисел 15 7 11 7
3	NumDatRan=1100 xmin=0 xmax=100 NInt=7 LGrInt={0, 5, 23, 56, 70, 75, 90}	№ 1 2 3 4 5 6 7	Лев.гр. 0 5 23 56 70 75 90	Кол-во чисел 48 211 360 149 56 155 121
4	NumDatRan=80 xmin=0 xmax=10 NInt=5 LGrInt={ 0, 2, 4, 6, 8}	№ 1 2 3 4 5	Лев.гр. 0 2 4 6 8	Кол-во чисел 14 13 13 15 25
5	NumDatRan=16000 xmin=-8 xmax=8 NInt=6 LGrInt={ -8 -4 -2 0 3 6}	№ 1 2 3 4 5 6	Лев.гр. -8 -4 -2 0 3 6	Кол-во чисел 3956 1982 2071 2983 3002 2006

Вывод.

В результате выполнения работы была освоена организация связи ЯВУ с Ассемблером, построения частотного распределения попаданий псевдослучайных чисел в заданные интервалы, были улучшены навыки программирования на Ассемблере.

ПРИЛОЖЕНИЕ А

РАЗРАБОТАННЫЙ КОД ПРОГРАММЫ

Файл Source.cpp

```
#include <iostream>
#include <fstream>
#include <windows.h>
#include <random>
#include <clocale>

using namespace std;

extern "C" { //подключаем модуль на языке ассемблера
	void MYASM(short int* arr, short int* LGrInt, unsigned short* res, unsigned short
NInt, unsigned short NumRanDat);
}

int main() {
	unsigned short int NumRanDat = 0; //длина массива псевдослучайных целых чисел
	short int xmin = 0, xmax = 0; //границы диапазона псевдослучайных чисел
	short int* arr; //массив псевдослучайных чисел
	unsigned short int NInt; //количество интервалов
	short int* LGrInt; //массив левых границ интервалов
	unsigned short int* res;//массив с количеством чисел в каждом интервале

	ofstream result("result.txt");

	setlocale(LC_CTYPE, "rus");

	do {
		cin.clear();
		cin.sync();
		cout << "Введите длину массива псевдослучайных чисел (<=16000): ";
		cin >> NumRanDat;
		cout << endl;
	} while (NumRanDat > 16000 || NumRanDat < 0);

	do {
		cin.clear();
		cin.sync();
		cout << "Введите xmin и xmax: ";
		cin >> xmin >> xmax;
		cout << endl;
	} while (xmax <= xmin);

	do {
		cin.clear();
		cin.sync();
		cout << "Введите количество интервалов (<=24): ";
		cin >> NInt;
	} while (NInt > 24 || NInt < 1);

	arr = new short int[NumRanDat];
	LGrInt = new short int[NInt];



	cout << "\nВведите левые границы интервалов:\n";
	cout << "1: " << xmin << endl << endl; //первая левая граница - начало диапазона,
т.е. xmin
	LGrInt[0] = xmin;
```

```
	//ввод остальных границ
	for (int i = 1; i < NInt; i++) {
		do {
			cin.clear();
			cin.sync();
			cout << i + 1 << ": ";
			cin >> LGrInt[i];
			cout << endl;
		} while (LGrInt[i]<xmin || LGrInt[i]>xmax);
	}

	//сортировка массива границ (по убыванию, т.к в asm модуль передаем инвертированный
массив)
	for (int j = 0; j < NInt - 1; j++) {
		for (int i = 0; i < NInt - j - 1; i++) {
			if (LGrInt[i] < LGrInt[i + 1]) {
				int b = LGrInt[i];
				LGrInt[i] = LGrInt[i + 1];
				LGrInt[i + 1] = b;
			}
		}
	}

	//равномерное распределение
	for (int i = 0; i < NumRanDat; i++) {
		for (int i = 0; i < NumRanDat; i++)
			arr[i] = xmin + rand() % (xmax - xmin);
	}

	res = new unsigned short int[NInt];
	for (int i = 0; i < NInt; i++) {
		res[i] = 0;
	}

	MYASM(arr, LGrInt, res, NInt, NumRanDat);

	cout << "result:\n";
	result << "result:\n";
	cout << "№\tЛев.Гр.\tКол-во чисел" << endl;
	result << "№\tЛев.Гр.\tКол-во чисел" << endl;
	for (int i = 0; i < NInt; i++) {
		cout << i + 1 << '\t' << LGrInt[NInt - 1 - i] << '\t' << res[NInt - 1 - i] <<
endl;
		result << i + 1 << '\t' << LGrInt[NInt - 1 - i] << '\t' << res[NInt - 1 - i]
<< endl;
	}
	cout << endl << endl;

	//сортировка сгенерированных чисел (для проверки)
	for (int j = 0; j < NumRanDat - 1; j++) {
		for (int i = 0; i < NumRanDat - j - 1; i++) {
			if (arr[i] > arr[i + 1]) {
				int b = arr[i];
				arr[i] = arr[i + 1];
				arr[i + 1] = b;
			}
		}
	}

	/*cout << "Сгенерированные числа: ";
	for (int q = 0; q < NumRanDat; q++){
	cout << arr[q] << " ";
```

```
	}*/

	cin.clear();
	cin.sync();
	cin.get();
	result.close();
	delete[] arr;
	delete[] LGrInt;
	delete[] res;
	return 0;

}
```

Файл assembler.asm

```
.MODEL FLAT, C
.CODE
MYASM PROC C
		push esi
		push edi
		push ebp

		mov eax, [esp+16]	;*arr
		mov edx, [esp+20]	;*LGrInt
		mov ebx, [esp+24]	;*res

		mov si, [esp+28]	;NInt
		and esi, 0000ffffh
		mov cx, [esp+32]	;NumRanDat
		and ecx, 0000ffffh
		xor ebp, ebp			;тоже самое что mov ebp,0

	for1:
		push ecx			;убираем итератор первого цикла в
стек
		mov cx,si			;заменяем на итератор второго
цикла
		mov bp, [eax]			;берем число из arr
		and ebp, 0000ffffh
		xor edi,edi			;тоже самое что mov edi,0

		for2:
			cmp bp, [edx+edi] ;сравниваем число и левую границу
			jge quit		;если число больше, значит входит
в интервал
```

```
                add edi,2               ;переход к след. интервалу
(увеличиваем смещение на два, т.к. int - 2 байта)
            loop for2                   ;в цикле, пока не пройдем по всем
интервалам

      quit:                             ;увеличение кол-ва попаданий в
интервал
            push edx                    ;сохраняем edx в стеке
            mov edx, [ebx+edi]          ;берем соответствующее интервалу
количество попаданий
            inc edx                             ;увеличиваем это число на 1
            mov [ebx+edi],edx       ;убираем его обратно в массив res
            pop edx                             ;возвращаем из стека edx
            pop ecx                             ;возвращаем     из     стека
итератор первого цикла
            add eax,2                   ;переход к следующему числу
      loop for1                         ;в цикле пока не пройдем по всем
числам

            pop ebp
            pop edi
            pop esi

            ret

MYASM ENDP
END
```