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 распределения попаданий псевдослучайных целых чисел в заданные интервалы.

Студент гр. 9382 _____ Докукин В.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить взаимодействие программы на ЯВУ с модулем на языке Ассеблера, написать ассемблерный модуль.

Задание:

7 Вариант – нечетный

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 целых чисел - 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)

В зависимости от номера бригады формирование частотного распределения должно производиться по одному из двух вариантов:

1. 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;

Ход работы:

В качестве ЯВУ использован язык C++. Вначале производится ввод необходимых данных, затем вызывается функция, определенная в модуле module.asm. В ней происходит обработка массива и выполняются необходимые действия для получения массива-ответа. В массиве-ответе хранится количество чисел, попавших в заданные диапазоны. Далее происходит вывод значений-ответов на экран и в файл.

Тестирование.

Входные данные	Выходные данные
Enter random nubmers array length: 20 Enter lower bound: 1 Enter higher bound:	# of intervals lower bound count 1 20 16 2 40 10 3 60 8

100 Enter number of ranges(<= 24): 5 Enter 5 lower bounds of intervals: 20 40 60 80 100	4 80 8 5 100 0

Выводы.

В результате выполнения лабораторной работы был написан код, состоящий из модуля на языке Ассемблера и остальной части на ЯВУ C++, который рассчитывает и выводит на экран и в файл частоту попадания псевдослучайных чисел в определенные пользователем диапазоны.

Приложение.

Имя файла: ConsoleApplication9.cpp

```
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <ctime>
#include <random>

extern "C" // C functions usage
{
	void _form_array(int NumRanDat, int numRanges, int* arr, int* LGrInt,
int* res_arr);
}

int main()
{
	srand(time(0));

	int NumRanDat = 0;
	std::cout << "Enter random nubmers array length:\n";
	std::cin >> NumRanDat;
	if (NumRanDat > 16 * 1024) {
		NumRanDat = 16 * 1024;
	}
	int Xmin = 0, Xmax = 0, NInt = 0;
	std::cout << "Enter lower bound:\n";
	std::cin >> Xmin;
	std::cout << "Enter higher bound:\n";
	std::cin >> Xmax;
	std::cout << "Enter number of ranges(<= 24): ";
	std::cin >> NInt;
	if (NInt > 24) {
		NInt = 24;
	}
	int* LGrInt = new int[NInt];
	std::cout << "Enter " << NInt << " lower bounds of intervals:\n";
	for (int i = 0; i < NInt; i++)
	{
		std::cin >> LGrInt[i];
		if (i != 0) while (LGrInt[i] < LGrInt[i - 1])
		{
			std::cout << "Entered bound " << LGrInt[i] << " is lower
than previous, try again.\n";
			std::cin >> LGrInt[i];
		}
		while (LGrInt[i] < Xmin || LGrInt[i] > Xmax)
		{
			std::cout << "Entered bound " << LGrInt[i] << " is not
included in the specified intervals, try again.\n";
			std::cin >> LGrInt[i];
		}
	}
```

```
	// End of input

	int* arr = new int[NumRanDat]();

	for (int i = 0; i < NumRanDat; i++)
	{
		int r = rand();
		arr[i] = Xmin + r % (Xmax - Xmin);
	}

	int* res_arr = new int[NInt];
	for (int i = 0; i < NInt; i++)
		res_arr[i] = 0;
	_form_array(NumRanDat, NInt, arr, LGrInt, res_arr); // Assembler

	// Output
	std::ofstream file("res.txt");
	std::cout << "Generated pseudo-random numbers:\n";
	file << "Generated pseudo-random numbers:\n";
	for (int i = 0; i < NumRanDat; i++)
	{
		std::cout << arr[i] << " ";
		file << arr[i] << " ";
	}
	std::cout << "\n";
	file << "\n";
	std::cout << "\n# of intervals\tlower bound\tcount\n";
	file << "\n# of intervals\tlower bound\tcount\n";
	for (int i = 0; i < NInt; i++) {
		int res = LGrInt[i];
		file << "	" << i + 1 << "\t\t	" << res << "\t\t	" <<
res_arr[i] << "\n";
		std::cout << "	" << i + 1 << "\t\t	" << res << "\t\t	"
<< res_arr[i] << "\n";
	}
	return 0;
}
```

Имя файла: module.asm

```
.686
.MODEL FLAT, C
.STACK
.DATA
.CODE
_form_array PROC C NumRanDat:dword, numRanges:dword, arr:dword, LGrInt:dword,
res_arr:dword ; Получаем данные из программы высокого уровня

mov ecx, 0 ; счетчик для прохода по массиву
mov ebx, [arr] ; массив случайных чисел
mov esi, [LGrInt] ; массив с левыми границами
mov edi, [res_arr] ; массив-результат

@begin:
	mov eax, [ebx] ; берем элемент входного массива
	push ebx  ; сохраняем указатель на текущий элемент
	mov ebx, 0 ; обнуляем указатель
```

```
@move:
      mov edx, ebx ; edx содержит текущий индекс массива границ
      shl edx, 2 ; индекс умножаем на 4, т.к. каждый элемент состоит из 4
байт
      cmp eax, [esi+edx] ; сравниваем текущий элемент с текущей левой
границей
      jge @is_in
      jmp @continue

@is_in:
      add edx, edi; в массиве-ответе инкерементируем счетчик
      push eax
      mov eax, [edx]
      inc eax
      mov [edx], eax
      pop eax
      jmp @continue

@continue: ; продолжаем обход массива границ
      inc ebx
      cmp ebx, numRanges
      jle @move
      jmp @end

@end:
      pop ebx  ; забираем текущий элемент и ссылаемся на новый
      add ebx, 4
      inc ecx ; инкрементируем счетчик
      cmp ecx, NumRanDat
    jl @begin

ret

_form_array ENDP
END
```