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.
Вариант 8

Студент гр. 0382 _____ Кондратов Ю.А.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2021

Цель работы.

Изучение основных принципов организации связи Ассемблера с ЯВУ. Написание программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы.

Задание.

На языке С программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих заданный закон распределения. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND_GEN (при его отсутствии получить у преподавателя).

Следует привести числа к целому виду с учетом диапазона изменения.

Далее должны вызываться 1 или 2 ассемблерных процедуры для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Ассемблерные процедуры должны вызываться как независимо скомпилированные модули. Передача параметров в процедуру должна выполняться через кадр стека.

Результирующий массив частотного распределения чисел по интервалам, сфор-мированный 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)
2. Диапазон изменения массива псевдослучайных целых чисел
   [Xmin, Xmax] (м.б. биполярный, например, [-100, 100])
3. Массив псевдослучайных целых чисел {Xi}.
4. Количество интервалов, на которые разбивается диапазон
   изменения массива псевдослучайных целых чисел - NInt (<=24)
5. Массив левых границ интервалов разбиения LGrInt .

В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину, левые границы могут

задаваться в произвольном порядке и иметь произвольные значения. Если Xmin < LGrInt(1), то часть данных не будет участвовать в формировании распределения. Каждый интервал, кроме последнего, следует интерпретировать как [LGrInt(i), LGrInt(i+1)). Если у последнего интервала правая граница меньше Xmax, то часть данных не будет участвовать в формировании распределения.

Результаты:

Текстовая таблица, строка которой содержит:

- номер интервала,
- левую границу интервала,
- количество псевдослучайных чисел, попавших в интервал.

Количество строк должно быть равно числу интервалов разбиения.

Таблица должна выводиться на экран и сохраняться в файле.

Задание на разработку программы выбирается из таблицы 1 в зависимости от номера студента в группе.

Замечания:

1) На ЯВУ следует реализовать только ввод исходных данных (возможно с контролем), вывод и генерацию псевдослучайных целых чисел. Всю остальную функциональность следует программировать на ассемблере.

2) В отладочной версии программы (при небольшом количестве псевдослучайных чисел, не превышающем 100 значений) для контроля работы датчика сгенериро-ванные числа, приведенные к целому виду, следует выводить на экран или в файл. В основной версии программы, предоставляемой для защиты, вывод сгенерированных псевдослучайных чисел выполнять не нужно.

Вариант 8:

№	Вид распредения	Число ассем. процедур	Nint $\geq D_x$	Nint $< D_x$	Lgi $\leq X_{min}$	Lg1 $> X_{min}$	ПГпосл $\leq X_{max}$	ПГпосл $> X_{max}$
8	нормал.	2	+	-	-	+	+	-

Выполнение работы.

В файле main.cpp расположена функция main, в которой производится вывод подсказок для пользователя, считывание исходных данных и генерация псевдослучайных чисел.

Генерация чисел происходит при помощи генератора mt19937 в соответствии с нормальным распределением, параметры для которого рассчитываются как среднее между границами интервала (математическое ожидание) и одна четвёртая длины интервала (стандартное отклонение).

Далее в функции main вызываются функции unit_distribution и intervals_distribution. Эти функции реализованы на Ассемблере в файлах module1.asm и module2.asm соответственно. Первая функция в качестве результата записывает в переданный ей массив распределение сгенерированных числе по единичным интервалам, а вторая в качестве результата записывает в переданный ей массив распределение сгенерированных числе по пользовательским интервалам.

Принцип работы функции unit_distribution следующий: в цикле просматриваются все сгенерированные числа, далее определяется единичный интервал, в который входит очередное число, вычисляется индекс этого интервала в результирующем массиве, записывается результат.

Принцип работы функции intervals_distribution следующий: в цикле просматриваются все интервалы, вычисляются все единичные интервалы, входящие в этот интервал, во вложенном цикле суммируется количество чисел в единичных интервалах, записывается рульзтат.

После того, как обе функции отработали, результат хранится в массиве result. На ЯВУ составляется таблица в соответствии с результатом, таблица выводится в файл и в консоль, программа завершает свою работу. Исходный код программы представлен в приложении А

Тестирование.

Разработанные тесты представлены и результаты тестирования представлены в приложении Б. По результатам тестирования был сделан вывод, что программа работает верно.

Выводы.

В ходе работы были изучены основные принципы построения собственных прерывания и их вызова из основной программы. Была написана программа, выводящая строку заданное количество раз, после выставляющая задержку на заданное время и выводящая завершающее сообщение.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

main.cpp:

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

using namespace std;

extern "C" void unit_distribution(int* numbers, int n, int* res, int x_min);
extern "C" void intervals_distribution(int* intervals, int n_int, int* units, int n_units, int x_min, int* res);

int main() {
	int n, x_min, x_max, n_int;
	cout << "Enter amount of numbers:" << endl;
	cin >> n;
	cout << "Enter Xmin and Xmax seperated by space:" << endl;
	cin >> x_min >> x_max;
	cout << "Enter number of intervals:" << endl;
	cin >> n_int;
	if (n_int < (x_max - x_min)) {
		cout << "Nint < D_x" << endl;
		return 0;
	}
	cout << "Enter Lgi seperated by spaces:" << endl;
	auto intervals = new int[n_int];
	for (int i = 0; i < n_int + 1; ++i) {
		cin >> intervals[i];
		if (intervals[i] < x_min){
			cout << "lg" << i << " <= X_min" << endl;
			return 0;
		}
	}
	if (intervals[n_int - 1] > x_max) {
		cout << "RBLast > X_max" << endl;
		return 0;
	}


	double mean = (x_min + x_max) / 2;
	double sigma = (x_max - x_min) / 4;
	cout << "\nNumbers generated with normal distribution (mean = " << mean << ", sigma = " << sigma << ").\n" << endl;
	random_device r_d;
	mt19937 generator(r_d());
	normal_distribution<double> distribution(mean, sigma);

	auto numbers = new int[n];
	for (int i = 0; i < n; ++i) {
		int number = distribution(generator);
		while (number < x_min || number > x_max)
			number = distribution(generator);
		numbers[i] = number;
```

```
	//		cout << number << " ";
	//		if ((i + 1) % 50 == 0) cout << "\n";
		}

		auto units = new int[x_max - x_min];
		auto result = new int[n_int];
		for (int i = 0; i < x_max - x_min; ++i)
			units[i] = 0;
		for (int i = 0; i < n_int; ++i)
			result[i] = 0;
		unit_distribution(numbers, n, units, x_min);
		intervals_distribution(intervals, n_int, units, x_max - x_min,
x_min, result);

		ofstream file("table.txt");
		auto head = "N\tLeft border\tAmount of umbers";
		file << head << endl;
		cout << head << endl;
		for (int i = 0; i < n_int; i++) {
			auto row = to_string(i) + "\t" + to_string(intervals[i])
+ "\t\t" + to_string(result[i]) + "\n";
			file << row;
			cout << row;
		}

		return 0;
	}

	module1.asm:
	.MODEL FLAT, C
	.CODE

	PUBLIC C unit_distribution
	unit_distribution PROC C numbers: dword, n: dword, res: dword, xmin:
dword
		mov esi, numbers
		mov edi, res
		mov ecx, n
	start_loop:
		mov eax, [esi]
		sub eax, xmin
		mov ebx, [edi + 4*eax]
		add ebx, 1
		mov [edi + 4*eax], ebx
		add esi, 4
		loop start_loop
	ret
	unit_distribution ENDP
	END

	module2.asm:
	.MODEL FLAT, C
	.CODE

	PUBLIC C intervals_distribution
	intervals_distribution PROC C intervals: dword, n_int: dword, units:
dword, n_units: dword, x_min: dword, res: dword
		mov esi, intervals
```

```
        mov edi, res
        mov ecx, n_int
    start_loop:
        mov eax, [esi]
        add esi, 4h
        mov ebx, [esi]
        sub ebx, eax
        sub eax, x_min

        push ecx
        push esi

        mov ecx, ebx
        mov ebx, 0h
        mov esi, units
        start_loop2:
            add ebx, [esi + eax*4]
            add eax, 1
            loop start_loop2
        mov [edi], ebx
        add edi, 4h

        pop esi
        pop ecx

        loop start_loop
    ret
    intervals_distribution ENDP
END
```

ПРИЛОЖЕНИЕ Б

РАЗРАБОТАННЫЕ ТЕСТЫ

1)

```
Enter amount of numbers:
100
Enter Xmin and Xmax seperated by space:
0 10
Enter number of intervals:
10
Enter Lgi seperated by spaces:
0 1 2 3 4 5 6 7 8 9 10

Numbers generated with normal distribution (mean = 5, sigma = 2).

N       Left border     Amount of umbers
0       0               3
1       1               3
2       2               10
3       3               14
4       4               16
5       5               24
6       6               13
7       7               11
8       8               4
9       9               2
```

2)

```
Enter amount of numbers:
100
Enter Xmin and Xmax seperated by space:
-5 5
Enter number of intervals:
10
Enter Lgi seperated by spaces:
-5 -4 -3 -2 -1 0 1 2 3 4 5

Numbers generated with normal distribution (mean = 0, sigma = 2).

N       Left border     Amount of umbers
0       -5              4
1       -4              1
2       -3              10
3       -2              9
4       -1              12
5       0               30
6       1               17
7       2               9
8       3               6
9       4               2
```

3)

```
Enter amount of numbers:
10000
Enter Xmin and Xmax seperated by space:
-10 5
Enter number of intervals:
15
Enter Lgi seperated by spaces:
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5

Numbers generated with normal distribution (mean = -2, sigma = 3).

N       Left border     Amount of umbers
0       -10             18
1       -9              52
2       -8              129
3       -7              275
4       -6              434
5       -5              667
6       -4              936
7       -3              1205
8       -2              1324
9       -1              1270
10      0               2112
11      1               721
12      2               438
13      3               240
14      4               127
```