МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5
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.

Студентка гр. 0382 _____ Михайлова О.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Изучить связь Ассемблера с ЯВУ на примере построения программы частотного распределения.

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, 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)
2. Диапазон изменения массива псевдослучайных целых чисел [Xmin, Xmax] (м.б. биполярный, например, [-100, 100])
3. Массив псевдослучайных целых чисел $\{X_i\}$.
4. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (<=24)
5. Массив левых границ интервалов разбиения LGrInt

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.

Вариант 24

№	Вид распредения	Число ассем. процедур	Nint ≥ D_x	Nint < D_x	Lgi ≤ X_{min}	Lg1 > X_{min}	ПГпосл ≤ X_{max}	ПГпосл > X_{max}
24	нормал.	2	+	-	+	-	-	+

Выполнение работы.

В функции main программы осуществляется ввод данных и проверка на их корректность. Если введены некорректные данные, на экран выводится соответствующее сообщение об ошибке и программа завершает свою работу. Далее генерируются числа. Нормально распределение осуществляется с помощью функции normal_distribution.

Далее в функции main вызываются функции distribution_1 и distribution_2, описанные на языке Ассемблер в файлах module1.asm и module2.asm соответственно.

Функция distribution_1:

Ей передаются массив сгенерированных чисел arr, длина этого массива, массив result1, в который будут записаны результаты, и Xmin. Рассматривается каждое число массива arr, вычисляется, в какой единичный интервал оно входит, и по соответствующему индексу увеличивается значение в массиве result1 на 1.

Функция distribution_2:

В функцию передаются следующие параметры: массив с границами boarders, количество интервалов, результат выполнения функции distribution_2,

записанный в массиве result1, Xmin, Xmax и массив res, в который будет записан результат. В данной функции происходит распределение чисел по заданным интервалам. Сначала запускается цикл по количеству интервалов. Рассматривается каждая левая граница, записанная в массив boarders. По условию границы могут быть <= Xmax.

Если рассматриваемая левая граница больше Xmin и меньше Xmax , то мы рассматриваем все единичные интервалы, которые входят в заданный интервал , суммируем количество чисел, которое в них входит, и записываем сумму в массив res по соответствующему индексу.

Если рассматриваемая левая граница меньше Xmin и следующая граница тоже меньше или равна Xmin, то происходит переход по метке miss, где мы записываем в массив res по соответствующему индексу значение 0 и далее продолжаем цикл.

Если рассматриваемая левая граница меньше или равна Xmin, а следующая граница больше Xmin, то происходит переход по метке left_boarder, где так же вычисляется, какие единичные интервалы входят в заданный, и суммируется количество чисел, входящих в них.

Если рассматриваемая левая граница равна Xmax, то происходит переход по метке last_boarder, где в массив res по соответствующему индексу записывается количество чисел Xmax в массиве сгенерированных чисел.

В конце программы происходит вывод результатов на экран и запись в файл.

Исходный код программы смотреть в приложении А.

Тестирование программы смотреть в приложении Б.

Выводы.

В ходе работы была изучена связь Ассемблера с ЯВУ на примере построения программы частотного распределения. Была написана программа построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab6.cpp

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

using namespace std;

extern "C" void distribution_1(int* arr, int len_arr, int* res, int Xmin);
extern "C" void distribution_2(int* boarders, int Nint, int* units, int Xmin, int Xmax, int* res);

int main() {

	int len_arr;
	cout << "Enter the length of the aray: ";
	cin >> len_arr;
	int Xmin, Xmax;
	cout << "Enter Xmin: ";
	cin >> Xmin;
	cout << "Enter Xmax: ";
	cin >> Xmax;
	if (Xmax < Xmin) {
		cout << "Xmax < Xmin";
		return 0;
	}
	int Nint;
	cout << "Enter the number of the intervals: ";
	cin >> Nint;
	if (Nint <= 0 || Nint > 24) {
		cout << "Nint must be > 0 and <= 24";
		return 0;
	}
	if (Nint < Xmax - Xmin) {
		cout << "Nint must be >= Dx";
		return 0;
	}


	int* boarders = new int[Nint+1];
	cout << "Enter the left border of each interval: ";
	cin >> boarders[0];
	for (int i = 1; i < Nint; i++) {
		cin >> boarders[i];
		if (boarders[i] > Xmax) {
			cout << "Boarders must be <= Xmax";
			return 0;
		}
	}
	if (boarders[0] > Xmin) {
		cout << "Lg1 mast be <= Xmin";
		return 0;
	}
```

```
for (int i = 0; i < Nint-1; i++) {
     for (int j = i+1; j < Nint; j++) {
          if (boarders[j] < boarders[i]) {
               swap(boarders[j], boarders[i]);
          }
     }
}

boarders[Nint] = Xmax;

random_device rd;
mt19937 gen(rd());
double mean = (Xmax + Xmin)/2;
double stddev = (Xmax - Xmin)/4;
normal_distribution<double> dis(mean, stddev);

int* arr = new int[len_arr];
for (int i = 0; i < len_arr; i++) {
     arr[i] = dis(gen);
     while (arr[i] < Xmin || arr[i] > Xmax) {
          arr[i] = dis(gen);
     }
}

ofstream file("out.txt");
file << "Result: ";
for (int i = 0; i < len_arr; i++) {
     file << arr[i] << ' ';
}
file << '\n';

cout << "Result: ";
for (int i = 0; i < len_arr; i++) {
     cout << arr[i] << ' ';
}
cout << '\n';

int* result1 = new int[Xmax - Xmin+1];
int* result2 = new int[Nint];
for (int i = 0; i < Xmax-Xmin+1; ++i) {
     result1[i] = 0;
}
for (int i = 0; i < Nint; ++i) {
     result2[i] = 0;
}
//int array[] = { 1, 2, 3, 4, 5 };
distribution_1(arr, len_arr, result1, Xmin);
for (int i = 0; i < Xmax-Xmin+1; i++) {
     cout << i + Xmin << ": " << result1[i] << '\t';
}
cout <<'\n';
distribution_2(boarders, Nint, result1, Xmin, Xmax, result2);

cout << "\tn_int\tLgi\tnumbers" << '\n';
file << "n_int\tLgi\tnumbers" << '\n';
for (int i = 0; i < Nint; i++) {
```

```
                cout << "\t" << i + 1 << "\t" << boarders[i] << "\t" <<
result2[i] << '\n';
                file << "\t" << i + 1 << "\t" << boarders[i] << "\t" <<
result2[i] << '\n';
           }
           file.close();
           return 0;
     }
```

Название файла: module1.asm

```
     .586
     .MODEL FLAT, C
     .DATA

     .CODE

     PUBLIC C distribution_1
     distribution_1 PROC C arr: dword, len_arr: dword, res: dword, Xmin:
dword

     push esi
     push edi

     mov esi, arr
     mov edi, res
     mov ecx, len_arr
     mov edx, 0h

     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

     pop edi
     pop esi

     ret
     distribution_1 ENDP
     END
```

Название файла: module2.asm

```
     .MODEL FLAT, C
     .CODE

     PUBLIC C distribution_2
     distribution_2 PROC C boarders: dword, Nint: dword, result1: dword,
Xmin: dword, Xmax: dword, res: dword

          mov esi, boarders
          mov edi, res
          mov ecx, Nint
```

```
start_loop:
	mov eax, [esi]

	cmp eax, Xmax
	je last_border

	add esi, 4h
	mov ebx, [esi]

	cmp ebx, Xmin
	jle miss ;пропускаем
	cmp eax, Xmin
	jle left_border

	sub ebx, eax
	sub eax, Xmin

	push ecx
	push esi

	mov ecx, ebx
	mov ebx, 0h
	mov esi, result1
	loop2:
		add ebx, [esi + eax*4]
		add eax, 1
		loop loop2
	mov [edi], ebx
	add edi, 4h

	pop esi
	pop ecx

	jmp final

	miss:
	    mov eax, 0h
		mov [edi], eax
		add edi, 4h
		jmp final

	left_border:
		mov eax, ebx
		sub eax, Xmin
		sub eax, 1h ;так как вернхняя граница интервала не вклю-
чается

		push ebx
		push esi

		mov ebx, 0h
		mov esi, result1
		loop3:
			add ebx, [esi + eax*4]
			sub eax, 1
			cmp eax, 0
			jge loop3
```

```
            mov [edi], ebx
            add edi, 4h

            pop esi
            pop ebx

            jmp final


      final:
            loop start_loop
            ;учитываем Xmax, если он не указан как левая граница
            mov esi, result1
            mov ebx, [esi + eax*4]
            sub edi, 4h
            add [edi], ebx
            jmp f_final

      last_border: ;учитываем Xmax, если он указан как левая граница
            sub eax, Xmin
            mov esi, result1
            mov ebx, [esi + eax*4]
            add [edi], ebx

      f_final:

ret
distribution_2 ENDP
END
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

```
Enter the length of the aray: 5
Enter Xmin: 1
Enter Xmax: 5
Enter the number of the intervals: 5
Enter the left border of each interval: 0 2 3 4 5
Result: 3 3 2 3 3
1: 0    2: 1    3: 4    4: 0    5: 0
        n_int   Lgi     numbers
        1       0       0
        2       2       1
        3       3       4
        4       4       0
        5       5       0
```

```
Enter the length of the aray: 10
Enter Xmin: -3
Enter Xmax: 3
Enter the number of the intervals: 7
Enter the left border of each interval: -5 -4 -2 -1 0 1 2
Result: 0 0 0 0 -1 0 0 0 0 0
-3: 0   -2: 0   -1: 1   0: 9    1: 0    2: 0    3: 0
        n_int   Lgi     numbers
        1       -5      0
        2       -4      0
        3       -2      0
        4       -1      1
        5       0       9
        6       1       0
        7       2       0
```

```
Enter the length of the aray: 10
Enter Xmin: 1
Enter Xmax: 10
Enter the number of the intervals: 9
Enter the left border of each interval: -3 0 2 3 4 5 6 7 8
Result: 5 7 8 5 3 1 3 6 5 6
1: 1    2: 0    3: 2    4: 0    5: 3    6: 2    7: 1    8: 1    9: 0    10: 0
        n_int   Lgi     numbers
        1       -3      0
        2       0       1
        3       2       0
        4       3       2
        5       4       0
        6       5       3
        7       6       2
        8       7       1
        9       8       1
```