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

Студент гр. 0383 _____ Девятериков И.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Вариант № 4.

Цель работы.

Изучить механизм создания программы с использованием языков высокого уровня и ассемблера.

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 {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.

- Вид распределения — нормальный.
- Число ассемблерных процедур — 2.
- $Nint < Dx$
- $Lgi \leq Xmin$
- Правая граница последнего интервала $Xmax$

Выполнение работы.

На языке C++ производится считывание начальных значений и их проверка, а также генерация чисел с нормальным распределением (с использованием библиотеки random). В реализации программы используется два ассемблерных модуля. Первый модуль содержит функцию first, которая считает количество каждого из чисел исходного массива на единичном интервале и записывает это количество в промежуточный массив. Второй модуль содержит функцию second, которая на основе промежуточного массива считает количество чисел в промежутках. Вывод производится на ЯВУ, также результаты дублируются в файл output.txt.

Разработанный программный код см. **Приложение А.**

Выводы.

В ходе лабораторной работы была реализована программа, в которой представлено взаимодействие ассемблерных модулей с программой на C++.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

main.cpp

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

const int MAX_N = 16000;
const int MAX_NINT = 24;

extern "C" void first(int* num, int NumRunDat, int* res, int Xmin);
extern "C" void second(int* first_res, int NumRunDat, int Xmin, int* borders, int Nint, int* res);



int main() {
	setlocale(LC_ALL, "rus");

	std::ofstream fout;
	fout.open("output.txt", std::ios_base::out);

	int NumRanDat;
	std::cout << "Введите размер массива: ";
	std::cin >> NumRanDat;
	if (NumRanDat > MAX_N) {
		std::cout << "Размер превышает максимальный допустимый размер " << MAX_N << "\n";
		return 0;
```

```
	}

	int Xmin, Xmax;
	std::cout << "Введите диапазон [Xmin, Xmax]: ";
	std::cin >> Xmin >> Xmax;
	int Dx = Xmax - Xmin;

	int Nint;
	std::cout << "Введите количество интервалов: ";
	std::cin >> Nint;
	if ((Nint >= Dx) || (Nint > MAX_NINT) || (Nint < 0)) {
		int min;
		if (MAX_N < Dx) {
			min = MAX_N;
		}
		else {
			min = Dx;
		}
		std::cout << "Количество интервалов должно быть меньше "
<< min << " и больше 0\n";
		return 0;
	}

	int* left_borders = new int[Nint];
	int* saved_borders = new int[Nint];
	std::cout << "Введите левые границы: ";
	for (int i = 0; i < Nint; i++) {
		std::cin >> left_borders[i];
	}
```

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

for (int i = 0; i < Nint; i++) {
	saved_borders[i] = left_borders[i];
	if (left_borders[i] < Xmin) {
		left_borders[i] = Xmin;
	}
}

std::mt19937 gen(time(nullptr));
std::uniform_int_distribution<int> dis(Xmin, Xmax-1);
int* num = new int[NumRanDat];
for (int i = 0; i < NumRanDat; i++)
	num[i] = dis(gen);

std::cout << "Сгенерированные числа: ";
fout << "Сгенерированные числа: ";
for (int i = 0; i < NumRanDat; i++) {
	std::cout << num[i] << " ";
	fout << num[i] << " ";
}
std::cout << "\n";
fout << "\n";
```

```
		int len1 = abs(Xmax - Xmin);
		int* first_res = new int[len1];
		for (int i = 0; i < len1; i++)
			first_res[i] = 0;

		int len2 = Nint + 1;
		int* final_res = new int[len2];
		for (int i = 0; i < Nint + 1; i++)
			final_res[i] = 0;

		first(num, NumRanDat, first_res, Xmin);

		std::cout << "Промежуточные результаты: ";
		NumRanDat = 0;
		for (int i = 0; i < len1; i++) {
			std::cout << first_res[i] << " ";
			NumRanDat += first_res[i];
		}
		std::cout << "\n";

		second(first_res, NumRanDat, Xmin, left_borders, Nint, final_res);

		std::cout << "№\tГраница\tКоличество\n";
		fout << "№\tГраница\tКоличество\n";
		for (int i = 1; i < Nint + 1; i++) {
			std::cout << i << "\t" << saved_borders[i - 1] << "\t" <<
final_res[i] << "\n";
```

```
                fout << i << "\t" << saved_borders[i - 1] << "\t" << final_res[i]
<< "\n";
        }

        delete[] first_res;
        delete[] final_res;
        delete[] left_borders;
        delete[] num;
        fout.close();
    }

    first.asm
    .586p
    .MODEL FLAT, C
    .CODE

    PUBLIC C first
    first PROC C num: dword, N: dword, res: dword, xmin: dword

    push esi
    push edi

    mov edi, num
    mov ecx, N                          ; Íåîáõîäèìî äëÿ ðàáîòû loop - èòåðàöèÿ
ïî ìàññèâó
    mov esi, res                    ; Ïðîìåæóòî÷íûé ìàññèâ

    for_additional_res:
        mov eax, [edi]                  ; Áåðåì î÷åðåäíîé ýëåìåíò èç num
```

```
        sub eax, xmin                   ; Íàéäîì èíäåêñ ÿ÷åéêè,
ñîîòâåòñòâóþùåé ÷èñëó â ïðîìåæóòî÷íîì ìàññèâå
        mov ebx, [esi + 4*eax]    ; Íàõîäèì ÿ÷åéêó ñ ýòèì èíäåêñîì â
ïðîìåæóòî÷íîì ìàññèâå è ïîìåùàì åž â ebx
        inc ebx                                 ; Óâåëè÷èâàì çíà÷åíèå íà 1
        mov [esi + 4*eax], ebx    ; Êëàäåì íîâîå çíà÷åíèå îáðàòíî â ïðîìåæóòî÷íûé
ìàññèâ
        add edi, 4                      ; Ïåðåõîäèì ê ñëåäóþùåìó ýëåìåíòó â
ìàññèâå num
        loop for_additional_res

    pop edi
    pop esi
    ret
    first ENDP
    END

    second.asm
    .586p
    .MODEL FLAT, C
    .CODE
    PUBLIC C second
    second PROC C first_res: dword, N: dword, xmin: dword, borders: dword,
Nint: dword, res: dword

    push esi
    push edi
    push ebp
```

```
        mov edi, first_res
        mov esi, borders
        mov ecx, Nint

        for_borders:
            mov eax, [esi]                  ; Äîñòàžì î÷åðåäíóþ ãðàíèöó
èíòåðâàëà
            sub eax, xmin                   ; Íàõîäèì èíäåêñ ãðàíèöû â
ïðîìåæóòî÷íîì ìàññèâå
            mov [esi], eax                  ; Êëàäžì îáðàòíî
            add esi, 4                      ; Ïåðåõîäèì ê ñëåäóþùåìó
ýëåìåíòó
            loop for_borders

        mov esi, borders
        mov ecx, Nint
        mov ebx, 0                          ; Ñ÷ž÷èê
        mov eax, [esi]                      ; Ïåðâûé ýëåìåíò îáíîâëåííîãî
borders

        for_start:
            push ecx                        ; Ñîõðàíèì çíà÷åíèå ecx
            mov ecx, eax                    ; Ïîìåñòèì â ecx çíà÷åíèå èíäåêñà
ãðàíèöû
            push esi                        ; Ñîõðàíèì esi
            mov esi, res                ; Áóäåì ðàáîòàòü ñ ìàññèâîì res

          for_array:
                cmp ecx, 0
```

```
                je for_end                      ; Åñëè äîñòèãëè êîíöà ãðàíèöû,
âûõîäèì èç öèêëà
            mov eax, [edi]                     ; Áåðì ëåæàùåå â ïðîìåæóòî÷íîì
ìàññèâå êîëè÷åñòâî ýëåìåíòîâ
            add [esi + 4*ebx], eax     ; Äîáàâëÿåì ê ðåçóëüòàòó äëÿ äàííîãî
èíòåðâàëà
            add edi, 4                         ; Ïåðåõîäèì ê ñëåäóþùåìó
            loop for_array

        for_end:
           pop esi                             ; Âîçâðàùàåìñÿ ê ìàññèâó
borders
           inc ebx                             ; Óâåëè÷èâàåì ñ÷ž÷èê
              mov eax, [esi]
              add esi, 4
              sub eax, [esi]                   ; Èç ïðåäûäóùåãî çíà÷åíèÿ
borders âû÷èòàåì ñëåäóþùèé - ïîëó÷àì äëèíó î÷åðåäíîãî èíòåðâàëà * -1
              neg eax                                  ; Äåëàåì çíà÷åíèå
ïîëîæèòåëüíûì
              pop ecx                                  ; Âîçâðàùàåìñÿ ê
èòåðàòîðó Nint
              loop for_start

        mov esi, res
        mov ecx, Nint
        mov eax, 0

        final_for:                             ; Ñ÷èòàåì êîëè÷åñòâî ÷èñåë,
êîòîðûå íå áûëè îáðàáîòàíû
```

```
        add eax, [esi]
        add esi, 4
        loop final_for

    mov esi, res
    sub eax, N
    neg eax

    add [esi + 4*ebx], eax          ; Ïîìåùàåì ýòî êîëè÷åñòâî â ïîñëåäíþþ ÿ÷åéêó ðåçóëüòàòà

    pop ebp
    pop edi
    pop esi

    ret
    second ENDP
    END
```